Kernelization of Generic Problems
Upper and Lower Bounds

Dissertation

zur Erlangung des Grades des
Doktors der Naturwissenschaften (Dr. rer. nat.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

vorgelegt von

Stefan Kratsch

Saarbrücken
2010

Tag des Kolloquiums: 30. August 2010

Dekan:
Professor Dr. Holger Hermanns

Berichterstatter:
Professor Dr. Kurt Mehlhorn, Max-Planck-Institut für Informatik, Saarbrücken
Professor Dr. Hans L. Bodlaender, Utrecht University, Utrecht, the Netherlands
Dr. Jiong Guo, Cluster of Excellence (MMCI), Saarbrücken

Vorsitz:
Professor Dr. Raimund Seidel, Saarland University, Saarbrücken

Akad. Mitarbeiter:
Ben Galehouse Ph.D., Max-Planck-Institut für Informatik, Saarbrücken

To my parents

Zusammenfassung

Diese Dissertation beschäftigt sich mit der Kernelisierbarkeit von generischen Problemen, definiert durch syntaktische Beschränkungen oder als Problemsystem. Polynomielle Kernelisierung ist eine Formalisierung des Konzepts der Datenreduktion für kombinatorisch schwierige Probleme. Sie erlaubt eine gründliche Untersuchung dieses wichtigen und fundamentalen Begriffs. Die Dissertation gliedert sich in zwei Hauptteile.

Im ersten Teil beweisen wir, dass alle Probleme aus zwei syntaktischen Teilklassen der Menge aller konstantfaktor-approximierbaren Probleme polynomielle Kernelisierungen haben. Die Probleme müssen durch Optimierung über Formeln in Prädikatenlogik erster Stufe mit beschränkter Quantifizierung beschreibbar sein. Eine Relaxierung dieser Beschränkungen gestattet bereits Probleme, die keine polynomielle Kernelisierung erlauben. Im Anschluss betrachten wir Kantenmodifizierungsprobleme und zeigen, dass diese im Allgemeinen keine polynomielle Kernelisierung haben.

Im zweiten Teil betrachten wir drei Arten von booleschen Constraint-Satisfaction-Problemen. Wir charakterisieren vollständig welche dieser Probleme polynomielle Kernelisierungen erlauben. Für jedes gegebene Problem zeigen unsere Resultate entweder eine polynomielle Kernelisierung oder sie zeigen, dass das Problem keine polynomielle Kernelisierung hat. Die Dichotomien sind durch Eigenschaften der erlaubten Constraints charakterisiert.

Die Arbeit ist in englischer Sprache verfasst.

Abstract

This thesis addresses the kernelization properties of generic problems, defined via syntactical restrictions or by a problem framework. Polynomial kernelization is a formalization of data reduction, aimed at combinatorially hard problems, which allows a rigorous study of this important and fundamental concept. The thesis is organized into two main parts.

In the first part we prove that all problems from two syntactically defined classes of constant-factor approximable problems admit polynomial kernelizations. The problems must be expressible via optimization over first-order formulas with restricted quantification; when relaxing these restrictions we find problems that do not admit polynomial kernelizations. Next, we consider edge modification problems, and we show that they do not generally admit polynomial kernelizations.

In the second part we consider three types of Boolean constraint satisfaction problems. We completely characterize whether these problems admit polynomial kernelizations, i.e., given such a problem our results either provide a polynomial kernelization, or they show that the problem does not admit a polynomial kernelization. These dichotomies are characterized by properties of the permitted constraints.

The thesis is written in English.

Acknowledgments

First of all I wish to thank Kurt Mehlhorn for giving me both motivating supervision and helpful advice as well as the freedom and confidence that made this thesis possible. I am grateful to Magnus Wahlström for introducing me to constraint satisfaction problems and for the patience this required. Similarly, I have to thank Pascal Schweitzer for sharing his insight into graph isomorphism and related problems of unknown complexity. I found it a pleasure to be a part of the algorithms and complexity group at MPI, in research as well as in social activities. I am greatly indebted to my parents for their ongoing support, encouragement, and advice; not least for reminding me to take the occasional vacation.

Finally, I want to thank the usual suspects for our long evenings of applied game theory; with a special thanks to the monster for not eating all the meeples.

Contents

I. Introduction 1

II. Upper and lower bounds for kernelization 11

1. Parameterized complexity and kernelization 13

1.1. Notation . . . 13
1.2. Parameterized complexity . . . 15
1.3. Kernelization . . . 18
1.4. Polynomial lower bounds for kernelization . . . 21

2. Sunflowers 27

2.1. Introduction . . . 27
2.2. Application in kernelization . . . 31

3. Polynomial kernelizations for MIN $F^+\Pi_1$ and MAX NP 33

3.1. Introduction . . . 33
3.2. Syntactical problems . . . 35
3.3. Polynomial kernelization for MIN $F^+\Pi_1$ . . . 37
3.4. Polynomial kernelization for MAX NP . . . 42
3.5. Linear kernelization for MAX SNP . . . 48
3.6. Summary . . . 49

4. Lower bounds for edge modification problems 51

4.1. Introduction . . . 51
4.2. Hardness of Not-1-In-3 SAT . . . 53
4.3. Lower bounds for two H-free modification problems . . . 57
4.4. Summary . . . 61

III. Kernelizability of Boolean constraint satisfaction problems 63

5. Boolean constraint satisfaction problems 65

5.1. Introduction . . . 65
5.2. Definitions . . . 67

6. Min ones constraint satisfaction problems **75**
6.1. Introduction . . . 75
6.2. Mergeability . . . 76
6.3. Characterization I: Polynomial kernelization . . . 78
6.4. Characterization II: Lower bound . . . 85
6.5. Summary . . . 94

7. Max ones constraint satisfaction problems **97**
7.1. Introduction . . . 97
7.2. Characterization . . . 98
7.3. Summary . . . 105

8. Exact ones constraint satisfaction problems **107**
8.1. Introduction . . . 107
8.2. Definitions and related results . . . 108
8.3. Characterization I: Zero-valid, Horn, and width-2 affine . . . 110
8.4. Characterization II: Remaining cases . . . 114
8.5. Summary . . . 120

IV. Conclusion and open problems **121**

9. Conclusion **123**
9.1. Beyond MIN $F^+\Pi_1$ and MAX NP . . . 124
9.2. Kernelization and approximation revisited . . . 127

10. Open problems and further research **129**

Bibliography **135**

Part I.

Introduction

Introduction

Preprocessing and data reduction are ubiquitous in the practical implementation as well as in the theoretical study of algorithms. While they will not fully solve a given problem, they are known to give significant speed-ups or to make a computation feasible at all. The latter task, i.e., reducing instances of hard problems to a computationally hard core, shall be the main focus of this thesis.

While, at first, we may have the impression that data reduction and preprocessing are artificial concepts solely related to computers, they are in fact very natural: As human beings we apply them all the time, for example to reach decisions:

- We often begin by eliminating the impossible.
- We are able to ignore parts of the problem that will not affect our decision.
- We discard a possible decision if we find another one that is at least as good.

We know that these ways of thinking have long since found their way into our algorithms, i.e., into our "recipes" for solving problems and making decisions by means of a computer. Already in abstracting and modeling a problem we are ignoring most of the possible input: One may think about the huge difference between actual scenery and landscape as opposed to a weighted graph, for the purpose of finding a shortest path or roundtrip in said scenery. Elimination of the impossible is often already implicit in the solution space that an algorithm explores. A nice example may be found in the Simplex algorithm for solving linear programs: Only feasible points are visited, in fact, only those which are extremal points of the polytope of feasible solutions (recall that extremal points are described by a small number of tight constraints); this avoids the possibly high number of constraint combinations that do not describe feasible solutions. The strategy to visit only extremal points is based on the well-known fact that for any optimal solution there is one of the same value which corresponds to a vertex of the polytope; this is a good example of the third paradigm.

Many more examples are known for problems that are computationally hard: Vertices of low degree can often be removed (in many graph problems), and variables that occur only signed or only unsigned (in satisfiability) can be fixed to true or false accordingly. Furthermore, there are often ways of solving problems when there are independent components, or we may have lower bounds on the solution cost obtained from relaxations of a problem (which then allow to reject an instance if the lower bound already violates the budget). However, in many cases the success of such *reduction rules* is only proven by experiment and we do not know any performance guarantees. In fact, some of the mentioned examples serve only to simplify the design or the analysis of algorithms.

Clearly, as theoreticians, we are interested not only in knowing certain efficient rules of how to simplify instances of a given problem, but we also desire to measure the actual impact of such rules. However, upon closer inspection, we quickly come across a well-known fact: There can be no efficient data reduction that provably shrinks every instance of a hard problem, unless the problem itself can be efficiently solved. Or, adopting polynomial-time solvability (i.e., membership in P) as our notion of efficiency and NP-completeness as our notion of hardness, there cannot be a polynomial-time data reduction unless P = NP (in this strict sense of reducing the size of every instance). The simple reason is that by each run of the data reduction we shrink the problem instance, until we reach an instance of constant size which we can solve by a look-up table.

Based on this discouraging observation about data reduction, which contrasts the fact that it is so widely applied, there are two perspectives: The first one being that data reduction is a heuristic approach, not open to theoretical analysis. The second is that we should take a closer look at the issue, and possibly redefine our concept of data reduction, in order to make it more robust. Let us follow the second possibility.

We have seen that there cannot be a polynomial-time algorithm that reduces the size of *every* instance of some NP-hard problem. Why should we not simply accept the fact that there may be instances on which we cannot make any significant progress in polynomial time? Following this train of thought, we would assume that the size of such instances "matches" their inherent complexity. Thus we require some measure of difficulty beyond merely considering the size of an instance; we saw that size alone is an ill-suited complexity measure in the context of data reduction.

Parameterized complexity Parameterized complexity is a multivariate approach to measuring and understanding the complexity of combinatorially hard problems. In this paradigm, the complexity of problems is investigated in finer detail than the polynomial-time solvability versus NP-hardness of classical complexity. With each instance of a problem an integer value is associated, the so-called *parameter*, intended to capture the complexity of the problem. The time complexity of algorithms is then measured with respect to the size *and* the parameter. There are different ways of motivating this approach; let us first consider polynomial-time algorithms.

While we expect that NP-hard problems cannot be solved in polynomial time, we find many special cases which we can solve efficiently. These special cases are frequently obtained by restricting some parameter of the instances to bounded (i.e., constant) values. For example, many graph problems can be solved in polynomial time on graphs of bounded treewidth or bounded degree, respectively. Let us note, however, that there are two fundamentally different runtime classes, both of which are polynomial for bounded values of the parameter, say k: The runtime may be $n^{f(k)}$ or it may be $f'(k) \cdot n^c$, for some constant c independent of k and functions f, f' (typical runtimes are $n^{\mathcal{O}(k)}$ and $\alpha^k \cdot n^c$). Thus for the latter type of runtime we even get a *uniformly* polynomial-time algorithm for bounded values of k, i.e., the runtime is $\mathcal{O}(n^c)$ for every fixed value of k. Algorithms with such a runtime scale much better when we increase k or n, whereas a runtime of $n^{f(k)}$ quickly becomes impractical. Parameterized complexity, amongst others, is a theory about when we can get algorithms of runtime $f(k) \cdot n^c$, so-called *fixed-parameter tractable* algorithms, or *fpt-algorithms*. FPT denotes the class of all fixed-parameter tractable problems, i.e., parameterized problems that have fpt-algorithms.

Another way of arriving at parameterized complexity starts with exponential-time algorithms. Assuming P $\neq$ NP, we expect to need worst-case exponential time with respect to the input size. Actually, we only know that super-polynomial runtime is required, but the widely believed *Exponential Time Hypothesis* (ETH) states that there is no subexponential-time algorithm for k-SAT for any $k > 2$ (see Woeginger's [112] nice survey on exact algorithms for NP-hard problems). However, we may be able to confine the combinatorial explosion to a parameter independent of the input size, and have only polynomial dependence on the input size. Again, this leads to desired runtimes of the form $f(k) \cdot n^c$, avoiding $\mathcal{O}(\alpha^n)$. From this perspective, fixed-parameter tractability is a relaxation of polynomial-time solvability.

A classic example of $f(k) \cdot n^c$ versus $n^{f'(k)}$ runtimes is that of VERTEX COVER versus CLIQUE; the two problems are equivalent in classical complexity. Both problems can be decided in time $n^{\mathcal{O}(k)}$ where k is the number of vertices in the cover or the clique, by simple enumeration of all size-k vertex subsets. However, while VERTEX COVER is known to be solvable in time $\mathcal{O}(1.274^k \cdot kn)$ [34], no such algorithm, even for larger functions $f(k)$, is known for CLIQUE. It is known that CLIQUE is complete for W[1], a parameterized analogue of NP, and showing W[1]-hardness is a standard tool of proving parameterized intractability. Interestingly, a paper by Chen et al. [32] shows that, assuming FPT $\neq$ W[1] (a fundamental hypothesis in parameterized complexity, which is also implied by ETH), there is no $f(k) \cdot n^{o(k)}$ time algorithm for CLIQUE for *any* function f.

Kernelization Now, using the framework of parameterized complexity, let us return to the issue of finding a reasonable definition for efficient data reduction. We have seen that we cannot simplify *all* instances of NP-hard problems in polynomial time. Furthermore, we have seen the notion of fixed-parameter tractability, where the super-polynomial part of the runtime is restricted to some parameter. For such problems, we know that the chosen parameter is a good measure of the complexity; simply because small parameter values mean that the runtime will be small as well. Hence, we would assume that the impact of data reduction should be limited in terms of the parameter. It is only natural to take this point of view as the basis for our definition.

A *kernelization* is a polynomial-time mapping $K : (I, k) \mapsto (I', k')$ such that the instances (I, k) and (I', k') are equivalent and the new parameter value k' as well as the size of I' are bounded by some function h of the parameter k. A kernelization is a *polynomial kernelization* if h is a polynomial. Note how this definition avoids the issue of instances which cannot be reduced any further, since we do not aim to shrink *all* instances, but try to reach a size depending on the parameter. Thus, multiple applications of a kernelization do not necessarily solve the problem.

Having defined kernelization in the context of parameterized complexity, we quickly see that it is also a technique for showing that a problem is fixed-parameter tractable.

Indeed, by reducing a (decidable) problem to a computationally hard core of size bounded by the parameter, we can apply any brute force algorithm to decide the core and obtain an fpt-algorithm. Interestingly, the converse holds as well and we have the following (folklore) relation: Any parameterized problem is fixed-parameter tractable if and only if it is decidable and admits some (not necessarily polynomial) kernelization. The proof for the converse is surprisingly simple (but it is often wrongly assumed that f is computable, see the discussion in [14]): Assume that we have some fpt-algorithm of runtime $f(k) \cdot n^c$. Given some instance, we run the algorithm for n^{c+1} steps. If it terminates then, according to the output, we return a dummy yes- or no-instance. If it does not terminate, then we must have that $n < f(k)$ and we may return the original instance as the kernel (of size at most $f(k)$). Thus every FPT problem has a kernel of size matching the runtime of any fpt-algorithm for the problem. Let us emphasize, however, that the kernelizations implied by this relation have fairly bad size guarantees, indeed $f(k)$ must be superpolynomial for NP-hard problems (assuming meaningful parameter values, e.g., $k \leq n$). In recent literature polynomial kernelizations, i.e., with polynomial size guarantee, have been adopted as the notion for efficient and effective data reduction; of course smallest possible kernels are highly sought after. From a practical point of view, kernelizations are toolboxes of reduction rules and smaller kernelizations mean better reduction rules; such a data reduction can be an invaluable opening step in any practical solution for a problem. From a complexity theoretic point of view, a kernelization is a compression and we want to know when, or for what sizes $h(k)$, this is possible.

Kernelization results: upper bounds Kernelization has received significant interest over the previous two decades, maturing from a technique to prove fixed-parameter tractability into its own field of research. In recent years there has been a large number of kernelization results for a variety of problems. We will highlight a few well-known results as well as some very recent developments; for further results we refer the reader to the excellent surveys of Guo and Niedermeier [63] as well as of Bodlaender [14]. All kernelization results are given with respect to the number of vertices or variables unless stated otherwise; this is a general habit since these values are strongly related to the runtimes. There is a linear kernel for VERTEX COVER due to Chen et al. [33] (based on a theorem by Nemhauser and Trotter [93]), a quadratic kernel for FEEDBACK VERTEX SET due to Thomassé [110], a linear kernel for FEEDBACK ARC SET in tournament graphs due to Bessy et al. [11], and a polynomial kernel for MULTICUT in trees due to Bousquet et al. [22] (the parameterized complexity of MULTICUT on general graphs is a famous open problem). Let us also mention the $\mathcal{O}(k^{d-1})$ kernel for d-HITTING SET by Abu-Khzam [1] as well as the $\mathcal{O}(k^{|V(H)|-1})$ kernel for the problem of packing k disjoint copies of some fixed graph H due to Moser [90]; both problems have an inherent notion of arity (i.e., d respectively $|V(H)|$), similar to problems considered in this thesis.

Recently Bodlaender et al. [17] presented polynomial kernelizations for a large class of problems on planar graphs, and also generalized their results to graphs of bounded genus. The first result of this type, a linear kernelization for DOMINATING SET on planar graphs, is due to Alber, Fellows, and Niedermeier [2]. Their work was followed by linear kernelizations for a large number of problems, and was generalized by Guo and Niedermeier [64] based on a region decomposition of the planar input graph and problem-specific reduction rules that shrink the size of the regions. Bodlaender et al. [17] showed how to obtain general reduction rules using certain properties of the problem. Fomin et al. [55] gave a similar meta-theorem about quadratic kernelizations for a large number of problems on graph classes that avoid some fixed minor.

