Parameterized complexity and kernelizability of Max Ones and Exact Ones problems*

Stefan Kratsch[†¶] Dániel Marx[‡‖] Magnus Wahlström[§¶]

October 9, 2015

Abstract

For a finite set Γ of Boolean relations, Max Ones SAT(Γ) and Exact Ones SAT(Γ) are generalized satisfiability problems where every constraint relation is from Γ, and the task is to find a satisfying assignment with at least/exactly k variables set to 1, respectively. We study the parameterized complexity of these problems, including the question whether they admit polynomial kernels. For Max Ones SAT(Γ), we give a classification into 5 different complexity levels: polynomial-time solvable, admits a polynomial kernel, fixed-parameter tractable, solvable in polynomial time for fixed k, and NP-hard already for $k = 1$. For Exact Ones SAT(Γ), we refine the classification obtained earlier by having a closer look at the fixed-parameter tractable cases and classifying the sets Γ for which Exact Ones SAT(Γ) admits a polynomial kernel.

1 Introduction

The constraint satisfaction problem (CSP) provides a framework in which it is possible to express, in a natural way, many combinatorial problems encountered in artificial intelligence and computer science. A CSP instance is represented by a set of variables, a domain of values for each variable, and a set of constraints on the values that certain collections of variables can simultaneously take. The basic aim is then to find an assignment of values to the variables that satisfies the constraints. Boolean CSP (when all variables have domain $\{0, 1\}$) generalizes satisfiability problems such as 2SAT and 3SAT by allowing that constraints are given by arbitrary relations, not necessarily by clauses.

As Boolean CSP problems are NP-hard in general, there have been intensive efforts at finding efficiently solvable special cases of the general problem. One well-studied type of special cases is obtained by restricting the allowed constraint relations to a fixed set Γ; we denote by SAT(Γ) the resulting problem. We expect that if the relations in Γ are simple, then SAT(Γ) is easy to solve. For example, if Γ contains only binary relations, then SAT(Γ) is polynomial-time solvable, as it can be expressed by 2SAT. On the other hand, if Γ contains all the ternary relations, then SAT(Γ) can express 3SAT, and hence it is NP-hard.

A celebrated classical result of T.J. Schaefer [23] characterizes the complexity of SAT(Γ) for *every* finite set Γ: it shows that if Γ has certain simple combinatorial properties, then SAT(Γ) is polynomial-time solvable, and if Γ does not have these properties, then SAT(Γ) is NP-hard. This result is surprising for two reasons. First, Ladner's Theorem [19] states that if P $\neq$ NP, then there are problems in NP that are neither in P nor NP-complete. Therefore, it is surprising that every SAT(Γ) problem is either in P or NP-complete, and no intermediate complexity appears for this family of problems. Second, it is surprising that the borderline between the P and NP-complete cases of SAT(Γ) can be conveniently characterized by simple combinatorial properties.

*An extended abstract of this work appeared in the proceedings of *Mathematical Foundations of Computer Science 2010* [17].

[†]University of Bonn (kratsch@cs.uni-bonn.de)

[‡]Institute of Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI) (dmarx@cs.bme.hu).

[§]Royal Holloway, University of London (Magnus.Wahlstrom@rhul.ac.uk)

[¶]Main work done while at Max-Planck-Institute for Informatics, Saarbrücken, Germany.

[‖]Research supported by the European Research Council (ERC) grant "PARAMTIGHT: Parameterized complexity and the search for tight complexity results," reference 280152 and OTKA grant NK105645.

Schaefer's result has been generalized in various directions. Bulatov [4] generalized it from Boolean CSP to CSP over a 3-element domain and it is a major open question if it can be generalized to arbitrary finite domains (see [5, 13]). Creignou et al. [8] classified the polynomial-time solvable cases of the problem Exact Ones SAT(Γ), where the task is to find a satisfying assignment such that exactly k variables have value 1, for some integer k given in the input. Natural optimization variants of SAT(Γ) have been considered [6, 9, 16] with the goal of classifying the approximability of the different problems. In Max SAT(Γ) we have to find an assignment maximizing the number of satisfied constraints, while in Min UnSAT(Γ) we have to find an assignment minimizing the number of unsatisfied constraints. Min Ones SAT(Γ) and Max Ones SAT(Γ) ask for a satisfying assignment minimizing and maximizing, respectively, the number of variables having value 1.

Parameterized complexity Recently, there have been investigations of the hardness of CSP from the viewpoint of parameterized complexity [20, 18]. This paradigm investigates hardness in finer detail than classical complexity, which focuses mostly on polynomial-time algorithms. A *parameterization* of a problem is assigning an integer k to each input instance. Consider, for example, two standard NP-complete problems Vertex Cover and Clique. Both have the natural parameter k: the size of the required vertex cover/clique. Both problems can be solved in time $n^{\mathcal{O}(k)}$ on n-vertex graphs by complete enumeration. Notice that the degree of the polynomial grows with k, so the algorithm becomes useless for large graphs, even if k is as small as 10. However, Vertex Cover can be solved in time $\mathcal{O}(2^k \cdot n^2)$ [14, 11]. In other words, for every fixed cover size there is a polynomial-time (in this case, quadratic in the number of vertices) algorithm solving the problem where the degree of the polynomial is independent of the parameter. Problems with this property are called *fixed-parameter tractable.* The notion of W[1]-hardness in parameterized complexity is analogous to NP-completeness in classical complexity. Problems that are shown to be W[1]-hard, such as Clique [14, 11], are very unlikely to be fixed-parameter tractable.

Kernelization One of the most basic techniques for showing that a problem is fixed-parameter tractable is to show that the computationally hard "core" of the problem can be extracted in polynomial time. Formally, *kernelization* is a polynomial-time transformation that, given an instance I of problem $\mathcal{Q}$ with parameter k, creates an equivalent instance I' of problem $\mathcal{Q}$ with parameter $k' \leq f(k)$ such that the size of I' is at most $g(k)$ for some functions f, g (usually, $k' \leq k$ is achievable). For example, a classical result of Nemhauser and Trotter [21] shows that every instance I of Vertex Cover with parameter k can be transformed into an instance I' with parameter $k' \leq k$ such that I' has at most $g(k) = 2k$ vertices. Observe that the existence of a kernelization algorithm for $\mathcal{Q}$ immediately implies that $\mathcal{Q}$ is FPT, assuming that $\mathcal{Q}$ is decidable: performing the kernelization and then doing a brute force solution on I' clearly takes only $n^{\mathcal{O}(1)} + f(k)$ time for some function f. From the practical point of view, *polynomial kernels,* i.e., kernelization algorithms where $g(k)$ is a polynomial, are of particular interest. If a problem has this property, then this means that there is an efficient preprocessing algorithm for the problem with a provable bound on the way it shrinks the instance. Such a preprocessing can be an invaluable opening step in any practical solution for the problem. Very recently, however, it has been shown that under standard complexity assumptions, not every FPT problem has a polynomial kernel: e.g., the Path(k) problem can be solved in (randomized) time $2^k \cdot n^{\mathcal{O}(1)}$ [24], but has no polynomial kernel unless NP $\subseteq$ co-NP/poly [2]. The negative toolkit developed in [2] has been successfully applied to a number of other problems [3, 10].

Results The parameterized complexity of Exact Ones SAT(Γ) was studied in [20], where it was shown that a property called weak separability characterizes the complexity of the problem: Exact Ones SAT(Γ) is FPT if Γ is weakly separable, and W[1]-complete otherwise. The problem Min Ones SAT(Γ) is FPT for every Γ by a simple branching algorithm, but it is not obvious to see for which Γ there is a polynomial kernel. This question has been resolved in [18] by showing that (unless NP $\subseteq$ co-NP/poly) Min Ones SAT(Γ) has a polynomial kernel if and only if Min Ones SAT(Γ) is in P or Γ has a property called mergeability.

We continue this line of research by considering the so far unexplored problem Max Ones SAT(Γ) and revisit Exact Ones SAT(Γ). We characterize (under standard complexity assumptions) parameterized Max Ones SAT(Γ) problems for finite constraint languages Γ as the following 5 types: solvable

Γ	Min Ones	Exact Ones	Max Ones
width-2 affine	P	P	P
$\{\text{ODD}_3\}$	PK	PK	P
$\{\text{EVEN}_3\}$	P	FPT	PK
$\{\text{EVEN}_3, (x)\}$	FPT	FPT	PK
$\{\text{ODD}_4\}$, general affine	FPT	FPT	PK
$\{(x \vee y), (x \neq y)\}$	PK	PK	PK
$\{((x \to y) \wedge (y \neq z))\}$	PK	FPT	FPT
$\{(x \vee y), (x \neq y), (x \to y)\}$	PK	W[1]-complete	FPT
bijunctive	PK	W[1]-complete	W[1]-hard, XP
$\{R_{1\text{-IN-}3}\}$	PK	PK	not in XP
$\{\sum_i x_i = p \pmod q\}$	FPT	FPT	not in XP
general	FPT	W[1]	not in XP

Table 1: Examples of sets of relations Γ and the properties for Min Ones SAT(Γ), Exact Ones SAT(Γ), and Max Ones SAT(Γ). Problems marked PK have polynomial kernels; problems marked FPT are FPT but admit no polynomial kernelization unless NP $\subseteq$ co-NP/poly.

in polynomial time; NP-hard, but having polynomial kernelization; being FPT but admitting no polynomial kernelization; being W[1]-hard and in XP; and not being in XP. For Exact Ones SAT(Γ), we refine the classification of [20] by precisely characterizing those weakly separable sets Γ for which Exact Ones SAT(Γ) is not only FPT, but admits a polynomial kernel. Table 1 shows some examples.

Overview of the paper Since the complexity characterizations of Max Ones SAT(Γ) and Exact Ones SAT(Γ) use some quite technical tools, we defer overviews of the proofs to Sections 3 and 4, respectively. However, let us say a few words here about the difference between the two proofs, in terms of the tools and resulting characterizations, and their relation to previous work.

For Max Ones SAT(Γ), we may observe that if SAT(Γ) is NP-hard, then so is Max Ones SAT(Γ) with parameter $k = 0$, i.e., for such a language Γ, Max Ones SAT(Γ) is not even in XP (let alone FPT) unless P=NP. Hence we may restrict our attention to those polynomial-time solvable cases identified by Schaefer [23]. This allows us to use powerful known results about the structure and expressiveness of such languages; in particular, we use the theory of *frozen partial co-clones* due to Nordh and Zanuttini [22] (see Section 3). This saves us a lot of technical work, and lets us give the resulting characterization essentially as a list of maximal positive cases.

For Exact Ones SAT(Γ), on the other hand, there are positive cases (i.e., languages for which Exact Ones admits a polynomial kernel) even among languages Γ such that SAT(Γ) is NP-hard. Since no useful structural characterization is known for such languages (at least, not to the detail that we would need), we need to resort to more low-level tools, and work directly with algebraic closure operations called *partial polymorphisms* (see Section 4). These are the same tools with which previous results have been derived [20, 18]. On the other hand, since both a P-vs-NP dichotomy and an FPT-vs-W[1]-dichotomy for Exact Ones SAT(Γ) is already known [8, 20], it only remains for us to state a dichotomy for the kernelizability of Exact Ones SAT(Γ). In this, we build explicitly on previous work [20, 18]. Concretely, we define a new relational property called *semi-separability*, which is a sharpening of weak separability, and conclude (through a list of cases) that Exact Ones SAT(Γ) admits a polynomial kernel if and only if Γ is semi-separable and mergeable, unless NP $\subseteq$ co-NP/poly.

The paper is structured as follows. In Section 2, we provide some standard definitions and preliminary results. We also define a maximization problem Multiple Compatible Patterns (MCP), which will be useful in the kernelization lower bounds for both the Exact Ones and Max Ones problems. We show that MCP is FPT, but admits no polynomial kernelization unless NP $\subseteq$ co-NP/poly. In Section 3, we give the complexity characterization for the Max Ones SAT(Γ) problem, and in Section 4 we complete the complexity characterization for Exact Ones SAT(Γ). We wrap the paper up in Section 5 with conclusions and further remarks.