Kernelization results: lower bounds Recently Bodlaender, Downey, Fellows, and Hermelin [16] presented the first negative results concerning the existence of polynomial kernelizations for some natural fixed-parameter tractable problems. Using their notion of a *distillation algorithm* and results due to Fortnow and Santhanam [56], they were able to show that the existence of polynomial kernelizations for so-called *compositional* parameterized problems implies a collapse of the polynomial hierarchy to the third level. Thus there are fairly simple FPT problems like the k-PATH problem (recently shown to be solvable in randomized time $2^k \cdot n^{\mathcal{O}(1)}$ [111]) which do not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly (which is known to imply the mentioned collapse of the polynomial hierarchy [113]). This negative toolkit has been successfully applied to a number of other problems [18, 46, 53]. A paper by Dell and van Melkebeek [44] refines the framework and proves concrete polynomial lower bounds for problems that do admit some polynomial kernelization. In fact, by means of a two-player oracle communication game, Dell and van Melkebeek exclude any compression of cost below a certain bound, e.g., to instances of a different problem or in multiple rounds.

Obtaining general results for kernelization Naturally, when studying an interesting notion like kernelization, we desire to obtain results which are as general as possible. For kernelization, the results by Bodlaender et al. [17] as well as by Fomin et al. [55] are two strong positive results that provide general kernelizations for certain classes of problems. The lower bound framework by Bodlaender et al. [16] provides a general strategy of excluding polynomial kernelizations for compositional problems, though in most cases it seems nontrivial to prove compositionality (e.g., see [46]).

In order to obtain general results, we have to focus on certain restricted classes of problems; clearly some parameterized problems are unlikely to admit polynomial kernelizations (e.g., it is easy to come up with problems that are NP-hard even for bounded values of the parameter). Common complexity classes are either of computational or of syntactical nature, i.e., they are defined by algorithmic properties (e.g., the class APX of constant-factor approximable problems) or by requiring the problems to be expressible in a certain way (e.g., graph problems definable in monadic second order logic). We will focus on classes of the latter type, since they provide some structural insight into the problems; in fact most general results are based on structural restrictions of the problems and/or restrictions on the input instances. It goes without saying that a main purpose of syntactical classes is that of allowing abstraction and generalization. Concretely, we consider the classes MAX SNP, MAX NP, and MIN $\mathrm{F}^{+}\Pi_1$ as well as certain types of Boolean constraint satisfaction problems.

	Approximation ratio		Kernel size	
VERTEX COVER	2	[66]	$\mathcal{O}(k)$	[33]
CONNECTED VERTEX COVER	2	[106]	not polynomial (*)	[46]
FEEDBACK VERTEX SET	2	[9]	$\mathcal{O}(k^2)$	[110]
BIN PACKING	1.5	[109]	none unless P=NP	[58]
SET SPLITTING	1.38	[4]	$\mathcal{O}(k)$	[85]
MAX CUT	1.1383	[59]	$\mathcal{O}(k)$	[86]
MAX SAT	1.2987	[8]	$\mathcal{O}(k)$	[86]
TRIANGLE PACKING	$1.5 + \epsilon$	[70]	$\mathcal{O}(k^2)$	[90]
MINIMUM FILL-IN	$\mathcal{O}(\text{opt})$	[91]	$\mathcal{O}(k^2)$	[91]
TREEWIDTH	$\mathcal{O}(\sqrt{\log \text{opt}})$	[51]	not polynomial (*)	[16]

Table 0.1.: Approximation ratio and size of problem kernels for some optimization problems. The kernel sizes are with respect to the number of vertices or variables, respectively. (*) CONNECTED VERTEX COVER does not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly. TREEWIDTH does not admit a polynomial kernelization unless there is an and-distillation algorithm for all NP-complete problems [16]. Though unlikely, this is not known to imply a collapse of the polynomial hierarchy.

Relation of kernelization and approximation Many syntactically defined classes of problems were introduced with the aim of getting a better understanding of the approximation properties of optimization problems, e.g., MAX SNP and MAX NP [99]. Similar to the parameterized complexity, the approximability of two optimization problems may differ greatly, even if the decision versions of the two problems are both NP-complete. There are a few proven connections between parameterized complexity and approximability. The best known one may be the fact that the decision versions of problems which admit an efficient polynomial-time approximation scheme (EPTAS) are fixed-parameter tractable (proven by Bazgan [10] and independently by Cesati and Trevisan [29]); an EPTAS is a family of $(1 + \epsilon)$-approximation algorithms of runtime $f(1/\epsilon) \cdot n^c$. Note that this result requires the target function to be integer valued. The essential idea is that running an EPTAS with $\epsilon = \frac{1}{2k}$ allows to correctly determine whether the optimum value is $\leq k$ (or $\geq k$, resp.). Using the same trick, it is easy to see that problems with a fully polynomial-time approximation scheme (FPTAS), i.e., when f is a polynomial, are FPT with a function f that is polynomial in k. Thus, using the relation between fixed-parameter tractability and kernelization, we see that such problems admit a polynomial kernelization. However, this "insight" is neither deep nor particularly helpful, since the parameter values of NP-hard problems which can be solved in time $\mathcal{O}(k^c \cdot n^{c'})$ must be super-polynomial in n (assuming P $\neq$ NP).

Considering known approximation and kernelization results (e.g., see Table 0.1) it would be tempting to conjecture a stronger relation, namely one between constant-factor approximability and the admittance of polynomial kernelizations. While some examples rule out a general correspondence, e.g., BIN PACKING and CONNECTED VERTEX COVER, we may be able to prove a strong correspondence when restricting our attention to certain syntactical classes or frameworks like constraint satisfaction problems. Some of the work presented in this thesis was motivated by this and we will see it in many of the results.

Our work This thesis studies kernelization, in particular polynomial kernelizations, by considering the kernelizability of whole classes of problems. We show that all problems from MIN $F^+\Pi_1$, MAX SNP, and MAX NP admit polynomial kernelizations (i.e., their natural decision versions do); note that all these problems were known to be constant-factor approximable. Then, we consider edge modification problems and show that (under standard assumptions) these problems do not generally admit polynomial kernelizations, even in the restricted case where they are known to be FPT by a simple branching algorithm. Finally, we address the polynomial kernelizability of three types of Boolean constraint satisfaction problems and give a full characterization (a dichotomy) into admittance or non-admittance of polynomial kernelizations (under standard assumptions) for each of the variants.

Organization The following chapters of the thesis are organized into three parts. In the first part, we introduce the general methodology of kernelization and present upper and lower bound results. The second part contains the results for Boolean constraint satisfaction problems (CSPs) as well as an introduction to that area. In the third part, we conclude the thesis and discuss some open problems.

The first part begins with an introduction to parameterized complexity and kernelization (Chapter 1), in particular we will consider techniques for proving upper and lower bounds for kernelization. In Chapter 2, we introduce sunflowers, a notion from extremal set theory playing an important role for our kernelizations, and show their application on a simple example. Then, in Chapter 3, we show that the natural decision versions of all problems in MIN $F^+\Pi_1$, MAX SNP, and MAX NP admit polynomial kernelizations, and also extend this result to weighted versions of the problems. In Chapter 4, we consider the kernelizability of edge modification problems; we exhibit a small graph H such that H-free EDGE DELETION and H-free EDGE EDITING do not admit polynomial kernelizations.

The second part starts with an introduction to Boolean constraint satisfaction problems, i.e., generalized satisfiability problems (Chapter 5). In Chapters 6 and 7, we consider min ones and max ones CSPs, i.e., the problem of minimizing respectively maximizing the number of true variables in a satisfying assignment. In Chapter 8, we address the task of finding satisfying assignments with a specified number of true variables. In all three cases we are able to fully characterize when the problems admit polynomial kernelizations, depending on the constraint language.

In the third part, we give a conclusion as well as a discussion of some open problems that relate to our work (Chapters 9 and 10). We will also revisit the relation of approximability and kernelization and relate parts of our work to known approximability results.

Part II.

Upper and lower bounds for kernelization

1. Parameterized complexity and kernelization

1.1. Notation

Graphs We consider undirected, finite, simple graphs $G = (V, E)$ where V is a finite set and $E \subseteq \binom{V}{2}$. The (open) neighborhood of a vertex $v \in V$ is the set $N(v) = \{u \mid \{u, v\} \in E\}$; the closed neighborhood is the set $N[v] = N(v) \cup \{v\}$. An *induced subgraph* of G is a graph $G' = (V', E')$ with $V' \subseteq V$ and with $E' = \{\{u, v\} \in E \mid u, v \in V'\}$. We denote this subgraph by $G[V']$. We also use the following abbreviations, let $G = (V, E)$:

$\boldsymbol{G - S}$ For a set $S \subseteq V$ of vertices, the graph $G - S$ is obtained by deleting all vertices in S, i.e., $G - S = G[V \setminus S]$.

$\boldsymbol{G - D}$ For a set $D \subseteq E$ of edges, the graph $G - D$ is obtained by deleting all edges in D, i.e., $G - D = (V, E \setminus D)$.

$\boldsymbol{G \triangle D}$ For a set $D \subseteq \binom{V}{2}$, the graph $G \triangle D$ is obtained by adding all edges in $D \setminus E$ and deleting all edges in $D \cap E$, i.e., $G \triangle D = (V, E \triangle D)$ where $E \triangle D = (E \setminus D) \cup (D \setminus E)$.

A graph G is *H-free* if it does not contain H as an induced subgraph (i.e., an induced subgraph G' which is isomorphic to H). For a set $\mathcal{H}$ of graphs, G is *$\mathcal{H}$-free* if it is H-free for every graph $H \in \mathcal{H}$. We denote by K_i, C_i, and P_i the clique, cycle, and path of i vertices respectively. The graph $2K_2$ is the disjoint union of two K_2.

Hypergraphs A *hypergraph* is a tuple $\mathcal{H} = (V, E)$ consisting of a finite set V, its vertices, and a family E of subsets of V, its edges. A hypergraph has *dimension d* if each edge has cardinality at most d. A hypergraph is *d-uniform* if each edge has cardinality exactly d. We often use hypergraphs in place of families of sets; each set family has a canonical hypergraph associated to it by letting the set of vertices be the union of all sets that it contains.

NP optimization problems *NP optimization problems* (NPO problems) are a restricted form of optimization problems, with the additional requirement that solutions shall be efficiently verifiable (cf. [41]). While we do not formally introduce our problems as NP optimization problems, their definition provides a good basis for defining some related terms, e.g., approximation algorithms. An NP optimization problem is a 4-tuple $\mathcal{Q} = (I, \mathrm{sol}, m, \mathrm{goal})$:

- I is the set of *instances* and it can be recognized in polynomial time.

- $\text{sol}(x)$ denotes the set of *feasible solutions*. There is a polynomial p such that the size of any solution $y \in \text{sol}(x)$ is bounded by $p(|x|)$, and for any x and y, with $|y| \leq p(|x|)$, it can be decided in polynomial time whether $y \in \text{sol}(x)$.
- $m(x, y)$ denotes the positive integer *measure* of y with respect to x; it is defined for each $x \in I$ and $y \in \text{sol}(x)$ and can be computed in polynomial time. It is also called the *objective function*.
- $\text{goal} \in \{\min, \max\}$ denotes whether the aim is to find feasible solutions with smallest or largest possible measure, respectively.

The *optimum value* of an instance x is defined as

$$\text{opt}_{\mathcal{Q}}(x) = \text{goal}\{m(x, y) \mid y \in \text{sol}(x)\}.$$

An *optimum solution* for $x \in I$ is a feasible solution $y \in \text{sol}(x)$ with $m(x, y) = \text{opt}_{\mathcal{Q}}(x)$.

The problem $\mathcal{Q}$ is *polynomially bounded* if there exists a polynomial q such that for any instance x and any feasible solution y the measure of y is bounded by $q(|x|)$, i.e., $m(x, y) \leq q(|x|)$. The class *NPO* contains all NP optimization problems and *NPO-PB* is the class of all polynomially bounded NP optimization problems.

Approximability Let $\mathcal{Q} = (I, \text{sol}, m, \text{goal})$ be an NPO problem. The *performance ratio* of a feasible solution $y \in \text{sol}(x)$ for an instance $x \in I$ is defined as

$$R(x, y) = \max\left\{\frac{m(x, y)}{\text{opt}_{\mathcal{Q}}(x)}, \frac{\text{opt}_{\mathcal{Q}}(x)}{m(x, y)}\right\},$$

thus the performance ratio is always at least one.

An algorithm A is $\rho(n)$*-approximate* for $\mathcal{Q}$ if, for each instance x, we have $A(x) \in \text{sol}(x)$ and

$$R(x, A(x)) \leq \rho(|x|),$$

where $\rho : \mathbb{N} \to \mathbb{R}_{\geq 1}$ is an arbitrary function. Let us also recall some standard complexity classes defined via approximability in polynomial time:

poly-APX All NPO problems that have polynomial-time $\rho(n)$-approximate algorithms where ρ is a polynomial.

APX All NPO problems that have polynomial-time $\mathcal{O}(1)$-approximate algorithms.

APX-PB All APX problems that are polynomially bounded.

PTAS All NPO problems that have $(1+\epsilon)$-approximate algorithms of runtime $\mathcal{O}(n^{f(1/\epsilon)})$ for any $\epsilon > 0$, where f is an arbitrary function.

EPTAS All NPO problems that have $(1+\epsilon)$-approximate algorithms of runtime $\mathcal{O}(f(1/\epsilon)n^c)$ for any $\epsilon > 0$, where f is an arbitrary function.

FPTAS All NPO problems that have $(1+\epsilon)$-approximate algorithms of runtime $\mathcal{O}(p(1/\epsilon)n^c)$ for any $\epsilon > 0$, where p is a polynomial.

1.2. Parameterized complexity

The field of parameterized complexity was pioneered by Rod Downey and Michael Fellows, who started its systematic study; see their monograph [47] for an introduction to the field including a good amount of historical notes. In this section we introduce the central notions of parameterized tractability and intractability; the definitions follow [47, 54].

Parameterized problems

Instances of parameterized problems come with an additional component, the so-called *parameter*. Accordingly, a *parameterized problem*, say $\mathcal{Q}$, over some finite alphabet Σ is defined as a subset of $\Sigma^* \times \mathbb{N}$; the second component is the parameter. We introduce parameterized problems in the following (widely adopted) form:

> **<Problem name>**
>
> **Input:**
>
> **Parameter:**
>
> **Task:**

The intention is that the components *input* and *task* constitute the classical version of the problem, whereas *parameter* highlights the chosen parameter (even if the chosen number is already part of the classical problem). Note that multiple parameters can be incorporated by using the sum or the maximum of the parameter values.

Remark. Some authors prefer to define a parameterized problem as a pair $(\mathcal{Q}, \kappa)$, where $\mathcal{Q}$ is a set of strings over Σ and κ is a *parameterization of* Σ^*, i.e., a polynomial-time computable mapping from Σ^* to the natural numbers (cf. [54]). Intuitively, the parameterization assigns a parameter value to each instance of the classical problem $\mathcal{Q}$. This perspective is nice for studying different parameterizations of the same problem as well as other complexity related issues. For this thesis we adopt the simpler (and more frequently used) definition as a subset of $\Sigma^* \times \mathbb{N}$. Most problems under consideration come with one or more integer values, e.g., asking whether the solution has cost at most k, and the considered parameterization would simply map the instance to k.

Before giving the remaining definitions let us recall the well-studied VERTEX COVER problem which occurs in many of the examples throughout this chapter:

> **VERTEX COVER**
>
> **Input:** A graph G and an integer k.
>
> **Parameter:** k.
>
> **Task:** Decide whether there is a set S of at most k vertices such that each edge of G has an endpoint in S.

Fixed-parameter tractability

A parameterized problem $\mathcal{Q}$ is *fixed-parameter tractable* if there is an algorithm that decides whether $(I, k) \in \mathcal{Q}$ in time $f(k) \cdot |I|^c$, where $f(k)$ is a function depending only on k and c is a constant independent of k. This captures exactly the idea of having an algorithm that runs in uniformly polynomial time for every fixed value of the parameter, indeed the $f(k)$ part gives a (possibly exponential in k) constant factor. Such an algorithm is called an *fpt-algorithm* for $\mathcal{Q}$ and FPT is the class of all parameterized problems which are fixed-parameter tractable. The class XP captures all parameterized problems which have a (not necessarily uniformly) polynomial-time algorithm for every fixed value of the parameter, i.e., problems with algorithms of runtime $|I|^{f(k)}$. Clearly FPT $\subseteq$ XP.

Remark. This (standard) definition of fixed-parameter tractability is also called uniform fixed-parameter tractability (here uniform relates to the fact that there is a single algorithm for all k). For *strongly uniformly FPT* there is the additional requirement of f being computable (as it is the case for most of the algorithmic results). A relaxed version called *non-uniform FPT*, requires that for each fixed k there is an $\mathcal{O}(n^c)$ time algorithm that decides all instances of parameter value k, i.e., there may be a different algorithm for each k. Results of the latter type come up, for example, when invoking the graph minors theorem due to Robertson and Seymour [104] (cf. [12]).

Parameterized intractability

There are two common ways to establish that a problem is unlikely to be fixed-parameter tractable. The first, and simple, way is to prove NP-hardness for some fixed value of the parameter (e.g., k-COLORING is NP-hard for $k = 3$); this implies that the problem is not in XP unless P = NP. However, many parameterized problems are trivially contained in XP (e.g., if it suffices to enumerate all solutions of size k) while still not being known to be fixed-parameter tractable. To further classify parameterized problems between FPT and XP a notion of reducibility is used (as it is a common strategy in complexity theory). Clearly, polynomial-time many-one reductions cannot help to further classify NP-complete problems. A *parameterized reduction* or *fpt-reduction* is permitted to use fpt-time, i.e., $f(k) \cdot |I|^c$, but must guarantee that the new parameter value is bounded by a function in the original parameter. Formally:

Definition 1.1. Let $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ and $\mathcal{Q}' \subseteq \Sigma'^* \times \mathbb{N}$ be parameterized problems. A *parameterized reduction* from $\mathcal{Q}$ to $\mathcal{Q}'$ is a mapping

$$\pi : \Sigma^* \times \mathbb{N} \to \Sigma'^* \times \mathbb{N} : (I, k) \mapsto (I', k')$$

that can be computed in time $f(k) \cdot |I|^c$ for some function $f : \mathbb{N} \to \mathbb{N}$ such that:

- $(I, k) \in \mathcal{Q}$ if and only if $(I', k') \in \mathcal{Q}'$ and
- $k' \leq g(k)$ for some function $g : \mathbb{N} \to \mathbb{N}$.

It is easy to see that parameterized reductions compose and that a parameterized reduction from $\mathcal{Q}$ to $\mathcal{Q}'$ together with an fpt-algorithm for $\mathcal{Q}'$ constitute an fpt-algorithm for $\mathcal{Q}$ (i.e., FPT is closed under parameterized reductions). Let us point out that many of the classic reductions (cf. [58]) are not parameterized reductions, e.g., VERTEX COVER and INDEPENDENT SET are equivalent under polynomial-time many-one reductions, but a vertex cover of size at most k being equivalent to an independent set of size at least $n-k$ does not give a parameterized reduction ($n-k$ cannot be bounded by a function only in k). The standard way to prove (likely) parameterized intractability of some problem is to show that it is W[1]-hard; W[1] being a parameterized analogue of NP. For the W[1]-hardness proofs in this thesis it suffices to know that problems complete for W[1] are those which are equivalent to the INDEPENDENT SET problem under parameterized reductions:

INDEPENDENT SET

Input: A graph G and an integer k.

Parameter: k.

Task: Decide whether G contains an independent set on at least k vertices.

The *W-hierarchy* of classes W[i] between FPT and XP is defined based on a satisfiability problem on decision circuits:

Definition 1.2. A *decision circuit* C is a directed acyclic graph having exactly one vertex of out-degree zero, called the *output (gate)*; the vertices of in-degree zero are called the *variables*. The remaining vertices are either AND, OR, or NOT vertices, NOT vertices have in-degree equal to one; all these vertices are called *gates*.

A *(truth) assignment for a decision circuit* C is a mapping of the variables of C to values 0 or 1 (false or true, respectively). Given an assignment, the value of any gate vertex of C depends in the canonical way on its in-neighbors (e.g., for an AND vertex it is the product of the values of its in-neighbors); these values are well-defined since the circuits are acyclic. An assignment for C is a *satisfying assignment for* C if the value of C's output vertex is 1 (true).

The *depth* of a decision circuit C is the maximum number of gates on any directed path from a variable to the output gate. The *weft*, the *large gate depth*, is the maximum number of gates of fan-in greater than two on any directed path from a variable to the output gate.

The defining problem for the W-hierarchy is that of WEIGHTED WEFT t DEPTH h CIRCUIT SATISFIABILITY:

WEIGHTED WEFT t DEPTH h CIRCUIT SATISFIABILITY

Input: A weft t depth h decision circuit C and an integer k.

Parameter: k.

Task: Decide whether C has a satisfying assignment with exactly k true variables.

The complexity class $W[i]$ is defined to contain all parameterized problems which reduce to WEIGHTED WEFT i DEPTH h CIRCUIT SATISFIABILITY for some $h \in \mathbb{N}$ by an fpt-reduction. This gives the following hierarchy:

$$\mathrm{FPT} \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \ldots \subseteq \mathrm{W}[t] \subseteq \ldots \subseteq \mathrm{XP}$$

We mention the parameterized complexity of a few well-known problems:

FPT: VERTEX COVER, FEEDBACK VERTEX SET, TREEWIDTH

W[1]-complete: INDEPENDENT SET, CLIQUE, VC DIMENSION

W[2]-complete: DOMINATING SET, SET COVER, HITTING SET

W[t]-hard for all t: BANDWIDTH

Optimization problems

The different classes of parameterized problems introduced so far are defined to contain only decision problems. To study optimization problems we consider the respective *standard parameterizations*. The standard parameterization of some unparameterized minimization (resp., maximization) problem is the decision version, asking whether the optimum value is at most k (resp., at least k) for some integer k, parameterized by k.

Further reading

For general introductions to parameterized complexity we refer the reader to the recent monographs [47, 54, 94]. There are also a number of interesting surveys:

Kernelization We will formally introduce kernelization in the following section. Nice surveys of recent trends were given by Guo and Niedermeier [63] as well as by Bodlaender [14]. The progress in between these two articles is fairly impressive.

Parameterized approximation Marx [89] gives a comprehensive survey of connections between parameterized complexity and approximation as well as of parameterized approximation algorithms, including many directions for future research.

Ecology of parameters and problem parameterization Finding and choosing a parameter (or multiple parameters) is an important and interesting issue in parameterized complexity, addressed by Fellows et al. [52] as well as by Niedermeier [95].

1.3. Kernelization

A *kernelization* of a parameterized problem $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ is a polynomial-time computable mapping

$$K : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N} : (I, k) \mapsto (I', k'),$$

such that there is a function $h : \mathbb{N} \to \mathbb{N}$ such that for all $(I, k) \in \Sigma^* \times \mathbb{N}$:

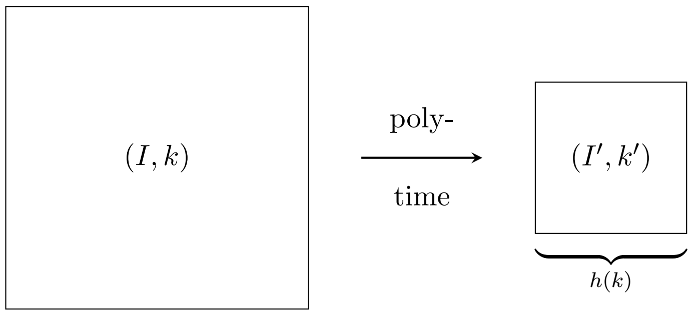

Figure 1.1.: A schematic presentation of kernelizations. The instances (I, k) and (I', k') are equivalent and k' is bounded by $h(k)$.

- $(I, k) \in \mathcal{Q}$ if and only if $(I', k') \in \mathcal{Q}$,
- the size of (I', k') is at most $h(k)$, and
- k' is bounded by $h(k)$.

We call h the *size* of the *problem kernel* $K((I, k))$. The kernelization K is *polynomial* if h is a polynomial. We say that $\mathcal{Q}$ *admits a (polynomial) kernelization* if there exists a (polynomial) kernelization of $\mathcal{Q}$ (see Figure 1.1).

In most cases, kernelizations are based on reduction rules which simplify the problem instances by removing or replacing certain (local) structures. Then, when none of the rules apply, we obtain the problem kernel. The analysis and proof of the kernel size make use of the non-existence of the structures (e.g., vertices of high degree) which the reduction rules addressed.

To convey some intuition about reduction rules let us consider three prevalent types of reduction rules, here we call them *mandatory rules*, *optimality rules*, and *equivalent replacement rules*. Note that, in contrast to fixed-parameter tractability results, strong kernelization results are mostly obtained for parameters relating to the solution size.

Mandatory rules

A mandatory reduction rule addresses structures in the problem instance where a certain decision is forced in order to obtain a solution of the desired maximum or minimum value. The classic example is that of VERTEX COVER: If a vertex has degree greater than k, then any vertex cover of size at most k must include it; it cannot include the more than k neighbors. Note how similar such a rule may be to a heuristic (e.g., "selecting high-degree vertices is good"), however, the presence of a bound on the solution size may turn heuristics into reduction rules (e.g., "vertices of degree larger than k must be selected").

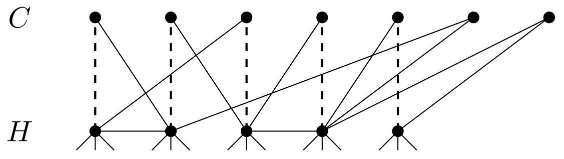

Figure 1.2.: Crown reduction for VERTEX COVER with head H and crown C. The dashed edges indicate the matching of the head into the crown. The reduction rule will select all vertices of H, implying that all vertices in $H \cup C$ are deleted and that the parameter is decreased by $|H|$.

Optimality rules

An optimality rule makes a decision on certain local structures, based on a proof that the locally optimal choice is also globally optimal. Such proofs usually work by replacement arguments: Given any solution to the complete instance, we can modify it at no additional cost (resp., without losing value) to make the same local choices. Again, there is a nice example for VERTEX COVER, the so-called *crown reduction*: Given an independent set (the "crown") and its neighborhood (the "head"), if there is a matching of the head into the crown, then it is optimal to select all vertices of the head (see Figure 1.2). The key observation is that each matching edge requires one of its vertices to be in the cover, and the vertices in the head can be chosen by a replacement argument. Note how the crown reduction generalizes the basic rule that it is optimal to select the neighbor of a degree-1 vertex.

Equivalent replacement

Equivalent replacement rules are part of many kernelizations, on different levels of difficulty. A simple but often vital reduction rule is that of deleting parts of the instance which do not contribute to the solution (i.e., they do not incur any cost or do not increase the value, respectively). Note that this is not always possible and that lower bounds for kernelizations use structures that may become important as a solution is constructed; hence they cannot be so easily reduced. As another example, many graph problems permit a simple treatment of vertices of degree two:

- For FEEDBACK VERTEX SET one may contract vertices of degree two (i.e., replace them by an edge between their neighbors). It is easy to see that this does not change the size of optimal feedback vertex sets.
- For VERTEX COVER there is the so-called *folding*: Given a vertex v of degree two with non-adjacent neighbors u and w, we obtain an equivalent instance by creating a new vertex x adjacent to all neighbors of u and w (except for v), deleting u, v, and w, and decreasing the parameter by one (see Figure 1.3). The argument is that there is an optimal solution which selects v or both u and w. This solution

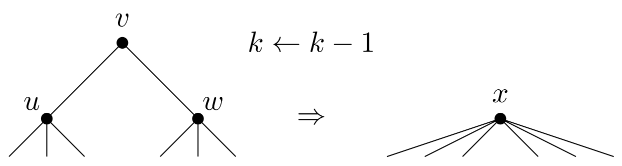

Figure 1.3.: Folding of a degree-2 vertex as a reduction rule for VERTEX COVER.

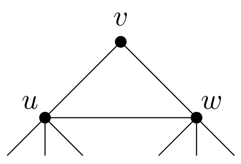

Figure 1.4.: When the two neighbors of a degree-2 vertex are adjacent then it is optimal to select them.

corresponds directly to not selecting respectively selecting x in the reduced instance and vice versa; in both cases the solution for the reduced instance is smaller by exactly one vertex.

Remark. Folding requires the two neighbors u, w of v to be non-adjacent. If u and w are adjacent, then it is optimal to select them. The reason is that u, v, and w induce a triangle, which requires two vertices to be selected; by replacement arguments u and w are an optimal choice (see Figure 1.4). This can be generalized to the case that the neighbors of some vertex (of arbitrary degree) form a clique, another example of an optimality rule.

More involved reduction rules of this style were for example used by Bodlaender et al. [17]: To obtain small kernels for problems on planar graphs they considered regions of the graph, i.e., induced subgraphs connected to the remaining graph by a set of boundary vertices. Then whole regions of the graph are replaced by smaller regions, having the same boundary vertices, such that the overall behavior does not change.

We conclude the chapter by reviewing the recent framework by Bodlaender et al. [16] for excluding polynomial kernelizations as well as work by Dell and van Melkebeek [44] on concrete polynomial lower bounds for kernelization.

1.4. Polynomial lower bounds for kernelization

Recently Bodlaender, Downey, Fellows, and Hermelin [16] provided the first polynomial lower bounds for the kernelizability of some parameterized problems as well as a framework for obtaining such results. They introduced the notions of *or-* respectively *and-composition algorithms* for parameterized problems and showed that such an algorithm together with a polynomial kernelization yields an *or-* respectively *and-distillation algorithm* for the unparameterized (classic) version of the problem. Based on the hypothesis that NP-complete problems do not admit either variant of distillation algorithm, this proves non-existence of polynomial kernelizations for some problems. Related work by Fortnow and Santhanam [56] showed that an or-distillation algorithm for any NP-complete problem would imply NP $\subseteq$ co-NP/poly, which is known to cause a collapse of the polynomial hierarchy to the third level [113]. It is a major open problem whether a similar consequence can be proven for and-distillation. In this thesis we refer only to the results for or-compositional problems and hence omit the prefix for readability.

Excluding polynomial kernelizations

Let $\mathcal{Q} \subseteq \Sigma^* \times \mathbb{N}$ be a parameterized problem. A *composition algorithm* for $\mathcal{Q}$ is an algorithm that on input of t instances $(I_1, k), \dots, (I_t, k) \in \Sigma^* \times \mathbb{N}$ uses time polynomial in $\sum_{i=1}^{t} |I_i| + k$ and outputs an instance (I', k') such that

- k' is bounded by a polynomial in k and
- $(I', k') \in \mathcal{Q}$ if and only if $(I_i, k) \in \mathcal{Q}$ for at least one $i \in \{1, \dots, t\}$.

We then say that $\mathcal{Q}$ is *compositional*.

Example 1.3. As a simple example let us consider the k-PATH problem; given a graph G and an integer k (the parameter) the task is to decide whether G contains a path on k vertices as a subgraph. The k-PATH problem is NP-complete and can be solved in randomized time $2^k \cdot n^{\mathcal{O}(1)}$ [111]. Given t instances $(G_1, k), \dots, (G_t, k)$ a composed instance can be obtained by taking the disjoint union of the graphs. Clearly, $G_1 \dot{\cup} \dots \dot{\cup} G_t$ contains a path on k vertices if and only if one of the graphs G_i contains such a path.

A composition algorithm constructs an instance with relatively small parameter value, which is equivalent to at least one of the given instances being a yes-instance. The idea is that, together with a polynomial kernelization, this allows a similar algorithm for the classic, unparameterized problem, a so-called *distillation algorithm.*

A *distillation algorithm* for a problem $L \subseteq \Sigma^*$ is an algorithm that given t instances $I_1, \dots, I_t \in \Sigma^*$ each of size n takes time polynomial in $t + n$ and outputs an instance I' such that

- the size of I' is bounded by a polynomial in n and
- $I' \in L$ if and only if $I_i \in L$ for at least one $i \in \{1, \dots, t\}$.

To obtain the kernel lower bound it is mandatory that the unparameterized version of the considered problem is NP-complete when the parameter is encoded in unary. Formally, the *unparameterized version* or *derived classical problem* $\tilde{\mathcal{Q}}$ of a parameterized problem $\mathcal{Q}$ is defined by $\tilde{\mathcal{Q}} = \{I\#1^k \mid (I, k) \in \mathcal{Q}\}$, where $\# \notin \Sigma$ is the blank letter and 1 is any letter from Σ.

Theorem 1.4 ([16, 56])**.** *Let $\mathcal{Q}$ be a compositional parameterized problem whose unparameterized version $\tilde{\mathcal{Q}}$ is* NP*-complete. Then $\mathcal{Q}$ does not admit a polynomial kernelization unless* NP $\subseteq$ co-NP/poly*.*

There exist a few results proving lower bounds via Theorem 1.4, e.g., [18, 46, 53, 80] (the work from [80] is presented in Chapter 4). At the end of this chapter we show an application of the framework to the so-called MULTIPLE COMPATIBLE PATTERNS problem. For many other problems, it seems hard to find a composition algorithm, even though the problems may have resisted attempts at polynomial kernelizations for a long time. In some of those cases it has turned out that a suitable notion of reduction may help to show lower bounds.

Transferring kernelization results via reductions

In [18] Bodlaender et al. proposed the use of polynomial-time reductions, where the new parameter value is polynomially bounded in the old one, as a tool to extend the lower bounds to further problems. These reductions are restrictions of the usual fpt-reductions, which allow fpt-time and a less constrained growth of the parameter; fpt-reductions are unable to distinguish problems inside FPT (observe that every non-trivial problem in FPT is complete for FPT under fpt-reductions). A similar notion was independently given by Harnik and Naor [67], so-called *W-reductions*, but their notion is related to the witness size of the instances.

A *polynomial time and parameter transformation* [18] from $\mathcal{Q}$ to $\mathcal{Q}'$ is a polynomial-time mapping $H : \Sigma^* \times \mathbb{N} \to \Sigma'^* \times \mathbb{N} : (x, k) \mapsto (x', k')$ such that

$$\forall (x, k) \in \Sigma^* \times \mathbb{N} : ((x, k) \in \mathcal{Q} \Leftrightarrow (x', k') \in \mathcal{Q}') \text{ and } k' \leq p(k),$$

for some polynomial p. We use the shorthand *polynomial parameter transformation.*

Theorem 1.5 ([18])**.** *Let $\mathcal{P}$ and $\mathcal{Q}$ be parameterized problems, and let $\tilde{\mathcal{P}}$ and $\tilde{\mathcal{Q}}$ be their derived classical problems. Suppose that $\tilde{\mathcal{P}}$ is* NP*-complete and $\tilde{\mathcal{Q}} \in$ NP. If there is a polynomial parameter transformation from $\mathcal{P}$ to $\mathcal{Q}$ and $\mathcal{Q}$ admits a polynomial kernelization, then $\mathcal{P}$ also admits a polynomial kernelization.*

Proof (sketch). The essential idea is that NP-completeness permits us to use the polynomial kernelization for $\mathcal{Q}$ also for $\mathcal{P}$: Given an instance (I, k) of $\mathcal{P}$ the polynomial parameter transformation computes an equivalent instance (I', k') for $\mathcal{Q}$. Using a polynomial kernelization on (I', k') creates an equivalent instance (I'', k'') of $\mathcal{Q}$, whose size is polynomial in k' and hence also in k. By NP-completeness of $\mathcal{P}$ we can map (I'', k'') to an equivalent instance of $\mathcal{P}$, with size polynomial in the size of (I'', k'') and hence polynomial in k. This procedure constitutes a polynomial kernelization for $\mathcal{P}$. □

Thus if $\mathcal{P}$ does not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly, then the same is true for $\mathcal{Q}$ if $\mathcal{P}$ reduces to $\mathcal{Q}$ by a polynomial parameter transformation (and if $\tilde{\text{P}}$ is NP-complete).