2 Preliminaries and Notation

Boolean CSP A *formula* ϕ is a pair (V, C) consisting of a set V of *variables* and a set C of *constraints*. Each constraint $c_i \in C$ is a pair $\langle \overline{s}_i, R_i \rangle$, where $\overline{s}_i = (x_{i,1}, \ldots, x_{i,r_i})$ is an r_i-tuple of variables (the *constraint scope*) and $R_i \subseteq \{0,1\}^{r_i}$ is an r_i-ary Boolean relation (the *constraint relation*). A function $f : V \to \{0,1\}$ is a *satisfying assignment* of ϕ if $(f(x_{i,1}), \ldots, f(x_{i,r_i}))$ is in R_i for every $c_i \in C$. Let Γ be a set of Boolean relations. A formula is a Γ-*formula* if every constraint relation R_i is in Γ. In this paper, Γ is always a finite set containing only non-empty relations. For a fixed finite Γ, every Γ-formula $\phi = (V, C)$ can be represented with length polynomial in $|V|$ and $|C|$: if r is the maximum arity of the relations in Γ, then the maximum number of different constraints is $|\Gamma| \cdot |V|^r$, and each constraint relation can be represented by constant number of bits (depending only on Γ). The *weight* $w(f)$ of an assignment f is the number of variables x with $f(x) = 1$.

We also use some definitions from Nordh and Zanuttini [22]. Let $\phi = (V, C)$ be a formula and $x \in V$ a variable. Then x is said to be *frozen* in ϕ if x takes the same value in every satisfying assignment of ϕ. Further, let Γ be a set of relations, and R an n-ary relation. Then Γ *freezingly implements* R if there is a formula ϕ over $\Gamma \cup \{=\}$ such that $R(x_1, \ldots, x_n) \equiv \exists X \phi$, where ϕ uses variables $X \cup \{x_1, \ldots, x_n\}$ only, and all variables in X are frozen in ϕ. If only relations of Γ are used, then we have a *frozen implementation without equality*. This is our standard notion of implementation in the paper, and as such is shortened to simply "implements" (though see Lemma 2.2 below).

We recall some standard definitions concerning Boolean constraints (cf. [6]):

- R is *0-valid* if $(0, \ldots, 0) \in R$.
- R is *1-valid* if $(1, \ldots, 1) \in R$.
- R is *Horn* or *weakly negative* if it can be expressed as a conjunction of clauses such that each clause contains at most one positive literal. It is known that R is Horn if and only if it is *AND-closed:* if $(a_1, \ldots, a_r) \in R$ and $(b_1, \ldots, b_r) \in R$, then $((a_1 \wedge b_1), \ldots, (a_r \wedge b_r)) \in R$.
- R is *anti-Horn* or *weakly positive* if it can be expressed as the conjunction of clauses such that each clause contains at most one negated literal. It is known that R is anti-Horn if and only if it is *OR-closed:* if $(a_1, \ldots, a_r) \in R$ and $(b_1, \ldots, b_r) \in R$, then $((a_1 \vee b_1), \ldots, (a_r \vee b_r)) \in R$.
- R is *bijunctive* if it can be expressed as the conjunction of constraints such that each constraint is the disjunction of two literals. It is known that R is bijunctive if and only if it is closed under *majority:* Let $maj(x, y, z) : \{0,1\}^3 \to \{0,1\}$ be such that $maj(x, x, y) = maj(x, y, x) = maj(y, x, x) = x$ (note that this fully defines maj). Then R is *majority-closed* if for any $(a_1, \ldots, a_r), (b_1, \ldots, b_r), (c_1, \ldots, c_r) \in R$ we have
$$(maj(a_1, b_1, c_1), \ldots, maj(a_r, b_r, c_r)) \in R.$$
- R is *affine* if it can be expressed as a conjunction of constraints of the form $x_1 + x_2 + \cdots + x_t = b$, where $b \in \{0,1\}$ and addition is modulo 2. The number of tuples in an affine relation is always an integer power of 2. We denote by EVEN_r the r-ary relation $x_1 + x_2 + \cdots + x_r = 0$ and by ODD_r the r-ary relation $x_1 + x_2 + \cdots + x_r = 1$.
- R is *width-2 affine* if it can be expressed as a conjunction of constraints of the form $x = y$, $x \neq y$, (x), and $(\neg x)$.
- R is *monotone* if $a \in R$ and $b \geq a$ implies $b \in R$, where $\geq$ is applied component-wise. Such a relation is implementable by positive clauses, by adding a clause over the false positions of every maximal false tuple.
- The relation $R_{p\text{-IN-}q}$ (for $1 \leq p \leq q$) has arity q and $R_{p\text{-IN-}q}(x_1, \ldots, x_q)$ is true if and only if exactly p of the variables $x_1, \ldots, x_q$ have value 1.

See also Creignou et al. [7]. The above is extended to properties of sets of relations, by saying that a set of relations Γ is 0-valid (1-valid, Horn, ...) if this holds for every $R \in \Gamma$.

Theorem 2.1 (Schaefer [23])**.** *Let Γ be a set of Boolean relations. Then* $\mathrm{SAT}(\Gamma)$ *is in* P *if Γ has one of the following properties: 0-valid, 1-valid, Horn, anti-Horn, bijunctive, or affine. Otherwise,* $\mathrm{SAT}(\Gamma)$ *is* NP*-complete.*

Problem definitions For a fixed set of relations Γ, Max Ones SAT(Γ) is the following problem:

Max Ones SAT(Γ)

Input: A formula ϕ over Γ; an integer k.

Parameter: k.

Task: Decide whether there is a satisfying assignment for ϕ of weight at least k.

For example, Max Ones SAT($\neg x \vee \neg y$) is equivalent to Independent Set, and is thus W[1]-complete. Further examples can be found in Table 1. Similarly, Exact Ones SAT(Γ), for a fixed set of relations Γ, is the following problem.

Exact Ones SAT(Γ)

Input: A formula ϕ over Γ; an integer k.

Parameter: k.

Task: Decide whether there is a satisfying assignment for ϕ of weight exactly k.

Parameterized complexity and kernelization A parameterized problem $\mathcal{Q}$ is a subset of $\Sigma^* \times \mathbb{N}$; the second component is called the *parameter*. The problem $\mathcal{Q}$ is *fixed-parameter tractable* (FPT) if there is an algorithm that decides $(I,k) \in \mathcal{Q}$ in time $f(k) \cdot n^{\mathcal{O}(1)}$, where f is some computable function. A *kernelization* is a polynomial-time mapping $K : (I,k) \mapsto (I',k')$ such that (I,k) and (I',k') are equivalent, $k' \leq f(k)$, and $|I'| \leq g(k)$, for some functions f and g. Usually, f can be taken as the identity function, i.e., $k' \leq k$; this will be the case throughout this paper. If $|I'|$ is bounded by a polynomial in k, then K is a *polynomial kernelization*. It is well-known that every decidable parameterized problem is fixed-parameter tractable if and only if it has a (not necessarily polynomial) kernelization [14]. A *polynomial parameter transformation* (short PPT) from $\mathcal{Q}$ to $\mathcal{Q}'$ is a polynomial-time mapping $\Phi : (I,k) \mapsto (I',k')$ such that $(I,k) \in \mathcal{Q}$ if and only if $(I',k') \in \mathcal{Q}$ and such that k' is polynomially bounded in k; we denote the existence of such a reduction by $\mathcal{Q} \leq_{ppt} \mathcal{Q}'$. These reductions were introduced by Bodlaender et al. [3], who also showed that they preserve polynomial kernelizability. If $\mathcal{Q} \leq_{ppt} \mathcal{Q}'$ and $\mathcal{Q}' \leq_{ppt} \mathcal{Q}$, then $\mathcal{Q}$ and $\mathcal{Q}'$ are *equivalent under PPT's*, written $\mathcal{Q} \equiv_{ppt} \mathcal{Q}'$.

Lemma 2.2. *Let Γ be a set of Boolean relations, and R an r-ary Boolean relation such that R has an implementation $\exists X \phi$ over Γ (using equality only if $=\in \Gamma$). Then* Max Ones SAT(Γ) $\equiv_{ppt}$ Max Ones SAT($\Gamma \cup \{R\}$) *and* Exact Ones SAT(Γ) $\equiv_{ppt}$ Max Ones SAT($\Gamma \cup \{R\}$). *In particular, this holds if Γ freezingly implements R and R has no repeated columns (i.e., there are no $i,j \in [r]$, $i \neq j$ such that $\alpha(i) = \alpha(j)$ for every $\alpha \in R$).*

Proof. We first claim that if R has no repeated columns, then there is an implementation of R that does not use equality. Indeed, consider an implementation $R(x_1,\ldots,x_r) \equiv \exists X \phi_R$ with a minimum number of frozen variables X. Discard any equality constraints $(x = x)$ in ϕ_R, as they don't restrict the set of solutions. Then ϕ_R cannot contain an equality $(x_i = x_j)$ for $i,j \in [r]$, as this would (by assumption) exclude some tuple $\alpha \in R$. But there also cannot be an equality constraint $(x = x')$ where $x \in X$, as we could instead simply replace every occurrence of x by x', creating an implementation with fewer variables in X. Hence ϕ_R does not contain any equality constraints. Thus, we may assume that there is a frozen implementation $\exists X \phi_R$ of R using only constraints from Γ. We further assume $|X| \leq 2$, i.e., the most that X can contain is one variable z_0 frozen to 0 and one variable z_1 frozen to 1.

Now let $\phi = (V,C)$ be a $\Gamma \cup \{R\}$-formula. If $z_1 \in X$, create a global variable c_1, and if $z_0 \in X$, create a global variable c_0. If ϕ contains no occurrence of R, then we are done. Otherwise we replace every occurrence of R in ϕ by the implementation ϕ_R, replacing each occurrence of a variable z_1 by c_1 and each occurrence of a variable z_0 by c_0. Note that by assumption, this creates a formula where the new global variables are frozen to $c_1 = 1$ and $c_0 = 0$. Finally, if we have created a variable c_1, then we increase k by 1. This is a reduction preserving the entire solution set to the original formula ϕ, hence a PPT for both Max Ones SAT($\Gamma \cup \{R\}$) and Exact Ones SAT($\Gamma \cup \{R\}$). Note that a PPT in the other direction is trivial, e.g., a Γ-formula ϕ is already a valid Γ'-formula for any $\Gamma' \supseteq \Gamma$. Hence the problems are equivalent under PPTs. □

The MCP problem Our kernelization lower bounds use the problem Multiple Compatible Patterns (MCP), defined as follows:

> **Multiple Compatible Patterns**
>
> **Input:** A set of patterns from $\{0, 1, \bigstar\}^r$, where $\bigstar$ (the wildcard character) matches 0 or 1; an integer k.
>
> **Parameter:** $r + k$.
>
> **Task:** Decide whether there is a string in $\{0, 1\}^r$ that matches at least k patterns.

For the kernelization lower bound, we give a PPT from Multicolored Clique($k \log n$), the usual Multicolored Clique problem under a parameter of $p = k \log n$. The latter problem admits no polynomial kernel unless NP $\subseteq$ co-NP/poly, and furthermore was used by Hermelin et al. [15] as one of the complete problems for the "kernelization hardness" class MK[1]; see the discussion in Section 5.

Lemma 2.3. Multiple Compatible Patterns *is* FPT*, NP-complete, and admits no polynomial kernelization unless* NP $\subseteq$ co-NP/poly *(and the polynomial hierarchy collapses).*

Proof. The problem is trivially FPT, by enumeration of all 2^r possible strings. We show the remaining results via a reduction from Multicolored Clique($k \log n$). We assume w.l.o.g. that every color class has cardinality exactly n, and that there are k classes; let the members of color class i be ordered as $v_{i,1}, \ldots, v_{i,n}$. Our patterns have length $k \log n$, divided into k chunks of length $\lceil \log n \rceil$ each; each such chunk encodes a choice of vertex in a color class. Our patterns are created as follows: For every edge $v_{i,p}v_{j,q}$ (where necessarily $i \neq j$), we create a pattern where chunk i encodes p in binary, chunk j encodes q in binary, and all other chunks are filled with $\bigstar$ (wildcards). We claim that there are $\binom{k}{2}$ compatible patterns if and only if the MCC instance is positive.

On the one hand, let C be a k-clique in G. Then C makes one selection in each color class. The pattern where each chunk i encodes (in binary) the selection that C makes in color class i will then match each of the patterns corresponding to the $\binom{k}{2}$ edges in C.

On the other hand, let $p \in \{0, 1\}^r$ be a pattern. It encodes a choice of one vertex in each color class; let this set of vertices be C. Observe that by construction, for each pair of color classes i, j, at most one pattern at a time can be matched. Therefore, it is only possible to match $\binom{k}{2}$ patterns if for every pair of color classes i, j, the choices of C in i and j are adjacent, i.e., if C forms a clique. This completes the proof. □

3 Max Ones Characterization

In this section we present our characterization of the parameterized complexity properties of Max Ones SAT(Γ) problems. As a very first distinction, we repeat the observation that if SAT(Γ) is NP-complete, then Max Ones SAT(Γ) is NP-complete even for a parameter $k = 0$: Clearly, for any formula ϕ there is a satisfying assignment with at least zero true variables if and only if ϕ is satisfiable. By Schaefer's characterization of SAT(Γ) (Theorem 2.1), SAT(Γ) is in P if Γ is zero-valid, one-valid, affine, Horn, anti-Horn, or bijunctive, and NP-complete otherwise. In the latter case Max Ones SAT(Γ) is not contained in XP, i.e., there is no $n^{f(k)}$ time algorithm, unless P = NP. Therefore, in the following we consider Max Ones SAT(Γ) for the six maximal constraint languages for which SAT(Γ) is in P.