Collapse of the polynomial hierarchy

Let us briefly consider the consequence NP $\subseteq$ co-NP/poly, respectively, the assumption that NP $\nsubseteq$ co-NP/poly. The class co-NP/poly contains all languages which can be accepted by a co-NP machine which is provided an *advice string* of length *poly*(n) for each input length n (e.g., for all inputs x of length $|x| = n$ the same advice string is given). Note that NP $\nsubseteq$ co-NP/poly is a stronger assumption than NP $\neq$ co-NP which is stronger than P $\neq$ NP. If NP $\subseteq$ co-NP/poly then, according to a result by Yap [113], the polynomial hierarchy collapses to the third level.

Concrete lower bounds

Very recently Dell and van Melkebeek [44] refined the approach of Bodlaender et al. [16] to address also problems which do have polynomial kernelizations, proving that the known polynomial kernelizations for some problems are best possible with respect to the total size, e.g., for VERTEX COVER, FEEDBACK VERTEX SET, and d-HITTING SET. In fact, even stronger, their work excludes oracle communication protocols of cost lower than some polynomial.

In the used two-player oracle communication protocol the first player is polynomially bounded (i.e., modeled as a deterministic, polynomial-time Turing machine) and the second player (the oracle) is computationally unbounded, but only the first player has access to the input. The cost of the protocol is the number of bits sent from the first player to the oracle; the answers do not incur any cost. The number of queries is only bounded by the total number of bits to be used. At the end of the protocol the first player has to decide whether the given input is in the language. Note that any kernelization gives an oracle communication protocol of cost matching the total size of the kernel (simply sending the problem kernel to the oracle).

Theorem 1.6 ([44])**.** *Let $d \geq 3$ be an integer and let ϵ be a positive real. If* NP $\nsubseteq$ co-NP/poly*, there is no protocol of cost $\mathcal{O}(n^{d-\epsilon})$ to decide whether an n-variable d-CNF formula is satisfiable.*

Theorem 1.7 ([44])**.** *Let $d \geq 2$ be an integer and let ϵ be a positive real. If* NP $\nsubseteq$ co-NP/poly*, there is no protocol of cost $\mathcal{O}(n^{d-\epsilon})$ to decide whether a d-uniform hypergraph on n vertices has a vertex cover of at most k vertices.*

Dell and van Melkebeek concluded some strong implications for kernelizability: Since the number k of vertices to be selected is bounded by n, Theorem 1.7 shows that the known kernelizations for VERTEX COVER and d-HITTING SET are essentially optimal with respect to the total size, assuming NP $\nsubseteq$ co-NP/poly: $\mathcal{O}(k)$ vertices as well as $\mathcal{O}(k^2)$ edges for VERTEX COVER [33] and $\mathcal{O}(k^{d-1})$ vertices as well as $\mathcal{O}(k^d)$ edges for d-HITTING SET [1]. It is not possible to improve the polynomial degree of the number of edges in these kernelizations since, for example, $\mathcal{O}(k^{d-\epsilon})$ edges can be encoded in $\mathcal{O}(k^{d-\epsilon'})$ bits (a list of the edges requires only a factor of $\mathcal{O}(\log k)$ to identify the vertices). The same is true for FEEDBACK VERTEX SET via a simple reduction from VERTEX COVER, namely by replacing every edge with a triangle (using a new, unique vertex) and without changing the parameter value (this fact motivates a more general proof in [44]). Thus a cheaper kernelization (or protocol) for FEEDBACK VERTEX SET would give a cheaper protocol for VERTEX COVER. Thus the kernelization with $\mathcal{O}(k^2)$ vertices and $\mathcal{O}(k^2)$ edges for

FEEDBACK VERTEX SET [110] is essentially optimal. For FEEDBACK VERTEX SET and d-HITTING SET it may, however, be possible to improve the number of vertices.

The Multiple Compatible Patterns problem.

We conclude the section on lower bounds for kernelization by showcasing a lower bound proof by means of proving compositionality. We introduce the MULTIPLE COMPATIBLE PATTERNS (MCP) problem and show that it does not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly; MCP will be a source problem for other lower bounds in Chapters 7 and 8:

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.

The MCP problem is trivially fixed-parameter tractable (even with respect to parameter r): Simply try all 2^r possible strings from $\{0, 1\}^r$. To prove that it does not admit a polynomial kernelization (unless NP $\subseteq$ co-NP/poly) we will show that it is NP-complete and compositional. Theorem 1.4 will then imply the desired lower bound.

Lemma 1.8. MULTIPLE COMPATIBLE PATTERNS *admits no polynomial kernelization unless* NP $\subseteq$ co-NP/poly.

Proof. First, let us show that the derived classical problem is NP-complete. We will reduce from CLIQUE. Let (G, k) be an instance of CLIQUE; we will create an instance of MCP with pattern length n, which contains k compatible patterns if and only if G has a k-clique. Order the vertices of G from 1 to n. For every vertex v_i, create a pattern where bit i is 1, where bit j is 0 for every non-neighbor v_j of v_i, and where all other bits (i.e., representing the neighbors of v_i) are wildcards. Now two patterns are compatible if and only if the corresponding vertices are neighbors, and the MCP problem has k compatible patterns if and only if G has a k-clique. Clearly, encoding the parameter values $r = n$ and k in unary uses additional space linear in the instance size, not harming NP-completeness.

We will now show a composition algorithm for MCP. Let $(I_1, r_1 + k_1), \ldots, (I_t, r_t + k_t)$ be t input instances with the same parameter value $r + k = r_i + k_i$; assume by padding that both r and k are identical for all input instances. Clearly the problem can be solved in $2^r \cdot n^{\mathcal{O}(1)}$ time, so we assume that $\log t = \mathcal{O}(r)$, or else solve all input problems and produce a dummy instance for the output. Now add $\lceil \log t \rceil$ bits to the pattern length, and for every input instance i, let all its patterns contain the binary expansion of i in the added bits. Clearly this comprises a composition algorithm. Any string from $\{0, 1\}^{r + \lceil \log t \rceil}$ has a specific binary number in its last $\lceil \log t \rceil$ positions, and hence it cannot match strings from an instance with a different number. Thus a 0/1-string matching at least k patterns must match k patterns from the same instance. Thus MCP is compositional and, by Theorem 1.4, it does not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly. $\square$

2. Sunflowers

2.1. Introduction

For most of the polynomial kernelizations presented in this thesis, sunflowers play an important role as a starting point for the main reduction rules. Sunflowers are a notion from extremal set theory; they are configurations of sets which intersect in a highly uniform way. The *Sunflower Lemma*, discovered by Erdős and Rado [48] in 1960, states that any sufficiently large family of equally-sized sets must contain large sunflowers; this is often named one of the most beautiful results of extremal set theory.

Definition 2.1. A *sunflower (or Δ-system) of cardinality k and with core C* is a family of k sets $\{f_1, \dots, f_k\}$ such that $f_i \cap f_j = C$ for all $i \neq j$, i.e., the pairwise intersection of any two of the sets is exactly the core C. The pairwise disjoint sets $f_1 \setminus C, \dots, f_k \setminus C$ are called the *petals*; they are required to be non-empty.

Figure 2.1 shows a schematic representation of a sunflower. It is important to keep in mind that the core of a sunflower may be empty, i.e., a sunflower may consist of pairwise disjoint sets. The property of sunflowers which is central for kernelization is the fact that the petals $f_i \setminus C$ are pairwise disjoint.

We continue by presenting the Sunflower Lemma due to Erdős and Rado [48]. Let us recall that a hypergraph is d-uniform if all its edges have cardinality exactly d.

Lemma 2.2 (Sunflower Lemma [48])**.** *Let $k, d \in \mathbb{N}$ and let $\mathcal{H}$ be a d-uniform hypergraph with more than $d! \cdot k^d$ edges. Then there is a sunflower of cardinality $k+1$ in $\mathcal{H}$. For every fixed d there is an algorithm that computes such a sunflower in time polynomial in $|E(\mathcal{H})|$.*

Proof. The lemma clearly holds for any $\mathcal{H}$ and k when $d = 1$ since any $d+1$ different singleton sets must be pairwise disjoint.

We assume for contradiction that there are integers $d > 1$ and k as well as a d-uniform hypergraph with more than $d! \cdot k^d$ edges which does not contain a sunflower of cardinality $k+1$. We choose such a counter example with minimum value of d. Let x be any vertex and let $\mathcal{H}_x$ be the hypergraph with edge set

$$\{e \mid e \cup \{x\} \text{ is an edge of } \mathcal{H}\}.$$

Clearly any sunflower of $\mathcal{H}_x$ with some core C gives rise to a sunflower in $\mathcal{H}$ with core $C \cup \{x\}$ since there is an edge $e \cup \{x\}$ in $\mathcal{H}$ for each edge e of $\mathcal{H}_x$. Thus, by selection of our counter example, $\mathcal{H}_x$ has at most $(d-1)! \cdot k^{d-1}$ edges. Hence x occurs in at most $(d-1)! \cdot k^{d-1}$ edges of $\mathcal{H}$.

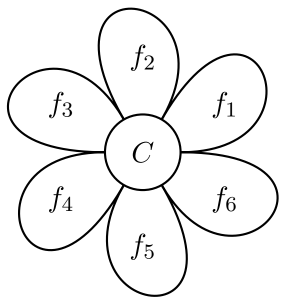

Figure 2.1.: A sunflower of cardinality six with core C.

Based on this fact, contradicting our assumption, we are able to find a sunflower in $\mathcal{H}$. This is achieved by greedily selecting a maximal set of pairwise disjoint edges of $\mathcal{H}$. Since each vertex occurs at most $(d-1)! \cdot k^{d-1}$ times, each edge can therefore intersect with at most $d \cdot ((d-1)! \cdot k^{d-1} - 1)$ other edges. Thus no selection of at most k edges can be maximal, since it intersects at most $d! \cdot k^d$ edges. Thus the greedy selection finds at least $k+1$ pairwise disjoint edges which constitute a sunflower with empty core in $\mathcal{H}$. $\square$

It is an open problem whether the bound of $d! \cdot k^d$ is best possible. It is straightforward to give a family of k^d sets which does not contain a sunflower of cardinality $k+1$ (see the following example), but that still leaves a gap of $d!$ between the two bounds. Erdős and Rado conjectured that the upper bound is at most C^d for every fixed k where C depends only on k. While there are some improvements to the upper bound, the conjecture is still open (see the nice book of Jukna [72] on extremal combinatorics).

Example 2.3. To illustrate the lower bound of k^d let us construct a family of k^d sets which does not contain a sunflower of cardinality $k+1$. Let us take $d = 4$; the construction will easily generalize to other values. We let

$$E(\mathcal{H}) = \{\{A_{i_1}, B_{i_2}, C_{i_3}, D_{i_4}\} \mid i_1, i_2, i_3, i_4 \in \{1, \dots, k\}\},$$

i.e., the family of all sets containing exactly one A, B, C, and D, each with some index from 1 to k. First, let us observe that there are k^4 such sets. Second, we observe that if the core of a sunflower contains an element, say A_i, then all sets contain A_i and no further A_j with $i \neq j$. Since the petals must be non-empty some given petal must at least contain one element, say D_i, implying that the core does not contain any D_j. Then however, each petal contains some D_j, but there are only k different indices, making $k+1$ pairwise disjoint petals impossible.

Implicit in the proof of the Sunflower Lemma there is an efficient algorithm to find a sunflower (see Algorithm 1). The Sunflower Lemma as well as the algorithm can be adapted to d-dimensional hypergraphs in a straightforward way (recall that a hypergraph is d-dimensional if each edge has cardinality *at most* d).

Algorithm 1 *FindSunflower*

Input: $(\mathcal{H}, d, k)$: A family $\mathcal{H}$ of sets of size d and an integer k.
Output: A sunflower in $\mathcal{H}$ of cardinality at least $k+1$, if $|\mathcal{H}| > d! \cdot k^d$, or report **false**.

```
    if $|\mathcal{H}| \leq d! \cdot k^d$ then
      return false
    end if
    if $d = 1$ then
 5:   return $\mathcal{H}$
    end if
    $\mathcal{F} \leftarrow$ a maximal selection of disjoint sets
    if $|\mathcal{F}| \geq k+1$ then
      return $\mathcal{F}$
10: end if
    Find an element $v$ of some set in $\mathcal{F}$ that occurs most frequently in $\mathcal{H}$
    $\mathcal{H}' \leftarrow \{e \mid e \cup \{v\} \in \mathcal{H}\}$
    $\mathcal{F}' \leftarrow FindSunflower(\mathcal{H}', d-1, k)$
    $\mathcal{F}'' \leftarrow \{e \cup \{v\} \mid e \in \mathcal{F}'\}$
15: return $\mathcal{F}''$
```

Corollary 2.4. *Let $k, d \in \mathbb{N}$ and let $\mathcal{H}$ be a hypergraph of dimension d and with more than $d \cdot d! \cdot k^d$ edges. Then there is a sunflower of cardinality $k+1$ in $\mathcal{H}$. For every fixed d there is an algorithm that computes such a sunflower in time polynomial in $|E(\mathcal{H})|$.*

Proof. For some $d' \in \{1, \ldots, d\}$, the hypergraph $\mathcal{H}$ has more than $d! \cdot k^d \geq d'! \cdot k^{d'}$ edges of cardinality d'. Let $\mathcal{H}_{d'}$ be the d'-uniform subgraph induced by the edges of cardinality d'. We apply the Sunflower Lemma on $\mathcal{H}_{d'}$ and obtain a sunflower $\mathcal{F}$ of cardinality $k+1$ in time polynomial in $|E(\mathcal{H}_{d'})| \leq |E(\mathcal{H})|$. Clearly $\mathcal{F}$ is also a sunflower of $\mathcal{H}$. $\square$

Sunflowers in sets of tuples

Sunflowers can also be defined as configurations of tuples (of the same arity), a variant that will be required for the kernelization results in Chapters 6 and 8. For tuples the positions of the elements matter and the same element may occur in several positions. We define sunflowers of tuples in such a way that the core is a collection of positions and the tuples must have the same element in each core position; the sets of elements of the remaining positions shall be disjoint.

Definition 2.5. A collection of k tuples $x_1, \ldots, x_k$ each of arity $d \in \mathbb{N}$ is a *sunflower* (*of tuples*) *with cardinality k and core $C \subseteq \{1, \ldots, d\}$* if

- $x_1(i) = \ldots = x_k(i)$ for all $i \in C$ (i.e., the tuples agree on the core positions) and
- $x_i(j) \neq x_{i'}(j')$ for all $j, j' \in \{1, \ldots, d\} \setminus C$ and $i \neq i'$ (i.e., no element occurs in the petal positions of more than one tuple).

The set C is also called the *core positions* and the remaining positions $P = \{1, \ldots, d\} \setminus C$ are the *petal positions.*

Example 2.6. As an example, the following sets constitute a sunflower of cardinality t and with core $C = \{1, \ldots, c\}$, if all y_{ij} and $y_{i'j'}$ are distinct when $i \neq i'$:

$$\begin{gathered}(x_1, \ldots, x_c, y_{11}, \ldots, y_{1p}) \\ (x_1, \ldots, x_c, y_{21}, \ldots, y_{2p}) \\ \vdots \\ (x_1, \ldots, x_c, y_{t1}, \ldots, y_{tp})\end{gathered}$$

Remark. There are different ways to extend the definition of sunflowers. We define this variant since it appears to be the least restrictive version, for which the kernelization will work. A similar but more restricted version was used by Marx [88], demanding the elements in the petal positions to occur only in one tuple. This variant would also work for our kernelizations, but the bound in the corresponding sunflower lemma would be higher. Another possibility would be to interpret tuples, say $(x_1, \ldots, x_d)$, as sets $\{(1, x_1), \ldots, (d, x_d)\}$ and to use the definition for sets, but then an x_i in different positions would not be the same element.

For sets of tuples $\mathcal{H} \subseteq \mathcal{U}^d$, we give an adaptation of the Sunflower Lemma. The proof is along the same lines as the original, only requiring an additional factor of $d!$ for picking the shared core positions. Same as the Sunflower Lemma, this immediately gives a polynomial-time algorithm for finding a sunflower of tuples.

Lemma 2.7. *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 $k + 1$.*

Proof. If $d = 1$, then a sunflower of size $k+1$ can be easily found, since any $k+1$ tuples of arity $d = 1$ form a sunflower with empty core. Now for induction, assume the lemma to be true for all $d' \leq d - 1$.

Let X contain $\{x_1, \ldots, x_d\}$ for each tuple $(x_1, \ldots, x_d) \in \mathcal{H}$, i.e., let X contain all subsets of $\mathcal{U}$ that correspond to tuples in $\mathcal{H}$. Select a maximal pairwise disjoint subset $F \subseteq X$. If $|F| \geq k + 1$ then its elements correspond to $k + 1$ tuples that share no element, i.e., a sunflower with empty core. Otherwise, if $|F| \leq k$, then all other sets of X have a non-empty intersection with some element of F. Since the sets correspond to the tuples of $\mathcal{H}$ there must be an element of some set in F, say x, that occurs in at least $|\mathcal{H}|/kd$ tuples of $\mathcal{H}$, as there are at most kd such elements. Therefore, there must be a position, $p \in \{1, \ldots, d\}$, such that x occurs in position p of at least $|\mathcal{H}|/kd^2 > k^{d-1}((d-1)!)^2$ tuples of $\mathcal{H}$.

Define $\mathcal{H}'$ by $\mathcal{H}' = \{(x_1, \ldots, x_{p-1}, x_{p+1}, \ldots, x_d) \mid (x_1, \ldots, x_{p-1}, x, x_{p+1}, \ldots, x_d) \in \mathcal{H}\}$. Observe that $\mathcal{H}' \subseteq U^{d-1}$ and $|\mathcal{H}'| > k^{d-1}((d-1)!)^2$, implying that a sunflower of cardinality $k + 1$ can be found in $\mathcal{H}'$; immediately giving a sunflower in $\mathcal{H}$. □

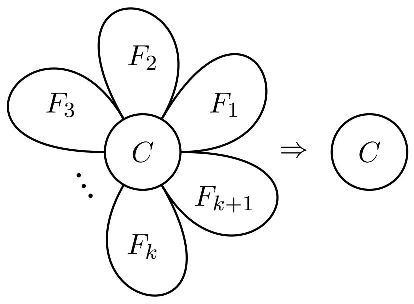

Figure 2.2.: Replacing a sunflower of cardinality $k+1$ by its core is a safe reduction rule for d-HITTING SET, i.e., when searching for a hitting set of size at most k.