We begin with the polynomial cases, as proven by Khanna et al. [16].

Theorem 3.1 ([16])**.** Max Ones SAT(Γ) *is in* P *if* Γ *is 1-valid, weakly positive (i.e. anti-Horn), or width-2 affine, and APX-hard in all other cases.*

In the following sections we address the remaining four maximal cases from Schaefer's characterization, namely affine, Horn, zero-valid, and bijunctive languages. Among those, the bijunctive case turns out to be the most involved one, as it contains cases of several different types (with somewhat involved relations). We address this case using Nordh and Zanuttini's [22] notion of frozen partial co-clones (see later).

3.1 Affine cases

We show that for every affine constraint language Γ the Max Ones SAT(Γ) problem admits a polynomial kernelization with a linear number of variables.

Lemma 3.2. *Let Γ be an affine constraint language. Then* Max Ones SAT(Γ) *admits a kernelization with $\mathcal{O}(k)$ variables (and polynomial total size).*

Proof. Let (ϕ, k) be an instance of Max Ones SAT(Γ); if ϕ is infeasible, we return a dummy negative instance. We call a variable *fixed* if it has the same value in every satisfying assignment. It can be checked in polynomial time if a variable is fixed to true (resp., false) by testing satisfiability with the variable set to false (resp., true).

If two variables x and y are fixed to the same value $d \in \{0, 1\}$, then we can obtain an equivalent instance with one less variable: we replace all occurrences of y by x and decrease the parameter by 1 if $d = 1$. Clearly, this does not permit x to take a value different from d (i.e., $1 - d$), else both x and y would not have been fixed.

Let (ϕ', k') be the instance obtained after replacing all but at most two fixed variables (one fixed to true and one fixed to false). If ϕ' contains less than $2k' + 2 \in \mathcal{O}(k)$ variables, then we return (ϕ', k') as a kernel. Otherwise, we show that (ϕ', k') has a satisfying assignment of weight at least k', in which case the kernelization algorithm can return a dummy positive instance. We begin by replacing the at most two fixed variables by the corresponding value, i.e., 0 or 1. We retain at least $2k'$ variables and apply the following procedure.

We pick an arbitrary variable x of ϕ'. Substituting x with 0 (resp., 1) makes a set S_0 (resp., S_1) of variables fixed. We claim that $S_0 = S_1$ (and hence we will denote $S_0 = S_1$ by S). If, say, $y \in S_0 \setminus S_1$, then the projection of ϕ' to $\{x, y\}$ would be a binary relation having exactly 3 tuples. However, the projection of an affine instance is an affine relation, which cannot have exactly 3 tuples (as the size of an affine subspace over the 2-element field is always a power of 2). Thus $S_0 = S_1$ as claimed. Furthermore, let $y \in S$; since y is not fixed, substituting x with 0 and 1 cannot force y to the same value. Therefore, one of the two substitutions forces at least half of the variables in $S \cup \{x\}$ to 1. Let us perform this substitution and remove the variables in $S \cup \{x\}$ by substituting them with the forced values. Observe that the resulting formula is still feasible and has no fixed variables (it may contain constants).

Let us repeat the procedure described in the previous paragraph until at least $2k'$ variables are removed. This means that at least k' variables are substituted with the value 1. Therefore, any solution of the remaining (feasible) instance, extended with the substituted values of the removed variables, gives a satisfying assignment of weight at least k' for (ϕ', k'). □

3.2 Horn cases

For constraint languages that are Horn but neither anti-Horn nor one-valid we show that Max Ones SAT(Γ) is W[1]-hard by a parameterized reduction from the W[1]-complete problem Independent Set. Recall that Max Ones SAT(Γ) is in P if Γ is anti-Horn or one-valid.

Lemma 3.3. *If Γ is Horn, but neither anti-Horn nor one-valid, then* Max Ones SAT(Γ) *is* W[1]*-hard.*

Proof. We first show that a constant zero variable c_0 can be created (which takes value zero (false) in any satisfying assignment). Let $R \in \Gamma$ be a relation which is not one-valid. If R is zero-valid, then putting variable c_0 into all positions makes c_0 a constant zero variable. If R is not zero-valid, then let $\alpha = (\alpha_1, \ldots, \alpha_r) \in R$ be a minimal tuple of R (w.r.t. number of ones). Put a variable c_1 in all positions that are true in α, and c_0 in all other positions. This forces $c_0 = 0$ and $c_1 = 1$: First, it is not possible that $c_0 = c_1$, since R is neither zero-valid nor one-valid. Second, it is not possible that $c_0 = 1$ and $c_1 = 0$, since R is Horn (i.e., invariant under conjunction) and $c_0 = 0$ and $c_1 = 1$ satisfies R, implying that $c_0 = c_1 = 0$ would also satisfy R. Thus, if R is not zero-valid, we can get the constants zero c_0 and one c_1 (in both cases we have constant zero).

Now we select a relation $R \in \Gamma$ that is not anti-Horn. Let $\alpha, \beta \in R$ be tuples such that $\alpha \vee \beta \notin R$, and group the positions of R according to their values in α and β into positions A through D. Note that $\alpha \wedge \beta \in R$ since R is Horn:

	A	B	C	D	
α	0	0	1	1	$\in R$
β	0	1	0	1	$\in R$
$\alpha \wedge \beta$	0	0	0	1	$\in R$
$\alpha \vee \beta$	0	1	1	1	$\notin R$

Observe that both positions B and C must occur to distinguish the excluded tuple $\alpha \vee \beta$ from α and β, respectively. If there are no positions of type D then we implement the independent set constraint $(\neg x \vee \neg y)$ by putting the constant c_0 into positions A (if they exist), x into positions B, and y into positions C. Thus we have a reduction from Independent Set to Max Ones SAT(Γ).

If positions of type D do exist, then we create a ternary relation R' by putting variables x, y, and z into all positions B, C, and D, respectively, as well as putting c_0 into positions A:

x	y	z	
0	1	1	$\in R'$
1	0	1	$\in R'$
0	0	1	$\in R'$
1	1	1	$\notin R'$

Then R' is not one-valid. If R' is not zero-valid either, then we have seen above that we can create a variable $c_1 = 1$. Putting this variable c_1 in place of z implements $(\neg x \vee \neg y)$, and we have a reduction from Independent Set.

Thus assume for the rest of the proof that R' is zero-valid, i.e., $(0,0,0) \in R'$. We observe that if $(1,0,0) \in R'$ then $R'(x,y,y) = (\neg x \vee \neg y)$. Similarly, if $(1,1,0) \in R'$ then $R'(x,x,y) = (\neg x \vee \neg y)$. In both cases we again have an immediate reduction from Independent Set. Finally, if $(1,0,0), (1,1,0) \notin R'$ then we have

x	y	z	
0	0	0	$\in R'$
0	1	1	$\in R'$
1	0	1	$\in R'$
0	0	1	$\in R'$
1	0	0	$\notin R'$
1	1	0	$\notin R'$
1	1	1	$\notin R'$

We implement $R''(x,y,z) = R'(x,y,z) \wedge R'(y,x,z)$ and observe that

$$R''(x,y,z) = \{(0,0,0), (0,0,1), (0,1,1), (1,0,1)\}.$$

This relation permits us to give a reduction from Independent Set. We create a global variable $\hat{z}$. For each $(\neg x \vee \neg y)$ that we need to implement, we add a constraint $R''(x,y,\hat{z})$. Now every non-zero satisfying assignment assigns true to $\hat{z}$. Thus an independent set of size k corresponds to a satisfying assignment with $k+1$ true variables and vice versa.

In each case, we were able to reduce the W[1]-complete Independent Set problem to Max Ones SAT(Γ), which proves the lemma. □

3.3 Zero-valid cases

For zero-valid constraint languages Γ that are not covered by the previously considered cases, we show that Max Ones SAT(Γ) is not in XP unless P = NP.

Lemma 3.4. *If Γ is zero-valid, but neither anti-Horn, one-valid, affine, nor Horn, then* Max Ones SAT(Γ) *is not in* XP *unless* P = NP.

Proof. The proof is organized in two parts. First we show that we can implement the relation $R = \{(0,0,0), (0,1,1), (1,0,1)\}$. Second we reduce the NP-complete SAT($\{R, (x \neq y)\}$) problem to Max Ones SAT(Γ) instances with parameter $k = 1$, proving the lemma. We will need a constant zero variable c_0, which can be implemented by taking a (zero-valid) relation R_0 that is not one-valid, and making a constraint $R_0(c_0, \ldots, c_0)$.

Implementing R. Note that a zero-valid relation is affine if and only if it is invariant under exclusive disjunction (XOR); this can be easily seen since affine relations are invariant under ternary XOR. Let R_1, R_2, R_3 be relations that are not Horn, not anti-Horn, and not XOR-closed, respectively. Let us choose a pair α_1, β_1 of tuples witnessing that R_1 is not invariant under conjunction, i.e., $\alpha_1, \beta_1 \in R_1$ but $a_1 \wedge b_1 \notin R_1$. Similarly choose a pair α_2, β_2 violating invariance under disjunction on R_2, and a pair α_3, β_3 violating XOR-invariance on R_3.

We introduce three variables x, y, z, and apply constraints R_1, R_2, R_3 on these variables by putting x (resp., y or z) at a position of constraint R_ℓ if at this position tuple α_ℓ has value 0 and tuple β_ℓ has value 1 (resp., 1 and 0 or 1 and 1). Further, we put c_0 in R_ℓ where α_ℓ and β_ℓ are both 0. That is, we create the following constraints.

	c_0	x	y	z	
α_1	0	0	1	1	$\in R_1$
β_1	0	1	0	1	$\in R_1$
$\alpha_1 \wedge \beta_1$	0	0	0	1	$\notin R_1$

	c_0	x	y	z	
α_2	0	0	1	1	$\in R_2$
β_2	0	1	0	1	$\in R_2$
$\alpha_2 \vee \beta_2$	0	1	1	1	$\notin R_2$

	c_0	x	y	z	
α_3	0	0	1	1	$\in R_3$
β_3	0	1	0	1	$\in R_3$
$\alpha_3 \oplus \beta_3$	0	1	1	0	$\notin R_3$

Note that a constraint might not contain all three variables x, y, z, but this will not cause any problems in the arguments to follow. Let us point out, however, that each variable occurs in at least one of the constraints: The variables x and y are placed at least into R_2 since their positions distinguish α_2 and β_2 from $\alpha_2 \vee \beta_2$. The variable z is at least placed in R_1 since those positions distinguish $0 \in R_1$ from $\alpha_1 \wedge \beta_1 \notin R_1$. Thus the conjunction of the three constraints contains all three variables x, y, and z.

Let us consider the constraint $R'(x, y, z)$ implemented this way. Since every relation is zero-valid, we have $(0, 0, 0) \in R'$. The chosen tuples and the way R' was constructed ensure that $(0, 1, 1), (1, 0, 1) \in R'$. Since $\alpha_1 \wedge \beta_1 \notin R_1$, the constraint R_1 ensures that $(0, 0, 1) \notin R'$. Similarly, R_2 and R_3 ensure that R' does not contain $(1, 1, 1)$ and $(1, 1, 0)$, respectively. Thus we know about the following included and excluded tuples:

0	0	0	$\in R'$
0	1	1	$\in R'$
1	0	1	$\in R'$
0	0	1	$\notin R'$
1	1	0	$\notin R'$
1	1	1	$\notin R'$

If $(0, 1, 0), (1, 0, 0) \notin R'$, then R' is the relation R that we wanted to obtain. Suppose that $(0, 1, 0) \in R'$. Then $R'(c_0, b, a)$ implements $(a \to b)$ and thus we get an implementation of the desired relation R as $R'(x, y, z) \wedge (x \to z) \wedge (y \to z)$. Note that $(x \to z)$ excludes $(1, 0, 0)$ and $(y \to z)$ excludes $(0, 1, 0)$; both are consistent with R.

If $(1, 0, 0) \in R'$, then $R'(b, c_0, a)$ implements $(a \to b)$ and we obtain R with the same formula (using, e.g., $R'(z, c_0, x)$ for $(x \to z)$).

Reduction. Let $\Gamma' = \{R, (x \neq y)\}$. Clearly, R is neither closed under conjunction nor disjunction; hence it is neither Horn nor anti-Horn. Note also that R is also not bijunctive or affine, since $(0, 0, 0), (0, 1, 1), (1, 0, 1)$ are a witness against invariance under majority and 3-way XOR. Clearly, disequality, $(x \neq y)$, is neither zero-valid nor one-valid. Therefore, Γ' is neither zero-valid, one-valid, Horn, anti-Horn, bijunctive, nor affine, which implies that SAT(Γ') is NP-complete (Theorem 2.1, i.e., Schaefer's dichotomy theorem). We will reduce SAT(Γ') to Max Ones SAT(Γ) with parameter $k = 1$.

Let ϕ' be a formula over Γ'. Assume that ϕ' is false for the all-zero assignment, otherwise we may reduce to a dummy positive instance. We will create a formula ϕ using only R, which has a non-zero solution if and only if ϕ' is satisfiable: Copy all constraints using R from ϕ' to ϕ, and add a single global variable $\hat{z}$. For every variable x in ϕ', create a variable x' intended to be its negation, and add a constraint $R(x, x', \hat{z})$. Additionally for every constraint $(x \neq y)$ in ϕ', create a constraint $R(x, y, \hat{z})$ in ϕ. Now every solution to ϕ where any variable is true must have $\hat{z} = 1$, which "activates" all inequalities

from ϕ'. Thus ϕ' is satisfiable if and only if ϕ has a satisfying assignment with at least one true variable, i.e., if $(\phi, 1) \in$ Max Ones SAT(Γ). □

3.4 Bijunctive cases

Concluding the characterization of Max Ones SAT(Γ) problems, we address the remaining case of bijunctive constraint languages, which (additionally) are not Horn, anti-Horn, or width-2 affine (from which it follows that Γ is not zero-valid or one-valid; see Lemma 3.6). This corresponds to the constraint languages Γ that, using existentially quantified variables, can implement all 2-SAT clauses; see Nordh and Zanuttini [22]. We find that the classification is not the same for these languages. There are cases that create problems which admit polynomial kernelizations, problems which are fixed-parameter tractable but admit no polynomial kernelization unless NP $\subseteq$ co-NP/poly, and problems which are W[1]-hard.

To state and prove our results, we need some results of Nordh and Zanuttini [22]. Recall the definition of a frozen implementation (with equality). The *frozen partial co-clone* $\langle\Gamma\rangle_{fr}$ generated by Γ is the set of all relations that can be freezingly implemented by Γ. The *co-clone* of Γ is the set of all relations that can be implemented by Γ by a more general notion of implementation that also allows existentially quantified variables (see [22]); thus frozen partial co-clones refine co-clones. Nordh and Zanuttini [22] gave a full description of all frozen partial co-clones contained in the bijunctive co-clone. The free use of equality constraints in frozen implementations is somewhat more general than what we wish to allow, but this issue can be dealt with (cf. Lemma 2.2).

In particular, we need the following languages, which form the bases of some frozen partial bijunctive co-clones. The definitions and naming is from Nordh and Zanuttini [22].

1. $\Gamma_2^{p\neq} = \{(x \vee y), (x \neq y)\}$. This represents the largest bijunctive case for which Max Ones SAT(Γ) admits a polynomial kernel.

2. $\Gamma_3^n = \{R_3^n\}$, where $R_3^n = (\neg x \vee \neg y) \wedge (x \neq z)$. This language can be freezingly implemented by every bijunctive language not covered by the previous case. It is used in the kernel lower bound, in a PPT from the MCP problem.

3. $\Gamma_2^{p\neq i} = \{(x \vee y), (x \neq y), (x \rightarrow y)\}$. This represents the largest bijunctive case for which Max Ones SAT(Γ) is FPT.

Finally, we give a technical lemma to show that we have access to the constants.

Lemma 3.5. *Let Γ be a bijunctive constraint language which is neither zero-valid, one-valid, nor width-2 affine. Then the constants can be implemented.*

Proof. We first claim that a bijunctive language which is closed under negation is width-2 affine (hence this is not true for Γ). This follows from the lattice of co-clones, but can also be shown directly as follows. Let $R \in \Gamma$ be an arbitrary relation, and let F_R be an expression of R as a 2-CNF formula (note that clauses (x) or $(\neg x)$ would be inconsistent with negation). We claim that we get a valid expression of R by replacing every clause $C = (\ell_x \vee \ell_y)$ in F_R by an expression $(\ell_x \neq \ell_y)$, where $\ell_x \in \{x, \neg x\}$ and $\ell_y \in \{y, \neg y\}$. Indeed, if there were a tuple $\alpha \in R$ where ℓ_x and ℓ_y were both true, then its complement $(\bar{\alpha}) \in R$ would falsify C. The resulting formula is an expression of R using constraints $(x = y)$ and $(x \neq y)$, proving that R is width-2 affine. Since R was arbitrary, we find that Γ is width-2 affine, contradicting our assumptions.

Thus we can assume that there is a relation $R \in \Gamma$ that is not closed under negation; let $\alpha \in R$ such that the negation of α is not in R. Put a variable x in all positions of R that are true in α, and y in all other positions (note that α might have no true or no false positions, but this does not cause any problem in this proof). This constraint forbids $x = 0$ and $y = 1$, but allows $x = 1$ and $y = 0$. Let $R_0 \in \Gamma$ be a relation that is not zero-valid, and let β be an arbitrary tuple of R_0. Put variable x into every position where β is 1 and variable y into every position where β is 0. This forbids $x = y = 0$, but allows $x = 1$ and $y = 0$. Similarly, let R_1 be a relation that is not one-valid, and use a tuple $\gamma \in R_1$ to forbid $x = y = 1$. Therefore, these three constraints enforce $x = 1$ and $y = 0$, obtaining the constants. □

Let us now proceed with settling the remaining cases of Max Ones SAT(Γ).

Lemma 3.6. *Let Γ be a set of Boolean relations that is bijunctive but neither Horn, anti-Horn, nor width-2 affine (i.e., Γ generates the co-clone of bijunctive relations).*

1. *If $\Gamma \subseteq \langle \Gamma_2^{p\neq} \rangle_{fr}$, then* Max Ones SAT($\Gamma$) *has a polynomial kernelization (with $\mathcal{O}(k^2)$ variables). Otherwise,* Max Ones SAT(Γ) *admits no polynomial kernelization, unless* NP $\subseteq$ co-NP/poly.

2. *If $\Gamma \subseteq \langle \Gamma_2^{p\neq i} \rangle_{fr}$, then* Max Ones SAT($\Gamma$) *is* FPT *(with running time $2^k \cdot n^{\mathcal{O}(1)}$). Otherwise,* Max Ones SAT(Γ) *is* W[1]*-hard.*

Proof. Let (ϕ, k) be a Max Ones SAT(Γ) instance. If ϕ is infeasible (which can be tested since Γ is bijunctive), then (ϕ, k) is negative regardless of k; hence we assume throughout that ϕ is feasible. By assumption, Γ is not Horn, anti-Horn, or width-2 affine. Hence it is also not zero-valid or one-valid, since, for example, invariance under majority applied to $\alpha, \beta, 0 \in R$ would give $\alpha \wedge \beta \in R$, implying that R would be Horn. Thus we may apply Lemma 3.5. We split the proof into four parts.

Polynomial kernelization. If $\Gamma \subseteq \langle \Gamma_2^{p\neq} \rangle_{fr}$, then by the definition of a frozen co-clone, every relation in Γ, and thus all of ϕ, has a frozen implementation over $\Gamma_2^{p\neq} \cup \{=\}$, i.e., using positive clauses, equality, and disequality. We refer to this implementation when constructing a kernel, but the kernelization will apply to the original Γ as well. Let a maximal set of at least two variables which are connected by disequality or equality, with at least one disequality, be referred to as a *class* of variables. If there are at least k variable classes, then every solution will contain at least k true variables, and one of them can be found in polynomial time (since Γ is bijunctive). If any class contains at least $2k$ variables, then either the variables of this class have fixed values, in which case we make the corresponding assignments, or there exists a solution with at least k true variables. Finally, if any variable does not occur in a variable class, it can safely be set to 1. These observations leave a kernel with $\mathcal{O}(k)$ variable classes and $\mathcal{O}(k^2)$ variables in total. Finally, as the only changes we made to the formula were assignments, we can apply the kernelization using only relations in Γ by replacing all assigned variables by the constant variables c_1 or c_0.

Kernel lower bound. If $\Gamma \not\subseteq \langle \Gamma_2^{p\neq} \rangle_{fr}$, then, by [22], Γ freezingly implements R_3^n. By Lemma 2.2, we thus have Max Ones SAT(Γ_3^n) $\leq_{ppt}$ Max Ones SAT(Γ). We show that, in turn, the Multiple Compatible Patterns (MCP) problem reduces to Max Ones SAT(Γ_3^n) by a PPT.

Observe that R_3^n can be written as $(x \neq z) \wedge (y \rightarrow z)$, or with renamed variables, $(x \neq y) \wedge (z \rightarrow y)$. Let (I, k) be an instance of MCP, with string length r. Create variables x_j and y_j and a constraint $(x_j \neq y_j)$ for $1 \leq j \leq r$, coding the entries of the solution string; these variables contribute weight exactly r to any solution. The intuition is that $x_j = 1$ codifies that the solution string has value 0 at position j, and that $y_j = 1$ codifies value 1. Now for every pattern i, create a variable z_i; for every position j of pattern i containing 0, add a constraint $(x_j \neq y_j) \wedge (z_i \rightarrow x_j)$; and for every position j of pattern i containing 1, add a constraint $(x_j \neq y_j) \wedge (z_i \rightarrow y_j)$. Thus if z_i is set to true, then the positions of the solution string must match pattern i. Hence, any solution with $r + k$ true variables corresponds one-to-one to a string in $\{0,1\}^r$ and k patterns matching it. Thus, by Lemma 2.3, Max Ones SAT(Γ) admits no polynomial kernelization unless NP $\subseteq$ co-NP/poly.

Fixed-parameter tractability. If $\Gamma \subseteq \langle \Gamma_2^{p\neq i} \rangle_{fr}$, then, as before, there is an implementation of ϕ over $\Gamma_2^{p\neq i} \cup \{=\}$. Again consider the variable classes; if they number at least k, then find a solution in polynomial time. Otherwise, we check all $\mathcal{O}(2^k)$ assignments to variables of the variable classes. For each such assignment, propagate assignments to the remaining variables along all three constraint types. Any formula that remains after this is one-valid, since there are only implications and positive clauses.

W[1]-hardness. If $\Gamma \not\subseteq \langle \Gamma_2^{p\neq i} \rangle_{fr}$, then, by [22], there is an implementation of $(\neg x \vee \neg y)$ over $\Gamma \cup \{=\}$, hence by Lemma 2.2 there is a polynomial parameter transformation (and hence an fpt-reduction) from Independent Set to Max Ones SAT(Γ). □

3.5 Summary

Our results for Max Ones SAT(Γ) can be summarized as follows.

Corollary 3.7. *Let Γ be a finite set of Boolean relations. Then* Max Ones SAT(Γ) *falls into one of the following cases.*

1. *If Γ is one-valid, anti-Horn, or width-2 affine, then* Max Ones SAT(Γ) *is in* P.

2. *If* Γ *is affine, or if* $\Gamma \subseteq \langle (x \vee y), (x \neq y) \rangle_{fr}$, *then* Max Ones SAT($\Gamma$) *admits a polynomial kernelization.*

3. *If* $\Gamma \subseteq \langle (x \vee y), (x \neq y), (x \to y) \rangle_{fr}$ *but the previous cases do not apply, then* Max Ones SAT(Γ) *is in* FPT*, with a running time of* $2^k \cdot n^{\mathcal{O}(1)}$*, but there is no polynomial kernelization unless* NP $\subseteq$ co-NP/poly*.*

4. *If* Γ *is Horn or bijunctive, but no previous case applies, then* Max Ones SAT(Γ) *is* W[1]*-hard and in* XP*.*

5. *Otherwise* Max Ones SAT(Γ) *is* NP*-complete for* $k = 1$*.*

Remark 3.8. Containment of Max Ones SAT(Γ) in XP, if Γ is Horn or bijunctive (or anti-Horn or affine), can be easily seen. For example, given an instance (ϕ, k) over a bijunctive constraint language, we may guess k variables (i.e., try all n^k choices), set them to true, and check satisfiability of the remaining formula. However, this argument has a subtle, but crucial point: we need that setting a variable to true still leaves a formula that is bijunctive; the validity of this can be seen by adding the bijunctive constraint (x) to the formula for each selected variable x. The same holds also for Horn formulas, but the argument does not work for general zero-valid constraint languages.