2.2. Application in kernelization

The archetypical application of sunflowers in kernelization is probably the polynomial kernelization for d-HITTING SET (see [54]). The d-HITTING SET problem is defined as follows:

> ***d*-HITTING SET**
>
> **Input:** A hypergraph $\mathcal{H} = (V, E)$ of dimension d and an integer k.
>
> **Parameter:** k.
>
> **Task:** Decide whether $\mathcal{H}$ has a hitting set of size at most k, i.e., a subset of at most k vertices that has a nonempty intersection with every edge of $\mathcal{H}$.

Let $(\mathcal{H}, k)$ be an instance of d-HITTING SET and let us consider a sunflower consisting of $k+1$ edges $F_1, \dots, F_{k+1}$ of $\mathcal{H}$, e.g.:

$$\begin{aligned} F_1 &= \{u_1, \dots, u_c, \quad v_{1,1}, \dots, v_{1,p}\} \\ F_2 &= \{u_1, \dots, u_c, \quad v_{2,1}, \dots, v_{2,p}\} \\ &\vdots \\ F_{k+1} &= \{u_1, \dots, u_c, \quad v_{k+1,1}, \dots, v_{k+1,p}\} \end{aligned}$$

By the *pigeon-hole-principle* we get that in any hitting set of at most k vertices there must be a vertex that is contained in two of the sets. The pairwise intersection of any two edges is the core of the sunflower, here $\{u_1, \dots, u_c\}$. Thus any hitting set intersects the core of any sunflower of cardinality at least $k+1$; we may therefore add the core as an additional edge without changing the outcome (see Figure 2.2). Now, however, we may also discard all edges of the sunflower, since intersecting the core is a stricter requirement. This already constitutes a kernelization that leaves at most $d \cdot d! \cdot k^d \in \mathcal{O}(k^d)$ edges. Given some instance $(\mathcal{H}, k)$ of d-HITTING SET, while $\mathcal{H}$ contains more than $d \cdot d! \cdot k^d$ edges, apply the following steps:

1. In polynomial time search for a sunflower of at least $k+1$ edges of $\mathcal{H}$ according to Corollary 2.4.

2. If the core of the sunflower is empty, then it consists of $k+1$ pairwise disjoint edges and we may reject the instance (as there cannot be a hitting set with k elements).
3. Otherwise, replace the edges that form the sunflower by the core. Repeat.

The total runtime is polynomial in the size of $\mathcal{H}$ since each iteration reduces the number of edges by at least k. Afterwards there are at most $\mathcal{O}(k^d)$ edges and at most $\mathcal{O}(k^d)$ vertices, since the edges have cardinality at most d.

Remark. There is a slightly stronger kernelization for d-HITTING SET due to Abu-Khzam [1], which uses crown reductions. It achieves $\mathcal{O}(k^{d-1})$ vertices but has the same bound of $\mathcal{O}(k^d)$ on the number of edges.

3. Polynomial kernelizations for MIN $F^+\Pi_1$ and MAX NP

3.1. Introduction

In this chapter we consider the kernelization properties of the syntactically defined complexity classes MIN $F^+\Pi_1$, MAX NP, and MAX SNP. We show that the standard parameterization of any problem from these classes admits a polynomial kernelization. We also extend these kernelizations to weighted versions of the respective problems. Recall that the standard parameterization of some maximization or minimization problem is the decision version, i.e., deciding whether the optimum value is at least k or at most k, respectively, with parameter k.

Two decades ago Papadimitriou and Yannakakis [99] initiated the syntactic study of optimization problems to extend the understanding of approximability. They introduced the classes MAX NP and MAX SNP as natural maximization variants of NP, based on Fagin's [49] syntactic characterization of NP. Essentially problems are in MAX NP or MAX SNP if their optimum value can be expressed as the maximum number of tuples for which some existential, respectively quantifier-free, first-order formula holds. They showed that every problem in these two classes is approximable to within a constant factor of the optimum. Arora et al. [7] complemented this by proving that no MAX SNP-hard problem has a polynomial-time approximation scheme (PTAS), unless P = NP. Contained in MAX SNP there are some well-known maximization problems, such as MAX CUT, MAX q-SAT, and INDEPENDENT SET on graphs of bounded degree. In its superclass MAX NP we also find, amongst others, the MAX SAT problem (with clauses of unbounded arity).

In follow-up work to [99], Panconesi and Ranjan [97] further investigated the relation between syntactic properties and approximability. They introduced classes MAX Π_1 and a subclass RMAX, motivated by unconditional proofs showing that some important optimization problems, e.g., CLIQUE and MAXIMUM MATCHING (which is in P), are not contained in MAX NP. We point out that if MAX NP *would* contain all problems from P then the fact that CLIQUE is not in MAX NP would (unconditionally) prove that P $\neq$ NP. The class MAX Π_1 contains all problems whose optimum value can be expressed as the maximum number of tuples for which some universally quantified first-order formula holds. However, MAX Π_1 (same as RMAX) already contains the CLIQUE problem, which was later showed to be hard to approximate; the strongest bound, NP-hardness to approximate within a factor of $\mathcal{O}(n^{1-\epsilon})$, was given by Håstad [68]. Similarly, since CLIQUE is complete for W[1], there is no hope for obtaining a general (polynomial) kernelization result for RMAX or MAX Π_1.

Kolaitis and Thakur [76, 77] generalized the approach of examining the logical definability of optimization problems and defined further classes of minimization and maximization problems. Amongst others they introduced the class MIN $F^+\Pi_1$ of problems whose optimum can be expressed as the minimum weight of an assignment (i.e., number of ones) that satisfies a certain restricted universal first-order formula. They proved that every problem in MIN $F^+\Pi_1$ is approximable to within a constant factor of the optimum. In MIN $F^+\Pi_1$ there are problems like VERTEX COVER, d-HITTING SET, and TRIANGLE EDGE DELETION. (For completeness of the picture, let us add that the class RMAX is called MAX $F^-\Pi_1$ in [76, 77], as it is the problem of finding a maximum weight satisfying assignment to some restricted universal first-order formula.)

Related work It was showed by Cai and Chen [28] that the standard parameterizations of all problems in MIN $F^+\Pi_1$ and MAX SNP are fixed-parameter tractable. Mahajan, Raman, and Sikdar [87] showed a different parameterization of MAX SNP problems to be fixed-parameter tractable, namely whether one can improve by at least k upon a guaranteed lower bound; for the lower bound they use the one proved by Papadimitriou and Yannakakis [99] which implied that problems in MAX SNP are constant-factor approximable (for a similar proof see Lemma 3.17). Earlier work of Mahajan and Raman [86] addressed MAX CUT and MAX SAT parameterized above lower bound.

Khanna et al. [74] proved that APX, the class of constant-factor approximable problems, is the closure of MAX SNP under PTAS-preserving reductions. The same is showed to be true for the subclass APX-PB of polynomially-bounded APX problems with respect to so-called E-reductions. Thus for every constant-factor approximable problem there is a PTAS-preserving reduction to a MAX SNP problem (in fact to MAX 3-SAT). Arora, Karger, and Karpinski [6] showed that dense instances of MAX SNP problems admit polynomial-time approximation schemes, later improved to efficient PTAS by Frieze and Kannan [57]; see also Karpinski's survey [73]. Furthermore, a large number of papers adopt the notion of MAX SNP-hardness as a tool to prove non-existence of polynomial-time approximation schemes.

Our work We prove that the standard parameterizations of problems in MIN $F^+\Pi_1$ and MAX NP admit polynomial kernelizations. For MAX SNP (a subclass of MAX NP) we show polynomial kernelizations with a linear base set (usually called linear kernelizations even though the total size may be larger). All kernelizations are then extended to weighted versions of the problems. We improve upon the above mentioned results of Cai and Chen [28] since fixed-parameter tractability only implies some (possibly exponential) kernelization. Most of the content in this chapter has been published in [78]; the extensions to the weighted problems are unpublished work with Bart Jansen.

Structure of this chapter Section 3.2 introduces formally the considered syntactical problem classes. In the subsequent Sections 3.3, 3.4, and 3.5 we prove the claimed kernelization results for MIN $F^+\Pi_1$, MAX NP, and MAX SNP, respectively. We conclude in Section 3.6 with a short summary.

3.2. Syntactical problems

Logic and finite structures

A *(relational) vocabulary* is a set σ of relation symbols, each having some fixed integer as its arity. *Atomic formulas over* σ are of the form $R(z_1, \ldots, z_t)$ where R is a t-ary relation symbol from σ and the z_i are variables. A *literal* is an atomic formula or the negation of an atomic formula. The set of *quantifier-free (relational) formulas over* σ is the closure of the set of all atomic formulas under negation, conjunction, and disjunction. A formula in *conjunctive normal form* (CNF) is a conjunction of disjunctions of literals, called *clauses*. A formula in *disjunctive normal form* (DNF) is a disjunction of conjunctions of literals, called *disjuncts*.

Definition 3.1 (Finite structure)**.** A tuple $\mathcal{A} = (A, R_1, \ldots, R_t)$ where A is a finite set and each R_i is an r_i-ary relation over A is called a *finite structure of type* $(r_1, \ldots, r_t)$.

MAX SNP

Definition 3.2 (MAX SNP)**.** An optimization problem $\mathcal{Q}$ on finite structures $\mathcal{A} = (A, R_1, \ldots, R_t)$ of type $(r_1, \ldots, r_t)$ is contained in MAX SNP if its optimum value can be expressed as

$$\text{opt}_{\mathcal{Q}}(\mathcal{A}) = \max_{\mathcal{S}} \left|\{\mathbf{x} \in A^{c_x} : (\mathcal{A}, \mathcal{S}) \models \psi(\mathbf{x}, \mathcal{A}, \mathcal{S})\}\right|,$$

where $\mathcal{S} = (S_1, \ldots, S_u)$ is a tuple of relation symbols of arities $s_1, \ldots, s_u$ and $\psi(\mathbf{x}, \mathcal{A}, \mathcal{S})$ is a quantifier-free formula in disjunctive normal form on variables $\{x_1, \ldots, x_{c_x}\}$ over the vocabulary $\{R_1, \ldots, R_t, S_1, \ldots, S_u\}$.

An optimization problem $\mathcal{Q}$ on inputs $(\mathcal{A}, w)$, consisting of a finite structure $\mathcal{A} = (A, R_1, \ldots, R_t)$ of type $(r_1, \ldots, r_t)$ and a weight function $w : A^{c_x} \to \mathbb{Q}_{\geq 0}$, is contained in WEIGHTED MAX SNP if its optimum value can be expressed as:

$$\text{opt}_{\mathcal{Q}}(\mathcal{A}, w) = \max_{\mathcal{S}} \sum \{w(\mathbf{x}) : \mathbf{x} \in A^{c_x} \wedge (\mathcal{A}, \mathcal{S}) \models \psi(\mathbf{x}, \mathcal{A}, \mathcal{S})\},$$

under the same restrictions as for MAX SNP. Here, the optimum value is the sum of $w(\mathbf{x})$ over all tuples $\mathbf{x}$ for which $\psi(\mathbf{x}, \mathcal{A}, \mathcal{S})$ holds, maximized over all choices of $\mathcal{S}$.

Example 3.3 (MAX CUT)**.** Let $G = (V, E)$ be a finite structure of type (2) that represents a graph by a set V of vertices and a symmetric binary relation E over V as its edges. The optimum of MAX CUT on structures G can be expressed as:

$$\max_{S \subseteq V} \left|\left\{(u, v) \in V^2 : (G, S) \models (E(u, v) \wedge S(u) \wedge \neg S(v))\right\}\right|.$$

The intention behind this formulation is that $S \subseteq V$ represents a partition of the vertices into S and $V \setminus S$. The expression maximizes the number of edges with one endpoint in S and the other in $V \setminus S$. Thus MAX CUT can be expressed as a MAX SNP problem.

MAX NP

Definition 3.4 (MAX NP)**.** An optimization problem $\mathcal{Q}$ on finite structures $\mathcal{A} = (A, R_1, \ldots, R_t)$ of type $(r_1, \ldots, r_t)$ is contained in MAX NP if its optimum value can be expressed as

$$\mathrm{opt}_{\mathcal{Q}}(\mathcal{A}) = \max_{\mathcal{S}} |\{\mathbf{x} \in A^{c_x} : (\mathcal{A}, \mathcal{S}) \models (\exists \mathbf{y} \in A^{c_y}) : \psi(\mathbf{x}, \mathbf{y}, \mathcal{A}, \mathcal{S})\}| \,,$$

where $\mathcal{S} = (S_1, \ldots, S_u)$ is a tuple of relation symbols of arities $s_1, \ldots, s_u$ and $\psi(\mathbf{x}, \mathbf{y}, \mathcal{A}, \mathcal{S})$ is a quantifier-free DNF formula on variables $\{x_1, \ldots, x_{c_x}, y_1, \ldots, y_{c_y}\}$ over the vocabulary $\{R_1, \ldots, R_t, S_1, \ldots, S_u\}$.

An optimization problem $\mathcal{Q}$ on inputs $(\mathcal{A}, w)$, consisting of a finite structure $\mathcal{A} = (A, R_1, \ldots, R_t)$ of type $(r_1, \ldots, r_t)$ and a weight function $w : A^{c_x} \to \mathbb{Q}_{\geq 0}$, is contained in WEIGHTED MAX NP if its optimum value can be expressed as:

$$\mathrm{opt}_{\mathcal{Q}}(\mathcal{A}, w) = \max_{\mathcal{S}} \sum \{w(\mathbf{x}) : \mathbf{x} \in A^{c_x} \wedge (\mathcal{A}, \mathcal{S}) \models (\exists \mathbf{y} \in A^{c_y}) : \psi(\mathbf{x}, \mathbf{y}, \mathcal{A}, \mathcal{S})\} \,,$$

under the same restrictions as for MAX NP. Similarly as for MAX SNP, the optimum value is the sum of $w(\mathbf{x})$ over all tuples $\mathbf{x}$ for which $(\exists \mathbf{y}) : \psi(\mathbf{x}, \mathbf{y}, \mathcal{A}, \mathcal{S})$ holds, maximized over all choices of $\mathcal{S}$.

Example 3.5 (MAX SAT)**.** Formulas in conjunctive normal form can be represented by finite structures $\mathcal{F} = (F, P, N)$ of type $(2, 2)$: Let F be the set of all clauses and variables, and let P and N be binary relations over F. Let $P(x, c)$ be true if and only if x is a literal of the clause c and let $N(x, c)$ be true if and only if $\neg x$ is a literal of the clause c. The optimum of MAX SAT on structures $\mathcal{F}$ can be expressed as:

$$\max_{T \subseteq F} |\{c \in F : (\mathcal{F}, T) \models (\exists x \in F) : (P(x, c) \wedge T(x)) \vee (N(x, c) \wedge \neg T(x))\}|.$$

Intuitively, $T \subseteq F$ is meant to encode a truth assignment to the variables ($P(x, c)$ and $N(x, c)$ ensure that x is a variable and c is a clause). The formula simply expresses that a clause is satisfied if it contains a satisfied literal. Thus MAX SAT can be expressed as a MAX NP problem.

MIN $F^+\Pi_1$

Definition 3.6 (MIN $F^+\Pi_1$)**.** An optimization problem $\mathcal{Q}$ on finite structures $\mathcal{A} = (A, R_1, \ldots, R_t)$ of type $(r_1, \ldots, r_t)$ is contained in MIN $F^+\Pi_1$ if its optimum value can be expressed as

$$\mathrm{opt}_{\mathcal{Q}}(\mathcal{A}) = \min_{S} \{|S| : (\mathcal{A}, S) \models (\forall \mathbf{x} \in A^{c_x}) : \psi(\mathbf{x}, \mathcal{A}, S)\},$$

where S is a single relation symbol of arity s and $\psi(\mathbf{x}, \mathcal{A}, S)$ is a quantifier-free formula in conjunctive normal form on variables $\{x_1, \ldots, x_{c_x}\}$ over the vocabulary $\{R_1, \ldots, R_t, S\}$. Furthermore, $\psi(\mathbf{x}, \mathcal{A}, S)$ is positive in S, i.e., S does not occur negated in $\psi(\mathbf{x}, \mathcal{A}, S)$.

An optimization problem $\mathcal{Q}$ on inputs $(\mathcal{A}, w)$, consisting of a finite structure $\mathcal{A} = (A, R_1, \ldots, R_t)$ of type $(r_1, \ldots, r_t)$ and a weight function $w : A^s \to \mathbb{Q}_{\geq 0}$, is contained in WEIGHTED MIN $\mathrm{F}^+\Pi_1$ if its optimum value can be expressed as:

$$\mathrm{opt}_{\mathcal{Q}}(\mathcal{A}, w) = \min_{S} \{w(S) : (\mathcal{A}, S) \models (\forall \mathbf{x} \in A^{c_x}) : \psi(\mathbf{x}, \mathcal{A}, S)\},$$

under the same restrictions as for MIN $\mathrm{F}^+\Pi_1$; $w(S)$ is the sum of $w(\mathbf{z})$ over all $\mathbf{z} \in S$.

Example 3.7 (VERTEX COVER)**.** Again let $G = (V, E)$ be a finite structure of type (2) that represents a graph by a set V of vertices and a symmetric binary relation E over V as its edges. The optimum of VERTEX COVER on structures G can be expressed as:

$$\min_{S \subseteq V} \{|S| : (G, S) \models (\forall (u, v) \in V^2) : (\neg E(u, v) \lor S(u) \lor S(v))\}.$$

Here $S \subseteq V$ is meant to encode the vertex cover. The universally quantified formula forces $S(u)$ or $S(v)$ to be true for each edge $\{u, v\}$. Thus VERTEX COVER can be expressed as a MIN $\mathrm{F}^+\Pi_1$ problem.