For Max Ones SAT(Γ), with the exception of a few cases of bijunctive languages, the FPT cases generally coincide with the cases that admit polynomial kernelizations. This contrasts Min Ones SAT(Γ), for which there is a rich class of languages Γ such that Min Ones SAT(Γ) is FPT but admits no polynomial kernelization unless NP $\subseteq$ co-NP/poly (indeed, this is true for every non-mergeable Γ such that Min Ones SAT(Γ) is NP-hard).

4 Exact ones characterization

In this section we classify Exact Ones SAT(Γ) into admitting or not admitting a polynomial kernelization depending on the set of allowed relations Γ. Creignou et al. [8] showed that Exact Ones SAT(Γ) is in P when Γ is width-2 affine and NP-hard otherwise. Fixed-parameter tractability of Exact Ones SAT(Γ), i.e., solvability in time $f(k) \cdot n^{\mathcal{O}(1)}$, was characterized by Marx [20]; the result is a dichotomy into FPT and W[1]-complete cases based on two partial polymorphisms (see Theorem 4.3). Observe that Exact Ones SAT(Γ) is always in XP (unlike Max Ones SAT(Γ)), as it can be solved in time $n^{O(k)}$ by trying all assignments of weight exactly k.

Clearly, our characterization of cases with polynomial kernels will be a refinement of the fixed-parameter tractable cases. Accordingly, let us begin by recalling the notion of a *partial polymorphism* and the concrete invariant called weak separability. We also introduce a joined, stronger version of the two partial polymorphisms defining weak separability; this is used to characterize kernelizability of Exact Ones SAT(Γ).

Definition 4.1. A t-ary *partial polymorphism* is a partially defined function $f : \{0,1\}^t \to \{0,1\}$. For an r-ary relation R, we say that R is *invariant under* f if for any t tuples $\alpha_1, \ldots, \alpha_t \in R$, such that $f(\alpha_1(i), \ldots, \alpha_t(i))$ is defined for every $i \in [r]$, we have

$$(f(\alpha_1(1), \ldots, \alpha_t(1)), \ldots, f(\alpha_1(r), \ldots, \alpha_t(r))) \in R.$$

We state partial polymorphisms in a matrix form, where the columns represent the tuples for which f is defined, and the value below the horizontal line is the corresponding value of f.

If f is fully defined, it is referred to as simply a *polymorphism* (cf. the list of relation types in Section 2). Now we are able to formally state the three 3-ary partial polymorphisms relating to weak separability. Note that ordering of columns is immaterial (we chose lexicographical ordering) and swapping of rows would give equivalent partial polymorphisms.

Definition 4.2 ([20])**.** Let FPT(1), FPT(2), and FPT($1 \bowtie 2$) denote the following partial polymorphisms:

FPT(1)			
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1

FPT(2)			
0	0	0	1
0	1	0	1
0	1	1	1
0	0	1	1

FPT($1 \bowtie 2$)				
0	1	0	0	1
0	1	0	1	1
0	0	1	0	1
0	0	1	1	1

A boolean relation R is *weakly separable* if it is invariant under FPT(1) and FPT(2). It is *semi-separable* if it is invariant under FPT($1 \bowtie 2$). Note that invariance under FPT($1 \bowtie 2$) implies invariance under FPT(1) and FPT(2).

Fixed-parameter tractability of Exact Ones SAT(Γ) is classified as follows.

Theorem 4.3 ([20])**.** Exact Ones SAT(Γ) *is fixed-parameter tractable if every relation $R \in \Gamma$ is weakly separable. In the remaining cases it is* W[1]-complete*.*

Since any kernelization for a problem also implies fixed-parameter tractability, we only need to further classify the weakly separable cases.

For this, our main negative cases come from Min Ones SAT(Γ) and the Multiple Compatible Patterns (MCP) problem. For the former case, we first show (Lemma 4.4) that Min Ones SAT(Γ) reduces to Exact Ones SAT(Γ) via a polynomial parameter transformation (PPT), hence all negative cases for Min Ones SAT(Γ) transfer to Exact Ones SAT(Γ). For additional negative cases (i.e., cases where Min Ones SAT(Γ) has a polynomial kernel, but Exact Ones SAT(Γ) does not, under our usual complexity assumption), we give conditions for having a PPT from MCP to Exact Ones SAT(Γ). For technical reasons, the proof has several parts, depending on the expressiveness of the language Γ, but the final conclusion admits a concise statement; see Corollary 4.24.

We now show that Min Ones SAT(Γ) $\leq_{ppt}$ Exact Ones SAT(Γ).

Lemma 4.4. Min Ones SAT(Γ) *reduces to* Exact Ones SAT(Γ) *by a polynomial parameter transformation.*

Remark 4.5. Let us first note that there is a trivial *Turing* reduction that effectively makes a disjunction over $k+1$ instances, each asking for a solution with exactly k' true variables, for $0 \leq k' \leq k$. To get a *many-one* reduction needed to rule out (many-one) polynomial kernelizations one may simply add k free variables to ϕ. This ensures that any satisfying assignment with at most k true variables can be padded to exactly k true variables. We give a slightly more technical reduction that requires no free variables, which establishes that the reducibility is not an artifact of defining CSPs to allow free variables (e.g., one may instead require instances to be conjunctions of constraint applications).

of Lemma 4.4. Let (ϕ, k) be an instance of Min Ones SAT(Γ). If Γ is zero-valid, then Min Ones SAT(Γ) is in P and we perform the reduction by solving the instance in polynomial time and, depending on the outcome, outputting either a dummy yes-instance or a dummy no-instance of Exact Ones SAT(Γ). Otherwise, let R be a relation from Γ that is not zero-valid. We show that we are able to create a formula ϕ_0 on new variables whose minimum weight assignment has weight $k_{\min} \in \mathcal{O}(k)$, and which has satisfying assignments with i true variables for all i from $k_{\min}$ to $k_{\min}+k$. Then, $(\phi \wedge \phi_0, k+k_{\min})$ is in Exact Ones SAT(Γ) if and only if (ϕ, k) is in Min Ones SAT(Γ).

We assume first that R contains two tuples α and β such that $\alpha < \beta$, i.e., with bitwise $\leq$ and $\alpha \neq \beta$. Putting variables x, y, and z into positions of R where α and β are both zero, both one, or zero and one (as $\alpha < \beta$), respectively, creates a relation R' of arity up to three. In any case, positions with variable z must exist. Let us first assume that R' is ternary. Thus we know that $(0,1,0), (0,1,1) \in R'$ and we let ϕ_0 consist of k variable-disjoint copies of R'-constraints. The crucial observation is that each R'-constraint has a satisfying assignment with one or two true variables, but not with zero true variables since it is not zero-valid. Thus the formula ϕ_0 has satisfying assignments with i true variables for all values $i = k, \ldots, 2k$, since the R' constraints are variable-disjoint. The cases where R' has arity less than three are along the same lines, since z is always present.

Now, otherwise, R contains no such tuples, and we let α and β be any two distinct tuples from R. Putting variables w, x, y, and z into positions of R as follows.

	w	x	y	z	
α	0	0	1	1	$\in R$
β	0	1	0	1	$\in R$

We implement a relation R''. Note that, by assumption $\alpha \not\leq \beta$ and $\beta \not\leq \alpha$, thus positions x and y must exist. We discuss the case that all four position types exist and R'' is 4-ary; the other cases are similar since x and y always exist. We implement a new relation R''' by $R'''(w,x,x',y,z) = R''(w,x,y,z) \wedge R''(w,x',y,z)$. Thus we have $(0,0,0,1,1),(0,1,1,0,1) \in R'''$. We let ϕ_0 consist of k variable-disjoint copies of R'''-constraints. Again ϕ_0 has satisfying assignments with i true variables for all values $i = k_{\min}, \ldots, k_{\min}+k$ where $k_{\min} = k$ or $k_{\min} = 2k$ depending on whether R''' has a satisfying assignment with one true variable (at least one is needed because R is not zero-valid). If yes then each of the k copies can be satisfied with one or two true variables so we get satisfying assignments with k to $2k$ true variables; we then set $k_{\min} = k$. Else, we know that assignments with two and three true variables are possible, so over all k copies we can get $2k$ to $3k$ true variables and set $k_{\min} = 2k$. □

Using the kernelization dichotomy for Min Ones SAT(Γ) [18], we may exclude further choices of Γ, i.e., show that Exact Ones SAT(Γ) does not admit a polynomial kernelization. We recall that the kernelizability of Min Ones SAT(Γ) is governed by mergeability, and that a relation is mergeable if it is invariant under the following partial polymorphism:

Mergeable						
0	1	0	1	1	0	1
0	1	0	0	0	0	1
0	0	1	1	0	1	1
0	0	1	0	0	0	1
0	1	0	1	0	0	1

Theorem 4.6 ([18])**.** *Let Γ be a finite Boolean constraint language. Then* Min Ones SAT(Γ) *falls into one of the following cases.*

1. *If Γ is 0-valid, Horn, or width-2 affine, then* Min Ones SAT(Γ) *is in* P *[16].*
2. *If Γ is mergeable, then* Min Ones SAT(Γ) *admits a polynomial kernelization.*
3. *Otherwise* Min Ones SAT(Γ) *is* FPT*, but does not admit a polynomial kernelization unless* NP $\subseteq$ co-NP/poly*.*

Using Lemma 4.4 we immediately obtain the following corollary.

Corollary 4.7. *Let Γ be any finite Boolean constraint language. If Γ is not mergeable and* Min Ones SAT(Γ) *is* NP*-hard then* Exact Ones SAT(Γ) *does not admit a polynomial kernelization unless* NP $\subseteq$ co-NP/poly*.*

By Khanna et al. [16], Min Ones SAT(Γ) is in P when Γ is zero-valid, Horn, or width-2 affine; in all other cases it is NP-hard (APX-hard). The kernelizability of Exact Ones SAT(Γ) for the former choices of Γ will be considered in the following section. In the remaining cases Min Ones SAT(Γ) is NP-hard and thus, due to Corollary 4.7 it remains to consider the case that Γ is mergeable; this is done in Section 4.2. Both sections combined constitute a proof for the following theorem, characterizing Exact Ones SAT(Γ).

Theorem 4.8. *Let Γ be a finite constraint language that is weakly separable.*

1. *If Γ is width-2 affine, then* Exact Ones SAT(Γ) *is in* P*.*
2. *If Γ is anti-Horn, or both mergeable and semi-separable, then* Exact Ones SAT(Γ) *admits a polynomial kernelization.*
3. *In all other cases* Exact Ones SAT(Γ) *does not admit a polynomial kernelization unless* NP $\subseteq$ co-NP/poly*.*

Any weakly separable Γ that is mergeable and either (1) Horn or (2) zero-valid is also width-2 affine.

4.1 Characterization I: Zero-valid, Horn, and width-2 affine

We begin by considering constraint languages Γ such that Min Ones SAT(Γ) is in P, i.e., zero-valid, Horn, and width-2 affine constraint languages.

4.1.1 Width-2 affine cases

If all relations in Γ are width-2 affine then, by Creignou et al. [8], Exact Ones SAT(Γ) is in P. To see this, recall that width-2 affine constraints can be implemented by assignments, equality, and disequality. It can be easily checked whether a given formula is satisfiable. If it is, then the equalities and disequalities partition the variables into equivalence classes. If disequalities are present, then certain pairs of classes must take different values (i.e., variables of one class must be assigned true and those of the other class must be assigned false, or vice versa). One may observe that the remaining problem of reaching exactly k true variables is simply a subset sum question with all numbers being less than the input size: For a pair of classes C_1 and C_2 whose variables must take opposite values we get either $\min(|C_1|, |C_2|)$ true variables or $\max(|C_1|, |C_2|)$ true variables. This can be taken into account by decreasing k by $\min(|C_1|, |C_2|)$ and leaving the option of getting $\max(|C_1|, |C_2|) - \min(|C_1|, |C_2|)$ further true variables. For a single class C_1 that is not in a pair (because there are no disequalities involving one of its variables) we directly have the choice between 0 and $|C_1|$ true variables. Because all numbers are less than the input size this can be solved in polynomial time by dynamic programming.

4.1.2 Horn cases

We show that every weakly separable relation that is also Horn can be implemented by $\{(\neg x), (x), (x = y)\}$. Thus if Γ is Horn and weakly separable, then it is also width-2 affine and Exact Ones SAT(Γ) is polynomial.

Lemma 4.9. *Let R be a weakly separable relation. If R is Horn, then R is implementable by $\{(\neg x), (x), (x = y)\}$.*

Proof. We show that R is closed under disjunction $(a \vee b)$ and under the polymorphism $f(a, b, c) = a \wedge (b \oplus c \oplus 1)$. Let $\alpha, \beta, \gamma \in R$:

α	0	0	0	0	1	1	1	1		
β	0	0	1	1	0	0	1	1		
γ	0	1	0	1	0	1	0	1		
$\beta \wedge \gamma$	0	0	0	1	0	0	0	1	closed under conjunction	(i)
$\beta \vee \gamma$	0	1	1	1	0	1	1	1	FPT(1) on (i), β, γ	
$\alpha \wedge (\beta \vee \gamma)$	0	0	0	0	0	1	1	1	closed under conjunction	(ii)
$\alpha \wedge (\beta \wedge \gamma)$	0	0	0	0	0	0	0	1	closed under conjunction	(iii)
$\alpha \wedge (\beta \oplus \gamma \oplus 1)$	0	0	0	0	1	0	0	1	FPT(2) on (iii),(ii), α	

Thus, by [7], R can be implemented by $\{(x), (\neg x), (x = y)\}$. (Note that the first two lines prove invariance under $(a \vee b)$.) □

We immediately get the following conclusion:

Lemma 4.10. *Let Γ be a finite constraint language that is weakly separable. If all relations in Γ are Horn, then* Exact Ones SAT(Γ) *is in* P.

4.1.3 Zero-valid cases

For zero-valid (and weakly separable) constraint languages Γ, we find that Exact Ones SAT(Γ) is either polynomial-time solvable, when Γ is width-2 affine, or that it does not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly. We will see that this coincides with whether or not Γ is mergeable. We begin by showing that for zero-valid Γ mergeability implies that Γ is width-2 affine (and Exact Ones SAT(Γ) is in P).

Lemma 4.11. *Let R be a mergeable and weakly separable relation. If R is also zero-valid, then it can be implemented using only equality, $(x = y)$, and negative assignments, $(\neg x)$.*

Proof. Let R be a mergeable and zero-valid relation that is invariant under FPT(1) and FPT(2). We show that R is invariant under conjunction $(a \wedge b)$ and exclusive disjunction $(a \oplus b)$. Let $\alpha, \beta \in R$:

0	0	0	0	0		
α	0	0	1	1		
β	0	1	0	1		
$\alpha \wedge \beta$	0	0	0	1	mergeability on α, 0, β, 0	
$\alpha \wedge \neg\beta$	0	0	1	0	FPT(2) on 0, $\alpha \wedge \beta$, α	(i)
$\neg\alpha \wedge \beta$	0	1	0	0	FPT(2) on 0, $\alpha \wedge \beta$, β	(ii)
$\alpha \oplus \beta$	0	1	1	0	FPT(1) on 0, (i), (ii)	

Thus R can be implemented by $(\neg x)$ and $(x = y)$ according to [7]. □

We will now prove that we may assume to have positive and negative assignments available in Γ. To this end, we give a polynomial parameter transformation from the case with assignments to the case without assignments.

Let us briefly recall the effect of a polynomial parameter transformation in this context: A lower bound for Exact Ones SAT($\Gamma \cup \{(x), (\neg x)\}$) transfers to Exact Ones SAT(Γ). A polynomial kernelization for Exact Ones SAT($\Gamma \cup \{(x), (\neg x)\}$) applies to instances of Exact Ones SAT(Γ): We apply the kernelization (which may introduce assignments) and use the polynomial parameter transformation to replace any assignments introduced in the kernelization to make the result an instance of Exact Ones SAT(Γ).

We first show how to implement assignments when we have equality available.

Lemma 4.12. *If Γ implements equality, then* Exact Ones SAT($\Gamma \cup \{(x), (\neg x)\}$) *reduces to* Exact Ones SAT(Γ) *by a polynomial parameter transformation.*

Proof. Let (ϕ, k) be an instance of Exact Ones SAT($\Gamma \cup \{(x), (\neg x)\}$); by Lemma 2.2, we may assume that $=\in \Gamma$. First, we make a copy x' of every variable x of ϕ and add the constraint $(x = x')$. Additionally we add a single new variable c_1 and let the new parameter be $k' = 2k + 1$. Thus the only way to have a satisfying assignment with exactly k' true variables is to set c_1 to true. This permits us to replace all (x) constraints by $(x = c_1)$. Second, we add k' variables $y_1, \ldots, y_{k'}$ as well as a variable c_0 and implement constraints $(c_0 = y_i)$ for all $i \in \{1, \ldots, k'\}$. Thus the variables y_i as well as c_0 take the same value in any satisfying assignment. Since assigning true to all these variables would exceed the number of k' true variables, they will be set to false. Hence, we may replace all $(\neg x)$ constraints by $(x = c_0)$.

The formula ϕ' obtained in this way has a satisfying assignment with exactly k' true variables if and only if ϕ has a satisfying assignment with k true variables. □

Now, we can show that for constraint languages that are zero-valid and weakly separable but not Horn, we may assume (up to equivalence under polynomial parameter transformations) that they contain assignments. That is, the case with assignment constraints (x) and $(\neg x)$ has a polynomial parameter transformation to the case without assignments.

Lemma 4.13. *Let Γ be a constraint language that is zero-valid but not Horn. Then* Exact Ones SAT($\Gamma \cup \{(x), (\neg x)\}$) *reduces to* Exact Ones SAT(Γ) *by a polynomial parameter transformation.*

Proof. Given any instance (ϕ, k) of Exact Ones SAT($\Gamma \cup \{(x), (\neg x)\}$), we will show how to express (x) and $(\neg x)$ to obtain an equivalent instance (ϕ', k') of Exact Ones SAT(Γ). We consider two cases depending on whether Γ contains one-valid relations (all relations in Γ are zero-valid):

I) Γ contains a relation R that is zero-valid and one-valid. Let α be a tuple that is not contained in R. We implement a binary constraint $R'(x, y)$ by putting x into all positions R where α is one and y into all positions where α is zero. Thus $(0, 0), (1, 1) \in R'$ and $(1, 0) \notin R'$. Hence $R'(x, y) \wedge R'(y, x)$ implements equality. We proceed as in Lemma 4.12 to obtain an equivalent instance (ϕ', k') of Exact Ones SAT(Γ).

II) No relation in Γ is one-valid. In this case, let us begin by observing that we can implement $(\neg x)$ by $R(x, \ldots, x)$ using any $R \in \Gamma$. We add a new variable c_0 and force it to be zero by $(\neg c_0)$.

Now, since Γ is not Horn, it must contain a relation R that is not invariant under conjunction. Let $\alpha, \beta \in R$ be tuples witnessing this fact, i.e., tuples α and β such that $\gamma = \alpha \wedge \beta \notin R$. Observe that $\gamma \neq 0 \in R$ and that $\gamma < \alpha \in R$. We implement a binary constraint $R'(x, y)$ by putting c_0 in all positions of R where α and γ are zero, putting x into all positions where α and γ are one, and y into all positions where α is one and γ is zero. Note that the latter two types of positions must exist to

distinguish γ from 0 and α, respectively. Thus $(0,0),(1,1) \in R'$ and $(1,0) \notin R'$. Again, $R'(x,y) \wedge R'(y,x)$ implements equality and we can proceed as in Lemma 4.12. □

Now we show that Exact Ones SAT(Γ) does not admit a polynomial kernelization if Γ is zero-valid and weakly separable but not width-2 affine (and hence also not Horn).

Lemma 4.14. *Let Γ be a zero-valid and weakly separable constraint language that is not width-2 affine. Then* Exact Ones SAT(Γ) *does not admit a polynomial kernelization unless* NP $\subseteq$ co-NP/poly.

Proof. Let Γ be a zero-valid constraint language that is not width-2 affine. We first observe that Γ is not Horn, since it would then be width-2 affine by Lemma 4.9. Then we see that Γ is not mergeable, since mergeability applied to α, 0, β, $0 \in R$ (where $R \in \Gamma$) implies $\alpha \wedge \beta \in R$; thus all relations in Γ would be Horn.

Let $\Gamma' = \Gamma \cup \{(x), (\neg x)\}$. By Lemma 4.13, Exact Ones SAT(Γ') reduces to Exact Ones SAT(Γ) by a polynomial parameter transformation. Therefore, it suffices to show that Exact Ones SAT(Γ') does not admit a polynomial kernelization. For this observe that Γ' is neither width-2 affine, Horn, zero-valid, nor mergeable (recall that closure properties must hold for every relation in $\Gamma' \supseteq \Gamma$). Thus, by [16], Min Ones SAT(Γ') is NP-hard, and by Corollary 4.7, Exact Ones SAT(Γ') does not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly. By the polynomial parameter transformation, the same holds for Exact Ones SAT(Γ). □

In conclusion we get the following picture for constraint languages which are zero-valid, Horn, or width-2 affine (i.e., constraint languages Γ such that Min Ones SAT(Γ) is in P).

Corollary 4.15. *Let Γ be a weakly separable constraint language that is zero-valid, Horn, or width-2 affine. Then* Exact Ones SAT(Γ) *is in* P *if Γ is width-2 affine, otherwise it does not admit a polynomial kernelization unless* NP $\subseteq$ co-NP/poly. *Equivalently* Exact Ones SAT(Γ) *is in* P *if Γ is mergeable, and otherwise it does not admit a polynomial kernelization unless* NP $\subseteq$ co-NP/poly.

4.2 Characterization II: Remaining cases

Having handled the cases for which Min Ones SAT(Γ) is in P, Corollary 4.7 allows us to restrict our attention to sets of relations Γ that are mergeable as well as weakly separable. We begin with a special case (for which we do not require mergeability).

Lemma 4.16. *Let R be a weakly separable relation. If R is anti-Horn, then R can be implemented by equality, positive clauses, and assignment; hence R is semi-separable and mergeable.*

Proof. We show that R is invariant under $a \vee (b \wedge \bar{c})$. Let $\alpha, \beta, \gamma \in R$:

α	0	0	0	0	1	1	1	1	
β	0	0	1	1	0	0	1	1	
γ	0	1	0	1	0	1	0	1	
$\alpha \vee \gamma$	0	1	0	1	1	1	1	1	closed under disjunction (i)
$\alpha \vee \beta \vee \gamma$	0	1	1	1	1	1	1	1	closed under disjunction (ii)
$\alpha \vee (\beta \wedge \bar{\gamma})$	0	0	1	0	1	1	1	1	FPT(2) on α, (i), (ii)

Thus R is invariant under $a \vee (b \wedge \bar{c})$ implying by [7] that it can be implemented using equality, negative assignments, and positive clauses all of which are semi-separable and mergeable. It follows directly that R is semi-separable and mergeable too. □

Lemma 4.17. *Let Γ be a constraint language that is anti-Horn and weakly separable. Then there is either a polynomial parameter transformation from* Exact Ones SAT(Γ) *to* d-Hitting Set, *for some constant d, or one from* Exact Ones SAT($\Gamma \cup \{(x), (\neg x)\}$) *to* Exact Ones SAT($\Gamma$).

Proof. By Lemma 4.16, Γ is mergeable; thus by Lemma 4.11 we may assume that Γ contains at least one relation that is not zero-valid (or else Exact Ones SAT(Γ) is in P and we produce a trivial reduction). Pick some $R \in \Gamma$; we will attempt to implement either (x) and $(\neg x)$ directly, or $(x = y)$ and invoke Lemma 4.12. Since the construction is similar to our previous constructions, we cover them only briefly.

If R is zero-valid and one-valid, then we implement $(x = y)$ by grouping x and y according to some tuple $\alpha \notin R$ (we cannot obtain an implication since R is invariant under FPT(2)).

If R is neither zero-valid nor one-valid, then grouping x and y according to some $\alpha \in R$ produces $(x = 0) \wedge (y = 1)$ (note that we cannot obtain $(x \neq y)$, as it is not anti-Horn).

If R is zero-valid and not one-valid, then we get the constraint $(\neg x)$, and using any non-zero-valid $R' \in \Gamma$ we get (x).