For a detailed introduction to MIN $\mathrm{F}^+\Pi_1$ as well as MAX NP and MAX SNP we refer the reader to [76, 77] and [99], respectively. An introduction to logic and complexity can be found in [98].

3.3. Polynomial kernelization for MIN $\mathrm{F}^+\Pi_1$

In this section we show that the standard parameterization of any problem in MIN $\mathrm{F}^+\Pi_1$ admits a polynomial kernelization. We will see that such problems can be easily reduced to d-HITTING SET for which polynomial kernelizations are known. The difficulty lies in finding a kernelization whose result can then be transferred to our original problem.

Let us fix some optimization problem $\mathcal{Q}$ from MIN $\mathrm{F}^+\Pi_1$ on finite structures of type $(r_1, \ldots, r_t)$. Accordingly let $R_1, \ldots, R_t$ be relation symbols of arities $r_1, \ldots, r_t$. Since $\mathcal{Q} \in$ MIN $\mathrm{F}^+\Pi_1$ there is an s-ary relation symbol S and a quantifier-free formula $\psi(\mathbf{x}, \mathcal{A}, S)$ in conjunctive normal form such that:

- the formula $\psi(\mathbf{x}, \mathcal{A}, S)$ is positive in S, i.e., there are no literals $\neg S(x_1, \ldots, x_s)$ and
- the optimum value of $\mathcal{Q}$ on input $\mathcal{A}$ of type $(r_1, \ldots, r_t)$ can be expressed as

$$\min_{S \subseteq A^s} \{|S| : (\mathcal{A}, S) \models (\forall \mathbf{x} \in A^{c_x}) : \psi(\mathbf{x}, \mathcal{A}, S)\}.$$

We denote by d the maximum number of occurrences of S in any clause of $\psi(\mathbf{x}, \mathcal{A}, S)$. This value plays a crucial role in our kernelization bound. For the polynomial kernelization we consider the standard parameterization of $\mathcal{Q}$, denoted by p-$\mathcal{Q}$:

Standard parameterization of $\mathcal{Q}$

Input: A finite structure $\mathcal{A}$ of type $(r_1, \ldots, r_t)$ and an integer k.

Parameter: k.

Task: Decide whether $\mathrm{opt}_{\mathcal{Q}}(\mathcal{A}) \leq k$.

We will see that, given an instance $(\mathcal{A}, k)$, deciding whether $\mathrm{opt}_{\mathcal{Q}}(\mathcal{A}) \leq k$ is equivalent to deciding an instance of d-HITTING SET; let us recall its definition:

> ***d*-HITTING SET**
>
> **Input:** A hypergraph $\mathcal{H} = (V, E)$ of dimension d and an integer k.
>
> **Parameter:** k.
>
> **Task:** Decide whether $\mathcal{H}$ has a hitting set of size at most k, i.e., a subset of at most k vertices that has a nonempty intersection with every edge of $\mathcal{H}$.

Given some instance $(\mathcal{A}, k)$ of p-$\mathcal{Q}$, with $\mathcal{A} = (A, R_1, \ldots, R_t)$, a natural first step is to consider the impact of the given relations R_i. For any fixed tuple $\mathbf{x} \in A^{c_x}$ we can evaluate all occurrences of the relation symbols R_i in $\psi(\mathbf{x}, \mathcal{A}, S)$ (recall that each R_i is given as a concrete subset of A^{r_i}). For each tuple $\mathbf{x}$ this will yield a formula containing only literals $S(\cdot)$. The following definition formalizes this procedure.

Definition 3.8. Let $\mathcal{A} = (A, R_1, \ldots, R_t)$ be a finite structure of type $(r_1, \ldots, r_t)$ and let $\mathbf{x} \in A^{c_x}$. We define $\psi_{\mathbf{x}}(\mathcal{A}, S)$ to be the formula obtained in the following way:

1. Replace all variables $x_1, \ldots, x_{c_x}$ by the chosen elements of A.
2. Replace all literals $R_i(\mathbf{z})$ and $\neg R_i(\mathbf{z})$ by 1 (true) or 0 (false) depending on whether $\mathbf{z}$ is contained in R_i (note that $\mathbf{z}$ is a concrete tuple from A^{r_i} by Step 1).
3. Delete all clauses that contain a 1 and delete all occurrences of 0.

We observe that the application of Definition 3.8 yields an equivalent formula in the sense that $(\mathcal{A}, S) \models \psi(\mathbf{x}, \mathcal{A}, S)$ if and only if $(\mathcal{A}, S) \models \psi_{\mathbf{x}}(\mathcal{A}, S)$: We only replace literals according to the input. It is easy to see that $\psi_{\mathbf{x}}(\mathcal{A}, S)$ is a formula in conjunctive normal form on literals $S(\mathbf{z})$ for some $\mathbf{z} \in A^s$ and with at most d literals per clause.

A formula $\psi_{\mathbf{x}}(\mathcal{A}, S)$ can have empty clauses when all literals $R_i(\cdot)$, $\neg R_i(\cdot)$ in a clause are evaluated to 0 and there are no literals $S(\cdot)$. In that case, no assignment to S can satisfy the formula $\psi_{\mathbf{x}}(\mathcal{A}, S)$, or equivalently $\psi(\mathbf{x}, \mathcal{A}, S)$. Thus $(\mathcal{A}, k)$ is a no-instance and we may reject it or return a dummy no-instance of constant size. Note that clauses of $\psi_{\mathbf{x}}(\mathcal{A}, S)$ cannot contain contradicting literals since $\psi(\mathbf{x}, \mathcal{A}, S)$ is positive in S. Henceforth we assume all clauses of formulas $\psi_{\mathbf{x}}(\mathcal{A}, S)$ to be nonempty.

We continue by defining a mapping Φ from finite structures $\mathcal{A}$ to hypergraphs $\mathcal{H}$. Then we show that $(\mathcal{A}, k)$ is a yes-instance for p-$\mathcal{Q}$ if and only if $(\Phi(\mathcal{A}), k)$ is a yes-instance for d-HITTING SET.

Definition 3.9. Let $\mathcal{A}$ be an instance of $\mathcal{Q}$. We define $\Phi(\mathcal{A}) := \mathcal{H}$ with $\mathcal{H} = (V, E)$. We let E be the family of all sets $e = \{\mathbf{z}_1, \ldots, \mathbf{z}_p\}$ such that $(S(\mathbf{z}_1) \vee \cdots \vee S(\mathbf{z}_p))$ is a clause of a $\psi_{\mathbf{x}}(\mathcal{A}, S)$ for some $\mathbf{x} \in A^{c_x}$. We let V be the union of all sets $e \in E$.

The hypergraphs $\mathcal{H}$ obtained from Φ have dimension d since each $\psi_{\mathbf{x}}(\mathcal{A}, S)$ has at most d literals per clause. The following lemma establishes the equivalence of $(\mathcal{A}, k)$ and $(\mathcal{H}, k) = (\Phi(\mathcal{A}), k)$.

Lemma 3.10. *Let $\mathcal{A} = (A, R_1, \dots, R_t)$ be a finite structure of type $(r_1, \dots, r_t)$ and let k be an integer. Then $(\mathcal{A}, k)$ is a yes-instance of p-$\mathcal{Q}$ if and only if $(\Phi(\mathcal{A}), k)$ is a yes-instance of d-*HITTING SET.

Proof. It suffices to show that for all $S \subseteq A^s$:

$$(\mathcal{A}, S) \models (\forall \mathbf{x} \in A^{c_x}) : \psi(\mathbf{x}, \mathcal{A}, S) \text{ if and only if } S \text{ is a hitting set for } \Phi(\mathcal{A}).$$

Let $\mathcal{H} = \Phi(\mathcal{A}) = (V, E)$ and let $S \subseteq A^s$:

$$\begin{aligned} & (\mathcal{A}, S) \models (\forall \mathbf{x} \in A^{c_x}) : \psi(\mathbf{x}, \mathcal{A}, S) \\ \Leftrightarrow\ & (\mathcal{A}, S) \models (\forall \mathbf{x} \in A^{c_x}) : \psi_{\mathbf{x}}(\mathcal{A}, S) \\ \Leftrightarrow\ & (\forall \mathbf{x} \in A^{c_x}) : \text{each clause of } \psi_{\mathbf{x}}(\mathcal{A}, S) \text{ has a literal } S(\mathbf{z}) \text{ for which } \mathbf{z} \in S \\ \Leftrightarrow\ & S \text{ has a nonempty intersection with every set } e \in E \\ \Leftrightarrow\ & S \text{ is a hitting set for } \mathcal{H} = (V, E). \end{aligned}$$

Since the number of ones in the assignment to S, i.e., the number of tuples $\mathbf{z} \in A^s$ with $S(\mathbf{z}) = 1$, translates directly to the cardinality of the hitting set and vice versa, the lemma follows. □

Using the connection to d-HITTING SET, our kernelization for p-$\mathcal{Q}$ will consist of the following three steps:

1. Map the given instance $(\mathcal{A}, k)$ for p-$\mathcal{Q}$ to an equivalent instance $(\mathcal{H}, k) = (\Phi(\mathcal{A}), k)$ for d-HITTING SET according to Definition 3.9 and Lemma 3.10.

2. Use a polynomial kernelization for d-HITTING SET on $(\mathcal{H}, k)$ to obtain an equivalent instance $(\mathcal{H}', k)$ with size polynomial in k.

3. Use $(\mathcal{H}', k)$ to derive an equivalent instance $(\mathcal{A}', k)$ of p-$\mathcal{Q}$. That way we will be able to conclude that $(\mathcal{A}', k)$ is equivalent to $(\mathcal{H}, k)$ and hence also to $(\mathcal{A}, k)$.

There are two kernelizations for d-HITTING SET: one by Flum and Grohe [54] based on the Sunflower Lemma and a recent one by Abu-Khzam [1] based on crown decompositions (see Section 2.2 for a presentation of the former). For our purpose of deriving an equivalent instance for p-$\mathcal{Q}$, these kernelizations have the drawback of shrinking sets during the reduction, since we need to find an equivalent instance of p-$\mathcal{Q}$ afterwards. To shrink edges we would need to shrink clauses of the formula $\psi(\mathbf{x}, \mathcal{A}, S)$, but we may only change the instance $(\mathcal{A}, k)$. Fortunately we are able to modify Flum and Grohe's kernelization to use only edge deletions.

Recall that the sunflower-based kernelization for d-HITTING SET uses the fact that a sunflower of cardinality greater than k forces an element of its core to be selected, since the petals are pairwise disjoint. Thus such a sunflower may be replaced by its core. In our case the idea is to shrink sunflowers from size at least $k+2$ down to size $k+1$. This way the selection of an element from the core is still forced, but we are able to reduce the size of our instance without shrinking of edges. Based on this idea we obtain the following theorem.

Theorem 3.11. *There exists a polynomial kernelization of* d-HITTING SET *that, given an instance* $(\mathcal{H}, k)$*, computes an equivalent instance* $(\mathcal{H}^*, k)$ *such that* $E(\mathcal{H}^*) \subseteq E(\mathcal{H})$ *and the graph* $\mathcal{H}^*$ *has* $\mathcal{O}(k^d)$ *vertices and* $\mathcal{O}(k^d)$ *edges.*

Proof. Let $(\mathcal{H}, k)$ be an instance of d-HITTING SET, with $\mathcal{H} = (V, E)$. If $\mathcal{H}$ contains a sunflower $F = \{f_1, \dots, f_{k+1}\}$ of cardinality $k+1$ then every hitting set of $\mathcal{H}$ of size at most k must have a nonempty intersection with the core C of F or with each of the $k+1$ disjoint sets $f_1 \setminus C, \dots, f_{k+1} \setminus C$. Thus every hitting set of at most k elements must have a nonempty intersection with C.

Now consider a sunflower $F = \{f_1, \dots, f_{k+1}, f_{k+2}\}$ of cardinality $k+2$ in $\mathcal{H}$ and let $\mathcal{H}' = (V, E \setminus \{f_{k+2}\})$. We show that the instances $(\mathcal{H}, k)$ and $(\mathcal{H}', k)$ are equivalent. Clearly every hitting set for $\mathcal{H}$ is also a hitting set for $\mathcal{H}'$ since $E(\mathcal{H}') \subseteq E(\mathcal{H})$. Let $S \subseteq V$ be a hitting set of size at most k for $\mathcal{H}'$. Since $F \setminus \{f_{k+2}\}$ is a sunflower of cardinality $k+1$ in $\mathcal{H}'$, it follows that S has a nonempty intersection with its core C. Hence S has a nonempty intersection with $f_{k+2} \supseteq C$ too. Thus S is a hitting set of size at most k for $\mathcal{H}$, implying that $(\mathcal{H}, k)$ and $(\mathcal{H}', k)$ are equivalent.

We turn this fact into a kernelization, by starting with $\mathcal{H}^* = \mathcal{H}$ and by repeating the following step while $\mathcal{H}^*$ has more than $(k+1)^d \cdot d! \cdot d$ edges. By Corollary 2.4 we can find a sunflower of cardinality $k+2$ in $\mathcal{H}^*$ in time polynomial in $|E(\mathcal{H}^*)|$. We delete an edge of the detected sunflower from the edge set of $\mathcal{H}^*$, thereby reducing the cardinality of the sunflower to $k+1$. Thus, by the argument from the previous paragraph, we maintain that $(\mathcal{H}, k)$ and $(\mathcal{H}^*, k)$ are equivalent.

Afterwards $E(\mathcal{H}^*) \subseteq E(\mathcal{H})$ and $\mathcal{H}^*$ has no more than $(k+1)^d \cdot d! \cdot d \in \mathcal{O}(k^d)$ edges. Since we delete an edge of $\mathcal{H}^*$ in each step, there are $\mathcal{O}(|E(\mathcal{H})|)$ steps, and the total time is polynomial in $|E(\mathcal{H})|$. Deleting all isolated vertices from $\mathcal{H}^*$ yields a total number of vertices of at most $\mathcal{O}(d \cdot k^d) = \mathcal{O}(k^d)$ since each edge contains at most d vertices. □

The following lemma proves that every d-HITTING SET instance that is "sandwiched" between two equivalent instances must be equivalent to both. This will greatly simplify the last part of our kernelization for p-$\mathcal{Q}$, namely that of finding an instance $(\mathcal{A}', k')$ equivalent to the kernelized d-HITTING SET instance.

Lemma 3.12. *Let* $(\mathcal{H}, k)$ *be an instance of* d-HITTING SET *and let* $(\mathcal{H}^*, k)$ *be an equivalent instance with* $E(\mathcal{H}^*) \subseteq E(\mathcal{H})$*. Then for any* $\mathcal{H}'$ *with* $E(\mathcal{H}^*) \subseteq E(\mathcal{H}') \subseteq E(\mathcal{H})$ *the instance* $(\mathcal{H}', k)$ *is equivalent to* $(\mathcal{H}, k)$ *and* $(\mathcal{H}^*, k)$*.*

Proof. Observe that hitting sets for $\mathcal{H}$ can be projected to hitting sets for $\mathcal{H}'$ (i.e., restricted to the vertex set of $\mathcal{H}'$) since $E(\mathcal{H}') \subseteq E(\mathcal{H})$. Thus if $(\mathcal{H}, k)$ is a yes-instance then $(\mathcal{H}', k)$ is a yes-instance too. The same argument holds for $(\mathcal{H}', k)$ and $(\mathcal{H}^*, k)$. Together with the fact that $(\mathcal{H}, k)$ and $(\mathcal{H}^*, k)$ are equivalent, this proves the lemma. □

Now we are well equipped to prove that p-$\mathcal{Q}$ admits a polynomial kernelization.

Theorem 3.13. *Let* $\mathcal{Q} \in$ MIN $F^{+}\Pi_1$*. The standard parameterization* p-$\mathcal{Q}$ *of* $\mathcal{Q}$ *admits a polynomial kernelization.*

Proof. Let $(\mathcal{A}, k)$ be an instance of p-$\mathcal{Q}$. By Lemma 3.10 we have that $(\mathcal{A}, k)$ is a yes-instance of p-$\mathcal{Q}$ if and only if $(\mathcal{H}, k) = (\Phi(\mathcal{A}), k)$ is a yes-instance of d-HITTING SET. We apply the kernelization from Theorem 3.11 to $(\mathcal{H}, k)$ and obtain an equivalent d-HITTING SET instance $(\mathcal{H}^*, k)$ such that $E(\mathcal{H}^*) \subseteq E(\mathcal{H})$ and such that $\mathcal{H}^*$ has $\mathcal{O}(k^d)$ edges.

Every edge of $\mathcal{H}$, say $\{\mathbf{z}_1, \dots, \mathbf{z}_p\}$, corresponds to a clause $(S(\mathbf{z}_1) \vee \cdots \vee S(\mathbf{z}_p))$ of $\psi_{\mathbf{x}}(\mathcal{A}, S)$ for some $\mathbf{x} \in A^{c_x}$, by Definition 3.9. Thus for each edge $e \in E(\mathcal{H}^*) \subseteq E(\mathcal{H})$ we can select a tuple $\mathbf{x}_e$ such that e corresponds to a clause of $\psi_{\mathbf{x}_e}(\mathcal{A}, S)$. Let X be the set of the selected tuples $\mathbf{x}_e$ for all edges $e \in E(\mathcal{H}^*)$. Let $A' \subseteq A$ be the set of all components of tuples $\mathbf{x}_e \in X$, ensuring that $X \subseteq A'^{c_x}$. Let R'_i be the restriction of R_i to A' and let $\mathcal{A}' = (A', R'_1, \dots, R'_t)$.

Let $(\mathcal{H}', k) = (\Phi(\mathcal{A}'), k)$. By definition of Φ and by construction of $\mathcal{H}'$ we know that $E(\mathcal{H}^*) \subseteq E(\mathcal{H}') \subseteq E(\mathcal{H})$ since $X \subseteq A'^{c_x} \subseteq A^{c_x}$. Thus, by Lemma 3.12, we have that $(\mathcal{H}', k)$ is equivalent to $(\mathcal{H}, k)$. Furthermore, by Lemma 3.10, $(\mathcal{H}', k)$ is a yes-instance of d-HITTING SET if and only if $(\mathcal{A}', k)$ is a yes-instance of p-$\mathcal{Q}$. Thus $(\mathcal{A}', k)$ and $(\mathcal{A}, k)$ are equivalent instances of p-$\mathcal{Q}$.

We conclude the proof by giving an upper bound on the size of $(\mathcal{A}', k)$ that is polynomial in k. The set X contains at most $|E(\mathcal{H}^*)| \in \mathcal{O}(k^d)$ tuples. These tuples have no more than $c_x \cdot |E(\mathcal{H}^*)|$ different components. Hence the size of A' is $\mathcal{O}(c_x \cdot k^d) = \mathcal{O}(k^d)$. Thus the size of $(\mathcal{A}', k)$ is $\mathcal{O}(k^{dm})$, where m is the largest arity of a relation R_i, i.e., $m = \max\{r_1, \dots, r_t\}$. Thus $(\mathcal{A}', k)$ is an instance equivalent to $(\mathcal{A}, k)$ with size polynomial in k, since c_x, d, and m are constants independent of the input. □

We conclude the section by extending the polynomial kernelization to problems from WEIGHTED MIN $\mathrm{F}^{+}\Pi_1$.

Corollary 3.14. *The standard parameterization of any* WEIGHTED MIN $\mathrm{F}^{+}\Pi_1$ *problem admits a polynomial kernelization when the weights are from* $\{0\} \cup \mathbb{Q}_{\geq 1}$.

Proof. Let p-$\mathcal{Q}$ be the standard parameterization of a WEIGHTED MIN $\mathrm{F}^{+}\Pi_1$ problem. Let $(\mathcal{A}, w, k)$ be an instance of p-$\mathcal{Q}$, where $\mathcal{A} = (A, R_1, \dots, R_t)$ and $w : A^s \to \{0\} \cup \mathbb{Q}_{\geq 1}$.

It is straightforward to adapt Definition 3.9 to the weighted case. For this we reduce to a vertex weighted version of d-HITTING SET, where each vertex $\mathbf{z}$ is assigned weight $w(\mathbf{z})$.

We observe that Lemma 3.10 still works: We showed that for any relation $S \subseteq A^s$

$$(\mathcal{A}, S) \models (\forall \mathbf{x} \in A^{c_x}) : \psi(\mathbf{x}, \mathcal{A}, S) \text{ if and only if } S \text{ is a hitting set for } \Phi(\mathcal{A}).$$

Clearly, the weight of S is the same for both problems, namely $\sum_{\mathbf{z} \in S} w(\mathbf{z})$. Thus, the two instances are equivalent, and we get a version of Lemma 3.10 for the weighted case.

We will argue that the polynomial kernelization of Theorem 3.11 easily generalizes to the case with weights from $\{0\} \cup \mathbb{Q}_{\geq 1}$: As a first step, we select all vertices of weight 0 and discard all sets which contain at least one such vertex. Then, if we have a sunflower consisting of $k+1$ sets (while aiming for a hitting set of *weight* at most k), we observe that it is still mandatory to select an element from the core. The reason is that each remaining element has weight at least one. Therefore, also our approach of shrinking sunflowers from cardinality at least $k+2$ to $k+1$ is still feasible.

Finally, we observe that Lemma 3.12 and Theorem 3.13 still apply since the argument there is on the level of equivalent instances, as provided by Lemma 3.10 and Theorem 3.11. This completes the proof. □

Remark. We point out that we omit a discussion of how to encode the weights. Values from $\{0\} \cup \mathbb{Q}_{\geq 1}$ allow us to reduce the number of elements, of tuples in relations, and of weight values to a polynomial in k. Restricting the weights to be integer, or to bounded precision, permits an encoding in $\mathcal{O}(\log k)$ bits since weights of value greater than k do not need to be distinguished (e.g., they can be set to $k+1$).

Remark. For this corollary we have to exclude negative weights as well as arbitrarily small positive weights. Both types of weights would permit problems that are NP-hard even for constant values of the parameter. Concerning negative weights, consider for example an instance $(\mathcal{H}, w, k)$ of weighted d-HITTING SET. If negative weights are allowed we can introduce a new vertex v of weight $-k$ and an edge $\{v\}$. We would obtain an equivalent instance $(\mathcal{H}', w', 0)$, since v must be in any hitting set, but adding it lowers its weight by k. (Note that we have excluded negative weights in the definitions of the weighted versions.)

Similarly, when arbitrarily small positive weights are allowed, dividing all weights by k would yield an equivalent instance $(\mathcal{H}, w'', 1)$. Thus, unless P = NP, the weighted variants of NP-hard MIN $F^+\Pi_1$ problems are not fixed-parameter tractable when weights are allowed to be negative, or fractions arbitrarily close to 0. Note that, assuming that there is a lower bound of ϵ on the positive weights, we may as well normalize the weights and the parameter by $\frac{1}{\epsilon}$. Clearly linear changes to the parameter do not affect the admittance of polynomial kernelizations: $k^{\mathcal{O}(1)} = (\frac{1}{\epsilon}k)^{\mathcal{O}(1)}$.

3.4. Polynomial kernelization for MAX NP

In this section we show that the standard parameterization of any MAX NP problem admits a polynomial kernelization.

Again let us fix some problem $\mathcal{Q} \in$ MAX NP on finite structures of type $(r_1, \dots, r_t)$. Let $R_1, \dots, R_t$ be matching relation symbols. By definition of MAX NP there is a tuple of relation symbols $\mathcal{S} = (S_1, \dots, S_u)$ of arities $s_1, \dots, s_u$ and a formula $\psi(\mathbf{x}, \mathbf{y}, \mathcal{A}, \mathcal{S})$ in disjunctive normal form over the vocabulary $\{R_1, \dots, R_t, S_1, \dots, S_u\}$ such that for all finite structures $\mathcal{A}$ of type $(r_1, \dots, r_t)$ the optimum value of $\mathcal{Q}$ on input $\mathcal{A}$ can be expressed as

$$\max_{\mathcal{S}} |\{\mathbf{x} \in A^{c_x} : (\mathcal{A}, \mathcal{S}) \models (\exists \mathbf{y} \in A^{c_y}) : \psi(\mathbf{x}, \mathbf{y}, \mathcal{A}, \mathcal{S})\}|.$$

Let d be the maximum number of occurrences of relations $S_1, \dots, S_u$ in any disjunct of $\psi(\mathbf{x}, \mathbf{y}, \mathcal{A}, \mathcal{S})$. The standard parameterization p-$\mathcal{Q}$ of $\mathcal{Q}$ is the following problem:

Standard parameterization of $\mathcal{Q}$

Input: A finite structure $\mathcal{A}$ of type $(r_1, \dots, r_t)$ and an integer k.

Parameter: k.

Task: Decide whether $\mathrm{opt}_{\mathcal{Q}}(\mathcal{A}) \geq k$.

We define formulas $\psi_{\mathbf{x},\mathbf{y}}(\mathcal{A},\mathcal{S})$ similarly to Definition 3.8 in Section 3.3.

Definition 3.15. Let $\mathcal{A} = (A, R_1, \dots, R_t)$ be a finite structure of type $(r_1, \dots, r_t)$, let $\mathbf{x} \in A^{c_x}$, and let $\mathbf{y} \in A^{c_y}$. We define $\psi_{\mathbf{x},\mathbf{y}}(\mathcal{A}, S)$ to be the formula obtained by the following steps:

1. Replace all variables $x_1, \dots, x_{c_x}, y_1, \dots, y_{c_y}$ by the chosen elements of A.
2. Replace all literals $R_i(\mathbf{z})$ and $\neg R_i(\mathbf{z})$, for some $\mathbf{z} \in A^{r_i}$, by 1 (true) or 0 (false) depending on whether $\mathbf{z}$ is contained in R_i.
3. Delete all disjuncts that contain a 0 and delete all occurrences of 1; note the difference to Definition 3.8 through using a different normal form.
4. Delete all disjuncts that contain contradicting literals $S_j(\mathbf{z})$ and $\neg S_j(\mathbf{z})$ since they cannot be satisfied.

We explicitly allow empty disjuncts that are satisfied by definition for the sake of simplicity (they occur when all literals in a disjunct are evaluated to 1).

It is easy to see that $\psi(\mathbf{x}, \mathbf{y}, \mathcal{A}, \mathcal{S})$ and $\psi_{\mathbf{x},\mathbf{y}}(\mathcal{A},\mathcal{S})$ are equivalent for any choice of $\mathbf{x}$, $\mathbf{y}$, and $\mathcal{S}$, i.e., $(\mathcal{A},\mathcal{S}) \models \psi(\mathbf{x}, \mathbf{y}, \mathcal{A}, \mathcal{S})$ iff $(\mathcal{A},\mathcal{S}) \models \psi_{\mathbf{x},\mathbf{y}}(\mathcal{A},\mathcal{S})$. Moreover, we can compute all formulas $\psi_{\mathbf{x},\mathbf{y}}(\mathcal{A},\mathcal{S})$ for $\mathbf{x} \in A^{c_x}$, $\mathbf{y} \in A^{c_y}$ in polynomial time, since c_x, c_y, and the length of $\psi(\mathbf{x}, \mathbf{y}, \mathcal{A}, \mathcal{S})$ are constants independent of $\mathcal{A}$.

The following definition introduces sets $X_{\mathcal{A}}$ and $Y_{\mathcal{A}}(\mathbf{x})$ intended to capture all tuples $\mathbf{x}$ and $\mathbf{y}$, respectively, for which $\psi_{\mathbf{x},\mathbf{y}}(\mathcal{A},\mathcal{S})$ is satisfiable. We will see afterwards that a large set $X_{\mathcal{A}}$ implies that the instance is positive.

Definition 3.16. Let $\mathcal{A} = (A, R_1, \dots, R_t)$ be a finite structure of type $(r_1, \dots, r_t)$.
(a) We define $X_{\mathcal{A}} \subseteq A^{c_x}$ as the set of all tuples $\mathbf{x}$ such that $(\exists \mathbf{y}) : \psi_{\mathbf{x},\mathbf{y}}(\mathcal{A},\mathcal{S})$ holds for some $\mathcal{S}$:

$$X_{\mathcal{A}} = \{\mathbf{x} : (\exists \mathcal{S}) : (\mathcal{A},\mathcal{S}) \models (\exists \mathbf{y}) : \psi_{\mathbf{x},\mathbf{y}}(\mathcal{A},\mathcal{S})\}.$$

(b) For $\mathbf{x} \in A^{c_x}$ we define $Y_{\mathcal{A}}(\mathbf{x})$ as the set of all tuples $\mathbf{y}$ such that $\psi_{\mathbf{x},\mathbf{y}}(\mathcal{A},\mathcal{S})$ holds for some $\mathcal{S}$:

$$Y_{\mathcal{A}}(\mathbf{x}) = \{\mathbf{y} : (\exists \mathcal{S}) : (\mathcal{A},\mathcal{S}) \models \psi_{\mathbf{x},\mathbf{y}}(\mathcal{A},\mathcal{S})\}.$$

The sets $X_{\mathcal{A}}$ and $Y_{\mathcal{A}}(\mathbf{x})$ can be computed in polynomial time because the number of tuples $\mathbf{x} \in A^{c_x}$ respectively $\mathbf{y} \in A^{c_y}$ is polynomial in the size of $\mathcal{A}$ and the formula $\psi(\mathbf{x}, \mathbf{y}, \mathcal{A}, \mathcal{S})$ is of constant length independent of $\mathcal{A}$.

Lemma 3.17. *Let $(\mathcal{A}, k)$ be an instance of p-$\mathcal{Q}$. If $|X_{\mathcal{A}}| \geq k \cdot 2^d$ then* $\mathrm{opt}_{\mathcal{Q}}(\mathcal{A}) \geq k$*, i.e., $(\mathcal{A}, k)$ is a yes-instance.*

Remark. In the following proof we consider assignments to variables of the formulas $\psi_{\mathbf{x},\mathbf{y}}(\mathcal{A},\mathcal{S})$. We point out that assigning true or false to some variable $S_i(\mathbf{z})$ corresponds to including or excluding, respectively, the tuple $\mathbf{z}$ in S_i. Note that there are $\sum_{i=1}^{u} |A|^{s_i}$ variables, one for each possible tuple of a relation S_i of arity s_i.

Proof of Lemma 3.17. We follow Papadimitriou and Yannakakis' [99] proof for the fact that all problems in MAX NP are constant-factor approximable. For each $\mathbf{x} \in X_\mathcal{A}$ we fix a tuple $\mathbf{y} \in Y_\mathcal{A}(\mathbf{x})$ such that $\psi_{\mathbf{x},\mathbf{y}}(\mathcal{A}, \mathcal{S})$ is satisfiable. This yields $m = |X_\mathcal{A}|$ formulas, say $\psi_1, \dots, \psi_m$. Now, for each formula ψ_i let f_i denote the fraction of all assignments to $\mathcal{S}$ (i.e., inclusion or exclusion of tuples $\mathbf{z}$ in the relations S_j) that satisfies ψ_i.

We will create an assignment that satisfies at least $\sum f_i$ formulas ψ_i. Let y be a variable that has not been assigned yet. We assume that ℓ variables are unassigned at that point and that $\sum f'_i \geq \sum f_i$, where the fractions f'_i are with respect to assignments to these ℓ remaining variables. For $i \in \{1, \dots, m\}$, let p_i and n_i denote the fraction of assignments to the remaining variables that satisfies ψ_i in which y is set to true or false, respectively. Thus there are $2^\ell(p_i + n_i)$ assignments which satisfy ψ_i. Assign true to y if $\sum p_i \geq \sum n_i$; else, assign false. We show that the sum of fractions f'_i never decreases (always taking f'_i to be with respect to the remaining unassigned variables): If y is set to true, then $2^\ell p_i$ assignments to the other $\ell - 1$ variables satisfy ψ_i, which corresponds to a fraction of $2^\ell p_i / 2^{\ell-1} = 2p_i$. Thus if $\sum p_i \geq \sum n_i$ then

$$\sum_{i=1}^m 2p_i \geq \sum_{i=1}^m p_i + \sum_{i=1}^m n_i \geq \sum_{i=1}^m f'_i \geq \sum_{i=1}^m f_i.$$

Note that $\sum_{i=1}^m 2p_i$ is the sum of fractions of satisfying assignments taken with respect to the remaining $\ell - 1$ variables. Similarly for the case that $\sum p_i < \sum n_i$ and y is assigned false. Thus the sum of fractions never decreases.

When all variables are assigned a value, f'_i is equal to 1 if ψ_i is satisfied and 0 else. Thus, this assignment satisfies at least $\sum f'_i \geq \sum f_i$ formulas ψ_i (recall that each satisfied formula contributes a tuple to the solution).

It is easy to see that $f_i \geq 2^{-d}$ for each formula ψ_i. Since ψ_i is satisfiable there exists a satisfiable disjunct. To satisfy a disjunct of at most d literals, at most d variables need to be assigned accordingly. Since the assignment to all other variables can be arbitrary this implies that $f_i \geq 2^{-d}$. Thus we have that $\sum f_i \geq m \cdot 2^{-d}$. Therefore $|X_\mathcal{A}| = m \geq k \cdot 2^d$ implies that the assignment satisfies at least k formulas, i.e., that $\mathrm{opt}_\mathcal{Q}(\mathcal{A}) \geq k$. □

Henceforth we assume that $|X_\mathcal{A}| < k \cdot 2^d$; we have seen that $(\mathcal{A}, k)$ is a yes-instance otherwise. The remaining and more involved part is to bound and reduce the size of the sets $Y_\mathcal{A}(\cdot)$. Let us emphasize the crucial difference between $X_\mathcal{A}$ and sets $Y_\mathcal{A}(\cdot)$: every tuple $\mathbf{x} \in X_\mathcal{A}$ can add to the solution value, whereas tuples $\mathbf{y} \in Y_\mathcal{A}(\mathbf{x})$ only provide different ways of satisfying $(\exists \mathbf{y} \in A^{c_y}) : \psi_{\mathbf{x},\mathbf{y}}(\mathcal{A}, \mathcal{S})$. Hence our goal is to shrink the sets $Y_\mathcal{A}(\mathbf{x})$ without harming satisfiability. We consider $(\exists \mathbf{y} \in A^{c_y}) : \psi_{\mathbf{x},\mathbf{y}}(\mathcal{A}, \mathcal{S})$ on the level of single disjuncts.

Definition 3.18. Let $(\mathcal{A}, k)$ be an instance of p-$\mathcal{Q}$ with $\mathcal{A} = (A, R_1, \dots, R_t)$. For $\mathbf{x} \in X_\mathcal{A}$ we define $D_\mathcal{A}(\mathbf{x})$ as the set of all disjuncts of $\psi_{\mathbf{x},\mathbf{y}}(\mathcal{A}, \mathcal{S})$ over all $\mathbf{y} \in Y_\mathcal{A}(\mathbf{x})$.

To reduce the size of sets $D_\mathcal{A}(\mathbf{x})$, which will lead to a decreased number of tuples in $Y_\mathcal{A}(\mathbf{x})$, we use the Sunflower Lemma. We will see that large sunflowers among disjuncts in $D_\mathcal{A}(\mathbf{x})$ represent redundant ways of satisfying $(\exists \mathbf{y} \in A^{c_y}) : \psi_{\mathbf{x},\mathbf{y}}(\mathcal{A}, \mathcal{S})$.

The following definition of intersections and sunflowers among disjuncts treats disjuncts like sets of literals.

Definition 3.19. We define the *intersection of two disjuncts* as the conjunction of all literals that occur in both disjuncts. A *sunflower of a set of disjuncts* is a subset such that each pair of disjuncts in the subset has the same intersection (modulo permutation of the literals).