Finally, we get to assume that every relation $R \in \Gamma$ is one-valid but not zero-valid; we immediately get the constraint $(x) = R(x, \ldots, x)$. If there are any tuples $\alpha < \beta$ such that $\alpha \in R$, $\beta \notin R$ for any $R \in \Gamma$, then using (x) we may proceed as in Lemma 4.13 by placing c_1 in the positions true in α, producing $(x = y)$. Otherwise, we find that every $R \in \Gamma$ is upwards closed, i.e., for each $\alpha \in R$ and any $\beta \geq \alpha$ we also have $\beta \in R$. In this case, every relation has an implementation using positive clauses only: To implement such a relation R, for each maximal (w.r.t. bitwise $\leq$) tuple not in R make a positive clause excluding the tuple. If k is larger than the number of variables, then the instance is negative and we reduce to a dummy no-instance. Otherwise, since positive clauses are upwards closed, the instance is a yes-instance if and only if there is a satisfying assignment with at most k true variables. Thus we may simply replace our instance by an equivalent instance of d-Hitting Set (where d is the maximum arity of any $R \in \Gamma$). □

Thus when Γ is anti-Horn, we either have a reduction from Exact Ones SAT(Γ) to d-Hitting Set, or we may conclude that we have constants available and that Γ is mergeable as well as semi-separable. The latter case will be treated later by Theorem 4.23. In the former case, we get a polynomial kernelization for Exact Ones SAT(Γ).

Now, we consider the case that Γ is mergeable and contains at least one relation which is not anti-Horn. We first show that we can implement disequality and assignments, then proceed with the kernelization dichotomy.

Lemma 4.18. *Let Γ be a weakly separable constraint language that is mergeable but not anti-Horn. Then* Exact Ones SAT($\Gamma \cup \{(x \neq y)\}$) *reduces to* Exact Ones SAT(Γ) *by a polynomial parameter transformation.*

Proof. There must be tuples $\alpha, \beta \in R$ for some $R \in \Gamma$ such that $\alpha \vee \beta \notin R$, i.e., witness tuples for the fact that R is not closed under disjunction (equivalently that R is not anti-Horn). For convenience we collect the positions of R according to the values of α and β into four types A through D. Note that positions of type B and C must exist to distinguish $\alpha \vee \beta$ from α and β (see table below). We first observe that $\alpha \wedge \beta \notin R$ as $\alpha \wedge \beta, \alpha, \beta \in R$ and $\alpha \vee \beta \notin R$ would form a witness against invariance under FPT(1). It follows that the zero-tuple cannot be contained in R since $\alpha, 0, \beta, 0 \in R$ and $\alpha \wedge \beta \notin R$ would form a witness against mergeability. We summarize our conclusions.

	A	B	C	D	
α	0	0	1	1	$\in R$
β	0	1	0	1	$\in R$
$\alpha \vee \beta$	0	1	1	1	$\notin R$
$\alpha \wedge \beta$	0	0	0	1	$\notin R$
0	0	0	0	0	$\notin R$

Let us now consider the case that R *is not one-valid*. In that case we can define a binary relation R' by plugging the first variable into positions A and B and the second variable into positions C and D (at least B and C exist). Clearly, this gives $(0, 1) \in R'$ since $\alpha \in R$ and $(0, 0), (1, 1) \notin R'$ since the zero and the one-tuple are not in R. Now if $(1, 0) \in R'$ then $R' = \{(0, 1), (1, 0)\}$, i.e., we have an implementation of disequality.

Otherwise $R' = \{(0, 1)\}$ and we can add two new variables c_0 and c_1 to any instance (ϕ, k) of Exact Ones SAT(Γ) as well as a constraint $R'(c_0, c_1)$ obtaining ϕ'. Clearly, (ϕ, k) and $(\phi', k+1)$ are equivalent and by c_0 and c_1 we have constants zero and one available. Using the constants allows us to implement disequality by putting c_0 into positions A, c_1 into positions D, the first variable into positions B, and the second variable into positions C.

Let us now address the case that R *is one-valid*; so far we know the following:

	A	B	C	D	
α	0	0	1	1	$\in R$
β	0	1	0	1	$\in R$
1	1	1	1	1	$\in R$
0	0	0	0	0	$\notin R$
$\alpha \wedge \beta$	0	0	0	1	$\notin R$
$\alpha \vee \beta$	0	1	1	1	$\notin R$

In this case we also know that positions of type A must exist, to distinguish the included one-tuple from the excluded $\alpha \vee \beta$. We may also assume that we have the constant one c_1 available since $R(c_1, \ldots, c_1)$ forces this.

Now, if there are no positions of type D, then we put x, y, and z into positions A, B, and C respectively, obtaining a ternary relation R'' such that:

	x	y	z	
α'	0	0	1	$\in R''$
β'	0	1	0	$\in R''$
1	1	1	1	$\in R''$
$\alpha' \wedge \beta'$	0	0	0	$\notin R''$
$\alpha' \vee \beta'$	0	1	1	$\notin R''$

Otherwise, i.e., if positions of type D exist then we can also obtain an R'' with these properties by additionally putting c_1 in positions D. We may conclude that $(1,0,1) \notin R''$, since otherwise $(1,0,1), (1,1,1), (0,0,1) \in R''$ but $(0,1,1) \notin R''$ would violate invariance under FPT(2). Now putting c_1 in place of z gives a binary relation R''' containing $(0,0)$ and $(1,1)$ since $(0,0,1)$ and $(1,1,1) \in R''$ and not containing $(0,1)$ as well as $(1,0)$ since $(0,1,1), (1,0,1) \notin R''$. Thus R''' is equality. Therefore we can implement the constant zero c_0 by adding k new variables $s_1, \ldots, s_k$ as well as equality constraints $R'''(c_0, s_i)$ for all i. Now we can implement disequality on two variables x and y as $R''(c_0, x, y)$. □

We continue by proving that we may assume to have assignments in Γ, if Γ implements (or contains) disequality.

Lemma 4.19. *If Γ implements disequality, then* Exact Ones SAT($\Gamma \cup \{(x), (\neg x)\}$) *reduces to* Exact Ones SAT(Γ) *by a polynomial parameter transformation.*

Proof. Let (ϕ, k) be an instance of Exact Ones SAT($\Gamma \cup \{(x), (\neg x)\}$). We show how to obtain an equivalent instance (ϕ', k') of Exact Ones SAT(Γ) by implementing (x) and $(\neg x)$: We start with $\phi' = \phi$ and let the new parameter be $k' = k+1$. We add a new variable c_1 as well as $k'+1$ variables $y_1, \ldots, y_{k'+1}$ and implementations of $(c_1 \neq y_i)$ for all $i \in \{1, \ldots, k'+1\}$. Furthermore, we add c_0 and an implementation of $(c_1 \neq c_0)$. Observe that every feasible assignment for ϕ' of weight k' assigns one to c_1 and zero to all y_i; otherwise assigning one to all y_i would give a total number of true variables greater than k'. Thus by $(c_1 \neq c_0)$ it must assign zero to c_0. Therefore we can implement (x) by $(x \neq c_0)$ and $(\neg x)$ by $(x \neq c_1)$.

It is easy to see that any satisfying assignment with k' true variables can be restricted to a feasible assignment for ϕ with k true variables. The converse is straightforward too. □

For the remaining cases we can now show that Exact Ones SAT(Γ) does not admit a polynomial kernelization if Γ is not semi-separable (i.e., not invariant under FPT($1 \bowtie 2$)).

Theorem 4.20. *Let Γ be a weakly separable constraint language. If Γ implements disequality and contains a relation that is not invariant under* FPT($1 \bowtie 2$)*, then* Exact Ones SAT(Γ) *does not admit a polynomial kernelization unless* NP $\subseteq$ co-NP/poly.

Proof. We give a polynomial parameter transformation from the Multiple Compatible Patterns (MCP) problem, already used in Lemma 3.6. The definition of MCP can be found in Section 2 and Lemma 2.3 shows that MCP does not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly. By Lemmas 4.19 and 2.2, we may assume that Γ contains disequality and the assignments (x) and $(\neg x)$.

Let R be any relation from Γ that is not invariant under FPT($1 \bowtie 2$). Let $\alpha, \beta, \gamma \in R$ and $\delta \notin R$ be a witness for this fact. Again we collect the positions of the tuples into groups, A through E:

	A	B	C	D	E	
α	0	1	0	0	1	$\in R$
β	0	1	0	1	1	$\in R$
γ	0	0	1	0	1	$\in R$
δ	0	0	1	1	1	$\notin R$

If no position in these tuples matches the D column, then $\delta = \gamma$. If no position of type B or C is present, then these tuples would also witness that R is not invariant under FPT(1) or FPT(2), respectively, contradicting our assumption on Γ.

We use R to implement $(x \to y) \wedge (y \neq z)$. Put constant zero into positions A and constant one into all positions E. Put x, y, and z into positions D, B, and C, respectively, and add a constraint $(y \neq z)$.

Now we can use the same reduction from MCP as in Lemma 3.6, since the reduction maps yes-instances to formulas with satisfying assignments having exactly $k+r$ ones and no-instances to formulas where all satisfying assignments have less than $k+r$ ones. □

It remains to consider the case that all relations in Γ are mergeable and semi-separable, i.e., invariant under FPT($1 \bowtie 2$). Note that this partial polymorphism contains all columns of the two partial polymorphisms FPT(1) and FPT(2) (if FPT(2) is rearranged so that the first tuple is last), thus FPT($1 \bowtie 2$) is stronger (i.e., it implies the other two). Our main tool for the kernelization, as in [18], is the notion of a *sunflower*. Ordinarily stated for set systems, we give here a version for sets of tuples.

Definition 4.21 ([18])**.** Let $\mathcal{U}$ be a finite set, let $d \in \mathbb{N}$, and let $\mathcal{H} \subseteq \mathcal{U}^d$. A *sunflower (of tuples) with cardinality t and core $C \subseteq \{1, \ldots, d\}$* in $\mathcal{U}$ is a subset consisting of t tuples that have the same element at all positions in C and, in the remaining positions, no element occurs in more than one tuple. The set of remaining positions $P = \{1, \ldots, d\} \setminus C$ is called the *petals*.

With an adaptation of Erdős and Rado's Sunflower Lemma [12], we are able to find a sunflower in any set of bounded-width tuples of sufficient cardinality.

Lemma 4.22 ([18])**.** *Let $\mathcal{U}$ be a finite set, let $d \in \mathbb{N}$, and let $\mathcal{H} \subseteq \mathcal{U}^d$. If the size of $\mathcal{H}$ is greater than $k^d(d!)^2$, then it contains a sunflower of cardinality $k+1$.*

Theorem 4.23. *Let Γ be a semi-separable and mergeable constraint language. Then* Exact Ones SAT(Γ) *admits a polynomial kernelization.*

Proof. If Γ is anti-Horn, then according to Lemma 4.17 we either have that Exact Ones SAT($\Gamma \cup \{(x), (\neg x)\}$) reduces to Exact Ones SAT(Γ), or Exact Ones SAT(Γ) reduces to the d-Hitting Set problem. In the latter case, we have a polynomial kernelization using the fact that d-Hitting Set admits a polynomial kernelization [1].

If Γ is not anti-Horn, then by Lemmas 4.18 and 4.19 we again have a polynomial parameter transformation from Exact Ones SAT($\Gamma \cup \{(x), (\neg x)\}$) to Exact Ones SAT($\Gamma$).

Thus it suffices to show that Exact Ones SAT($\Gamma \cup \{(x), (\neg x)\}$) admits a polynomial kernelization; we remark that Exact Ones SAT(Γ) trivially reduces to Exact Ones SAT(Γ') for any $\Gamma' \supseteq \Gamma$. For ease of notation, we assume w.l.o.g. that $(x), (\neg x) \in \Gamma$.

Let (ϕ, k) be an instance of Exact Ones SAT(Γ). We begin by adding two new variables c_0 and c_1 as well as constraints $(\neg c_0)$ and (c_1), and by increasing the parameter value by one. Clearly, the obtained instance is equivalent to (ϕ, k) and c_0 and c_1 must take values false and true, respectively, in any satisfying assignment. Slightly abusing notation we call this new instance (ϕ, k).

Non zero-valid constraints. First we reduce the number of constraints that are not zero-valid. We exhaustively apply the following steps for each non zero-valid relation R in Γ (let d denote its arity):

- If the number of R-constraints in ϕ is greater than $(d!)^2 \cdot k^d$, then we can find a sunflower consisting of $k+1$ R-constraints by applying Lemma 4.22 to the set of all tuples of variables that occur in R-constraints. If the core of the sunflower is empty, then no assignment of weight exactly k can be feasible, since the petal positions do not share variables. In this case we reject the instance.

- Otherwise we observe that a sunflower can be simplified: Fix any constraint of the sunflower and consider any two assignments α and β to the core variables, such their extensions by setting all the petals to 0 (that is, the assignments $(\alpha, 0)$ and $(\beta, 0)$) are feasible. Note that the assignment to the

core variables must always allow for all petal variables to be set to zero, else such an assignment cannot be extended to a satisfying assignment with k true variables (each of the $k+1$ disjoint petals would require at least one true variable). Furthermore, let γ be any assignment to the petal variables, such that (α, γ) is feasible. By an application of semi-separability it follows that (β, γ) is also feasible:

$(\alpha, 0)$	0	0	1	1	0	0	
(α, γ)	0	0	1	1	1	0	
$(\beta, 0)$	0	1	0	1	0	0	
(β, γ)	0	1	0	1	1	0	$\mathrm{FPT}(1 \bowtie 2)$

Thus assignments to the petals are independent of the assignments to the core variables.

- We have observed that it is sufficient and necessary for the core variables to take any values such that the petal variables may take value zero. Therefore, we add one new R-constraint, with only the core variables in their positions and the constant-zero variable c_0 in the petal positions.
- For each constraint of the sunflower, we replace the core variables by the constant variables c_0 and c_1 matching some feasible assignment to the core, say α. The assignment to the petals will then be consistent with any core assignment. Thus we make the following replacements (w.l.o.g. the core consists of the first c positions):

$$\begin{aligned} R(x_1, \ldots, x_c, y_{1,1}, \ldots, y_{1,p}) &\Rightarrow R(\alpha, y_{1,1}, \ldots, y_{1,p}) \\ R(x_1, \ldots, x_c, y_{2,1}, \ldots, y_{2,p}) &\Rightarrow R(\alpha, y_{2,1}, \ldots, y_{2,p}) \\ \vdots & \quad \vdots \\ R(x_1, \ldots, x_c, y_{k+1,1}, \ldots, y_{k+1,p}) &\Rightarrow R(\alpha, y_{k+1,1}, \ldots, y_{k+1,p}) \end{aligned}$$

  Let us recall, however, that each constraint $R(\alpha, y_{i,1}, \ldots, y_{i,p})$ induces a zero-valid constraint on $y_{i,1}, \ldots, y_{i,p}$. Therefore, we may implement these induced constraints using only equality and negative assignments, according to Lemma 4.11. Thus in each replacement step we replace $k+1$ R-constraints by 1 R-constraint for the core variables plus a number of equality and negative assignment constraints on the petal variables.
- Each replacement decreases the number of R-constraints. Thus the number of repetitions is bounded by the initial size of ϕ.

Zero-valid constraints. If R is zero-valid, then the sunflower lemma is not helpful, as it may simply return a packing of $k+1$ pairwise disjoint constraints (leading to no further conclusion). Instead, zero-valid constraints are handled as follows. First, we replace each zero-valid constraint by an implementation using negative assignments and equality constraints, according to Lemma 4.11. Then we replace all occurrences of variables x that have a negative assignment $(\neg x)$ by c_0.

Next we consider the equivalence classes given by the equality constraints. Observe that these classes implicitly assign a weight equal to the size of the class. If any class contains more than k variables, then all those variables must take value zero in any satisfying assignment of weight k. Accordingly, we replace the variables by c_0. For the remaining equivalence classes we choose one representative variable per class, say variable x for some class C. All occurrences of variables from $C \setminus \{x\}$ (except for those in the equality relations) are then replaced by x. Essentially we have simplified the class to one variable of weight $|C|$. Note that equality constraints are the only remaining zero-valid constraints.

Bounding the kernel size. After these replacements there are at most $(d!)^2 \cdot k^d$ copies of any non-zero-valid R-constraint of arity d in Γ. Thus the total number of variables in such constraints is bounded by $\mathcal{O}(d \cdot (d!)^2 \cdot k^d)$, since Γ is finite and independent of the input. All other variables, i.e., those that only occur in equality constraints, must be in some equivalence class of size at most k. The total number of such variables for which the representative is contained in a non-zero-valid constraint is clearly bounded by k times the number of representatives, i.e., bounded by $\mathcal{O}(d \cdot (d!)^2 \cdot k^{d+1})$.

It remains to bound and reduce the number of variables in classes where the representative occurs only in equality constraints. (This includes free variables, i.e., variables that appear in no constraints, as a special case with equivalence class of size one.) Note that, in the min ones setting, we would simply

set those variables to zero. Here they may be needed to reach the exact total weight of k. However, it clearly suffices to keep $\lfloor k/t \rfloor$ classes of size t for this purpose; additional copies have no impact. The total number of variables in these classes is at most

$$\sum_{t=1}^{k} \lfloor k/t \rfloor \cdot t \leq k^2.$$

Thus the reduced instance has $\mathcal{O}(d \cdot (d!)^2 \cdot k^{d+1})$ variables; where d is the maximum arity of relations in Γ, i.e., a constant independent of (ϕ, k). □

4.3 Summary

We sum up the properties of Exact Ones SAT(Γ) in the following corollary.

Corollary 4.24. *Let Γ be a finite Boolean constraint language. Then* Exact Ones SAT(Γ) *is* FPT *if and only if Γ is weakly separable, unless* FPT = W[1]*; and admits a polynomial kernel if and only if Γ is semi-separable and mergeable, unless* NP ⊆ co-NP/poly*.*

Similarly to Min Ones SAT(Γ), the classification of Exact Ones SAT(Γ) problems does not follow the lattice of Boolean co-clones. It is natural to ask how the characterizing partial polymorphisms relate to one another; we find that they are orthogonal: Semi-separability does not imply mergeability, e.g., $(x \oplus y \oplus z = 0)$ is semi-separable but not mergeable (thus also weak separability does not imply mergeability). Mergeability does not imply weak separability (or semi-separability), e.g., implications are mergeable but not weakly separable. Finally, answering another natural question, mergeability and weak separability do not imply semi-separability, e.g., $((x \to y) \wedge (y \neq z))$ is weakly separable and mergeable but not semi-separable.

5 Conclusion

We considered the (Boolean) CSP optimization problems Max Ones SAT(Γ) and Exact Ones SAT(Γ) from the parameterized complexity and kernelizability perspectives, and arrived at a complete classification for both problems. For Max Ones SAT(Γ), the problem had previously been classified for every choice of Γ as either in P or NP-complete [16]; we refine the NP-complete cases into cases with polynomial kernels; cases which are FPT but do not admit polynomial kernels unless NP ⊆ co-NP/poly; and two levels of hardness (W[1]-hard and in XP, respectively NP-complete for $k = 1$). For Exact Ones SAT(Γ), the problem had previously been classified into P vs NP-complete [8], and into FPT vs W[1]-complete [20]; we refine the FPT-cases as either admitting a polynomial kernel, or not admitting a polynomial kernel unless NP ⊆ co-NP/poly. This complements the previously achieved kernelizability characterization of the Min Ones SAT(Γ) problem [18].

In addition to polynomial (many-one) kernels as discussed in this paper, it is also interesting to consider the existence of so-called *Turing kernels* of polynomial size (see [15]), as these bring many of the same advantages as polynomial kernels. Such kernels are not excluded by the framework of Bodlaender et al. [2] applied in this paper, and indeed there are problems with small Turing kernels which do not admit polynomial many-one kernels unless NP ⊆ co-NP/poly [15]. In [15], a hierarchy for *kernelization hardness* was proposed, with classes defined under PPT-closure (which preserves existence of both polynomial many-one kernels and polynomial Turing kernels). The fundamental classes of the hierarchy are MK[1] (representing, essentially, problems with polynomial kernels) and WK[1] (representing the kernelization hardness version of W[1], with complete problems including Clique($k \log n$), the reparameterization of Clique(k)). Thus, a problem which is WK[1]-hard admits no kind of Turing kernelization unless all problems in WK[1] do; it is conjectured in [15] that this is not the case.

With this setup, it is not hard to show the following strengthenings of the kernelizability characterizations given above.

Corollary 5.1. *For every finite constraint language Γ, both* Exact Ones SAT(Γ) *and* Max Ones SAT(Γ) *are either in MK[1] (and admit polynomial kernels) or WK[1]-hard.*

Proof. Recall that kernelization hardness for Multiple Compatible Patterns (MCP) was shown via a PPT from Multicolored Clique($k \log n$), which is WK[1]-complete by [15]. Thus all cases of our problems for which a PPT was given from MCP are WK[1]-hard as well. For Max Ones SAT(Γ), an inspection of the proof shows that every negative case for polynomial kernelizability is either MCP-hard by PPT, Clique(k)-hard by PPT, or NP-hard for $k = 1$. In the former two cases, WK[1]-hardness follows directly; in the latter case, there is a polynomial-time many-one reduction from Clique($k \log n$) to Max Ones SAT(Γ) with parameter $k = 1$, which is certainly a PPT. For Exact Ones SAT(Γ), we need the following claims: If Exact Ones SAT(Γ) is W[1]-hard, then Clique(k) $\leq_{ppt}$ Exact Ones SAT(Γ) (see [20]); and if Min Ones SAT(Γ) is NP-hard and Γ not mergeable, then Min Ones SAT(Γ) is WK[1]-complete (see [18, 15]). The result follows since every negative case for polynomial kernelization of Exact Ones SAT(Γ) is either by PPT from MCP, or by PPT from Min Ones SAT(Γ), or by W[1]-hardness of the problem. □

In fact, it is possible to show that both Exact Ones SAT(Γ) and Max Ones SAT(Γ) break down more specifically into the cases of P, MK[1]-complete, WK[1]-complete, and W[1]-hard (making for more proper dichotomies), but we omit the proof of this here.

References

[1] F. N. Abu-Khzam. A kernelization algorithm for d-hitting set. *J. Comput. Syst. Sci.*, 76(7):524–531, 2010.

[2] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009.

[3] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. *Theoretical Computer Science*, 412(35):4570–4578, 2011.

[4] A. A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *Proc. 43th Symp. Foundations of Computer Science*, pages 649–658. IEEE, 2002.

[5] A. A. Bulatov. Tractable conservative constraint satisfaction problems. In *LICS*, page 321. IEEE, 2003.

[6] N. Creignou, S. Khanna, and M. Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*, volume 7 of *SIAM Monographs on Discrete Mathematics and Applications*. 2001.

[7] N. Creignou, P. G. Kolaitis, and B. Zanuttini. Structure identification of Boolean relations and plain bases for co-clones. *Journal of Computer and System Sciences*, 74(7):1103–1115, 2008.

[8] N. Creignou, H. Schnoor, and I. Schnoor. Non-uniform boolean constraint satisfaction problems with cardinality constraint. In *CSL*, volume 5213 of *LNCS*, pages 109–123. Springer, 2008.

[9] P. Crescenzi and G. Rossi. On the Hamming distance of constraint satisfaction problems. *Theoretical Computer Science*, 288(1):85–100, 2002.

[10] M. Dom, D. Lokshtanov, and S. Saurabh. Incompressibility through colors and ids. In *ICALP (1)*, volume 5555 of *LNCS*, pages 378–389. Springer, 2009.

[11] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, New York, 1999.

[12] P. Erdős and R. Rado. Intersection theorems for systems of sets. *J. London Math. Soc.*, 35:85–90, 1960.

[13] T. Feder and M. Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM J. Comput.*, 28(1):57–104, 1999.

[14] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[15] D. Hermelin, S. Kratsch, K. Soltys, M. Wahlström, and X. Wu. A completeness theory for polynomial (turing) kernelization. In *IPEC*, pages 202–215, 2013.

[16] S. Khanna, M. Sudan, L. Trevisan, and D. P. Williamson. The approximability of constraint satisfaction problems. *SIAM J. Comput.*, 30(6):1863–1920, 2000.

[17] S. Kratsch, D. Marx, and M. Wahlström. Parameterized Complexity and Kernelizability of Max Ones and Exact Ones Problems. In *MFCS*, volume 6281 of *LNCS*, pages 489–500. Springer Berlin Heidelberg, 2010.

[18] S. Kratsch and M. Wahlström. Preprocessing of Min Ones Problems: A Dichotomy. In *ICALP (1)*, volume 6198 of *LNCS*, pages 653–665. Springer Berlin Heidelberg, 2010.

[19] R. E. Ladner. On the structure of polynomial time reducibility. *J. Assoc. Comput. Mach.*, 22:155–171, 1975.

[20] D. Marx. Parameterized complexity of constraint satisfaction problems. *Computational Complexity*, 14(2):153–183, 2005.

[21] G. L. Nemhauser and L. E. Trotter, Jr. Vertex packings: structural properties and algorithms. *Math. Programming*, 8:232–248, 1975.

[22] G. Nordh and B. Zanuttini. Frozen boolean partial co-clones. In *ISMVL*, pages 120–125, 2009.

[23] T. J. Schaefer. The complexity of satisfiability problems. In *STOC*, pages 216–226. ACM, 1978.

[24] R. Williams. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318, 2009.