Definition 3.20. A *partial assignment* is a set L of literals such that no literal is the negation of another literal in L. A formula is *satisfiable under* L if there exists an assignment that satisfies the formula and each literal in L.

The following lemma is the basis of our data reduction. It shows that satisfiability under small partial assignments can be preserved in a reduced set of disjuncts. The small partial assignments will later arise in the main theorem, since we ask for solutions of value at least k. This corresponds to (at least) k satisfied tuples, each requiring only one satisfied disjunct (of at most d literals); hence it suffices to consider partial assignments of at most dk literals.

Lemma 3.21. *Let $(\mathcal{A}, k)$ be an instance of p-$\mathcal{Q}$. For each $\mathbf{x} \in X_{\mathcal{A}}$ there exists a set $D^*_{\mathcal{A}}(\mathbf{x}) \subseteq D_{\mathcal{A}}(\mathbf{x})$ of cardinality $\mathcal{O}(k^d)$ such that:*

1. *For every partial assignment L of at most dk literals, $D^*_{\mathcal{A}}(\mathbf{x})$ contains a disjunct satisfiable under L, if and only if $D_{\mathcal{A}}(\mathbf{x})$ contains a disjunct satisfiable under L.*
2. *$D^*_{\mathcal{A}}(\mathbf{x})$ can be computed in time polynomial in $|\mathcal{A}|$.*

Proof. Let $\mathcal{A} = (A, R_1, \dots, R_t)$ be a finite structure of type $(r_1, \dots, r_t)$, let $\mathbf{x} \in X_{\mathcal{A}}$, and let $D_{\mathcal{A}}(\mathbf{x})$ be a set of disjuncts according to Definition 3.18. We compute the set $D^*_{\mathcal{A}}(\mathbf{x})$ starting from $D^*_{\mathcal{A}}(\mathbf{x}) = D_{\mathcal{A}}(\mathbf{x})$ and successively shrinking sunflowers while the cardinality of $D^*_{\mathcal{A}}(\mathbf{x})$ is greater than $(dk+1)^d \cdot d! \cdot d$.

We compute a sunflower of cardinality $dk+2$, say $F = \{f_1, \dots, f_{dk+2}\}$, in time polynomial in $|D^*_{\mathcal{A}}(\mathbf{x})|$ using Corollary 2.4. We delete a disjunct of F, say f_{dk+2}, from $D^*_{\mathcal{A}}(\mathbf{x})$. Let O and P be copies of $D^*_{\mathcal{A}}$ before respectively after deleting f_{dk+2}. Observe that $F' := F \setminus \{f_{dk+2}\}$ is a sunflower of cardinality $dk+1$ in P. Let L be a partial assignment of at most dk literals and assume that no disjunct in P is satisfiable under L. This means that for each disjunct of P there is a literal in L that contradicts it, i.e., a literal that is the negation of a literal in the disjunct. We focus on the sunflower F' in P. There must be a literal in L, say ℓ, that contradicts at least two disjuncts of F', say f and f', since $|F'| = dk+1$ and $|L| \leq dk$. Therefore ℓ is the negation of a literal in the intersection of f and f', i.e., the core of F'. Thus ℓ contradicts also f_{dk+2} and we conclude that no disjunct in $O = P \cup \{f_{dk+2}\}$ is satisfiable under the partial assignment L. The reverse argument holds since all disjuncts of P are contained in O. Thus each step maintains the desired property (1).

At the end $D^*_{\mathcal{A}}(x)$ contains no more than $(dk+1)^d \cdot d! \cdot d \in \mathcal{O}(k^d)$ disjuncts. The size of $D_{\mathcal{A}}(\mathbf{x})$ is bounded by the size of $Y_{\mathcal{A}}(\mathbf{x}) \subseteq A^{c_y}$ times the number of disjuncts of $\psi(\mathbf{x},\mathbf{y},\mathcal{A},\mathcal{S})$ which is a constant independent of $\mathcal{A}$. Thus the size of $D_{\mathcal{A}}(\mathbf{x})$ is bounded by a polynomial in $|\mathcal{A}|$. Hence,the computation takes time polynomial in the size of $\mathcal{A}$ since a disjunct is deleted in each step. □

As in the previous section we are able to generate a kernelized instance of another problem that is easier to handle. The sets $D^*_{\mathcal{A}}(\mathbf{x})$ describe a possibly different formula for each $\mathbf{x}$. However, it is more intuitive to view them as an image of the original instance on which it is easier to draw conclusions.

We can now show the main result of this section.

Theorem 3.22. *Let $\mathcal{Q} \in$ MAX NP. The standard parameterization p-$\mathcal{Q}$ of $\mathcal{Q}$ admits a polynomial kernelization.*

Proof. The proof is organized in three parts. First, given an instance $(\mathcal{A},k)$ of p-$\mathcal{Q}$, we construct an instance $(\mathcal{A}',k)$ of p-$\mathcal{Q}$ in time polynomial in the size of $(\mathcal{A},k)$. In the second part, we prove that $(\mathcal{A},k)$ and $(\mathcal{A}',k)$ are equivalent. In the third part, we complete the proof by showing that the size of $(\mathcal{A}',k)$ is bounded by a polynomial in k.

Transformation Let $(\mathcal{A},k)$ be an instance of p-$\mathcal{Q}$. We use the sets $D_{\mathcal{A}}(\mathbf{x})$ and $D^*_{\mathcal{A}}(\mathbf{x})$ according to Definition 3.18 and Lemma 3.21. Recall that $D_{\mathcal{A}}(\mathbf{x})$ is the set of all disjuncts of $\psi_{\mathbf{x},\mathbf{y}}(\mathcal{A},\mathcal{S})$ for $\mathbf{y} \in Y_{\mathcal{A}}(\mathbf{x})$. Thus, for each disjunct $\delta \in D^*_{\mathcal{A}}(\mathbf{x}) \subseteq D_{\mathcal{A}}(\mathbf{x})$, we can select a $\mathbf{y}_\delta \in Y_{\mathcal{A}}(\mathbf{x})$ such that δ is a disjunct of $\psi_{\mathbf{x},\mathbf{y}_\delta}(\mathcal{A},\mathcal{S})$. Let $Y^*_{\mathcal{A}}(\mathbf{x}) \subseteq Y_{\mathcal{A}}(\mathbf{x})$ be the set of these selected tuples $\mathbf{y}_\delta$.

For each $\mathbf{x}$ this takes time $\mathcal{O}(|D^*_{\mathcal{A}}(\mathbf{x})| \cdot |Y_{\mathcal{A}}(\mathbf{x})|) \subseteq \mathcal{O}(k^d \cdot |A|^{c_y})$. Computing $Y^*_{\mathcal{A}}(\mathbf{x})$ for all $\mathbf{x} \in A^{c_x}$ takes time $\mathcal{O}(|A|^{c_x} \cdot k^d \cdot |A|^{c_y})$, i.e., time polynomial in the size of $(\mathcal{A},k)$ since k is never larger than $|A|^{c_x}$ (i.e., $(\mathcal{A},k)$ is a no-instance if $k > |A|^{c_x}$ since k exceeds the number of tuples $\mathbf{x} \in A^{c_x}$).

Let $A' \subseteq A$ be the set of all components of $\mathbf{x} \in X_{\mathcal{A}}$ and $\mathbf{y} \in Y^*_{\mathcal{A}}(\mathbf{x})$ for all $\mathbf{x} \in X_{\mathcal{A}}$. This ensures that $X_{\mathcal{A}} \subseteq (A')^{c_x}$ and $Y^*_{\mathcal{A}}(\mathbf{x}) \subseteq (A')^{c_y}$ for all $\mathbf{x} \in X_{\mathcal{A}}$. Let R'_i be the restriction of R_i to A' and let $\mathcal{A}' = (A', R'_1, \ldots, R'_t)$.

Correctness We will now prove that $\mathrm{opt}_{\mathcal{Q}}(\mathcal{A}) \geq k$ if and only if $\mathrm{opt}_{\mathcal{Q}}(\mathcal{A}') \geq k$, i.e., that $(\mathcal{A},k)$ and $(\mathcal{A}',k)$ are equivalent. Assume that $\mathrm{opt}_{\mathcal{Q}}(\mathcal{A}) \geq k$ and let $\mathcal{S} = (S_1,\ldots,S_u)$ such that

$$|\{\mathbf{x} : (\mathcal{A},\mathcal{S}) \models (\exists \mathbf{y}) : \psi(\mathbf{x},\mathbf{y},\mathcal{A},\mathcal{S})\}| \geq k.$$

This implies that there must exist tuples $\mathbf{x}_1,\ldots,\mathbf{x}_k \in A^{c_x}$ and $\mathbf{y}_1,\ldots,\mathbf{y}_k \in A^{c_y}$ such that $\mathcal{S}$ satisfies $\psi_{\mathbf{x}_i,\mathbf{y}_i}(\mathcal{A},\mathcal{S})$ for $i = 1,\ldots,k$. Thus $\mathcal{S}$ must satisfy at least one disjunct in each $\psi_{\mathbf{x}_i,\mathbf{y}_i}(\mathcal{A},\mathcal{S})$ since these formulas are in disjunctive normal form. Accordingly let $\delta_1,\ldots,\delta_k$ be disjuncts such that $\mathcal{S}$ satisfies the disjunct δ_i in $\psi_{\mathbf{x}_i,\mathbf{y}_i}(\mathcal{A},\mathcal{S})$ for $i = 1,\ldots,k$. We show that there exists $\mathcal{S}'$ such that

$$|\{\mathbf{x} : (\mathcal{A}',\mathcal{S}') \models (\exists \mathbf{y}) : \psi(\mathbf{x},\mathbf{y},\mathcal{A}',\mathcal{S}')\}| \geq k.$$

For $p = 1,\ldots,k$ we apply the following procedure: If $\mathbf{y}_p \in Y^*_{\mathcal{A}}(\mathbf{x}_p)$ then do nothing. Otherwise consider the partial assignment L consisting of the at most dk literals of the disjuncts $\delta_1, \ldots, \delta_k$. The set $D_{\mathcal{A}}(\mathbf{x}_p)$ contains a disjunct that is satisfiable under L, namely δ_p. By Lemma 3.21, the set $D^*_{\mathcal{A}}(\mathbf{x}_p)$ also contains a disjunct satisfiable under L, say δ'_p. Let $\mathbf{y}'_p \in Y^*_{\mathcal{A}}(\mathbf{x}_p)$ such that δ'_p is a disjunct of $\psi_{\mathbf{x}_p,\mathbf{y}'_p}(\mathcal{A},\mathcal{S})$. Change $\mathcal{S}$ in the following way to satisfy the disjunct δ'_p. For each literal of δ'_p of the form $S_i(\mathbf{z})$ add $\mathbf{z}$ to the relation S_i. Similarly for each literal of the form $\neg S_i(\mathbf{z})$ remove $\mathbf{z}$ from S_i. This does not change the fact that $\mathcal{S}$ satisfies the disjunct δ_i in $\psi_{\mathbf{x}_i,\mathbf{y}_i}(\mathcal{A},\mathcal{S})$ for $i = 1, \ldots, k$ since, by selection, δ'_p is satisfiable under L. Then we replace $\mathbf{y}_p$ by $\mathbf{y}'_p$ and δ_p by δ'_p. Thus we maintain that $\mathcal{S}$ satisfies δ_i in $\psi_{\mathbf{x}_i,\mathbf{y}_i}(\mathcal{A},\mathcal{S})$ for $i = 1, \ldots, k$.

Afterwards we obtain $\mathcal{S}$ as well as tuples $\mathbf{x}_1, \ldots, \mathbf{x}_k$, $\mathbf{y}_1, \ldots, \mathbf{y}_k$ with $\mathbf{y}_i \in Y^*_{\mathcal{A}}(\mathbf{x}_i)$, and disjuncts $\delta_1, \ldots, \delta_k$ such that $\mathcal{S}$ satisfies δ_i in $\psi_{\mathbf{x}_i,\mathbf{y}_i}(\mathcal{A},\mathcal{S})$ for $i = 1, \ldots, k$. Let $\mathcal{S}'$ be the restriction of $\mathcal{S}$ to A'. Then we have that $(\mathcal{A}',\mathcal{S}') \models \psi_{\mathbf{x}_i,\mathbf{y}_i}(\mathcal{A}',\mathcal{S}')$ for $i = 1, \ldots, k$ since A' is defined to contain the components of tuples $\mathbf{x} \in X_{\mathcal{A}}$ and of all tuples $\mathbf{y} \in Y^*_{\mathcal{A}}(\mathbf{x})$ for $\mathbf{x} \in X_{\mathcal{A}}$. Hence $\mathbf{x}_i \in \{\mathbf{x} : (\mathcal{A}',\mathcal{S}') \models (\exists \mathbf{y}) : \psi(\mathbf{x},\mathbf{y},\mathcal{A}',\mathcal{S}')\}$ for $i = 1, \ldots, k$. Thus $\text{opt}_{\mathcal{Q}}(\mathcal{A}') \geq k$.

For the reverse direction assume that $\text{opt}_{\mathcal{Q}}(\mathcal{A}') \geq k$. Since $A' \subseteq A$ it follows that

$$\{\mathbf{x} : (\mathcal{A}',\mathcal{S}') \models (\exists \mathbf{y}) : \psi(\mathbf{x},\mathbf{y},\mathcal{A}',\mathcal{S}')\} \quad \subseteq \quad \{\mathbf{x} : (\mathcal{A},\mathcal{S}') \models (\exists \mathbf{y}) : \psi(\mathbf{x},\mathbf{y},\mathcal{A},\mathcal{S}')\}.$$

Thus $|\{\mathbf{x} : (\mathcal{A},\mathcal{S}') \models (\exists \mathbf{y}) : \psi(\mathbf{x},\mathbf{y},\mathcal{A},\mathcal{S}')\}| \geq k$, implying that $\text{opt}_{\mathcal{Q}}(\mathcal{A}) \geq k$. Therefore $\text{opt}_{\mathcal{Q}}(\mathcal{A}) \geq k$ if and only if $\text{opt}_{\mathcal{Q}}(\mathcal{A}') \geq k$. Hence $(\mathcal{A}, k)$ and $(\mathcal{A}', k)$ are equivalent instances of p-$\mathcal{Q}$.

Bounding the size We conclude the proof by providing an upper bound on the size of $(\mathcal{A}', k)$ that is polynomial in k. For the sets $Y^*_{\mathcal{A}}(\mathbf{x})$ we selected one tuple $\mathbf{y}$ for each disjunct in $D^*_{\mathcal{A}}(\mathbf{x})$. Thus $|Y^*_{\mathcal{A}}(\mathbf{x})| \leq |D^*(\mathbf{x})| \in \mathcal{O}(k^d)$ for all $\mathbf{x} \in X_{\mathcal{A}}$. The set A' contains the components of tuples $\mathbf{x} \in X_{\mathcal{A}}$ and of all tuples $\mathbf{y} \in Y^*_{\mathcal{A}}(\mathbf{x})$ for $\mathbf{x} \in X_{\mathcal{A}}$. Thus

$$\begin{aligned} |A'| &\leq c_x \cdot |X_{\mathcal{A}}| + c_y \cdot \sum_{\mathbf{x} \in X_{\mathcal{A}}} |Y^*_{\mathcal{A}}(\mathbf{x})| \\ &\leq c_x \cdot |X_{\mathcal{A}}| + c_y \cdot |X_{\mathcal{A}}| \cdot \mathcal{O}(k^d) \\ &< c_x \cdot k \cdot 2^d + c_y \cdot k \cdot 2^d \cdot \mathcal{O}(k^d) = \mathcal{O}(k^{d+1}). \end{aligned}$$

For each relation R'_i we have $|R'_i| \leq |A'|^{r_i} \in \mathcal{O}(k^{(d+1)r_i})$. Thus the size of $(\mathcal{A}', k)$ is bounded by $\mathcal{O}(k^{(d+1)m})$, where m is the largest arity of a relation R_i. □

Again, we are able to extend the kernelization to the weighted case, provided that the weights are neither negative nor arbitrarily close to zero.

Corollary 3.23. *The standard parameterization of any* WEIGHTED MAX NP *problem admits a polynomial kernelization when the weights are from* $\{0\} \cup \mathbb{Q}_{\geq 1}$.

Proof. Let $(\mathcal{A}, w, k)$ be an input instance. We let $X_{\mathcal{A}}$ denote the tuples $\mathbf{x}$ with weight greater than zero for which $(\exists \mathbf{y}) : \psi(\mathbf{x},\mathbf{y},\mathcal{A},\mathcal{S})$ is satisfiable. Then, by Lemma 3.17, if $|X_{\mathcal{A}}| \geq k \cdot 2^d$, we can efficiently find an assignment which satisfies $(\exists \mathbf{y}) : \psi(\mathbf{x},\mathbf{y},\mathcal{A},\mathcal{S})$ for at least k tuples $\mathbf{x} \in X_{\mathcal{A}}$. Since $w(\mathbf{x}) \geq 1$ for such tuples, this assignment corresponds to a solution with value at least k, and thus $(\mathcal{A}, k)$ is a yes-instance. Otherwise we have $|X_{\mathcal{A}}| < k \cdot 2^d$.

In Theorem 3.22, it is showed that for the obtained instance $(\mathcal{A}', k)$ the exact same tuples $\mathbf{x}$ can be satisfied as for $(\mathcal{A}, k)$ (the converse is trivial). Hence, equivalence holds for the weighted case as well. Note that since positive weights are of value at least one, it again suffices to satisfy the formula for k tuples (or less). □

3.5. Linear kernelization for MAX SNP

For MAX SNP there is an implicit stronger kernelization result based on Lemma 3.17.

Corollary 3.24. *Let $\mathcal{Q} \in$* MAX SNP*. The standard parameterization p-$\mathcal{Q}$ of $\mathcal{Q}$ admits a polynomial kernelization with a linear bound on the size of the base set of the obtained finite structure.*

Proof. Let $\mathcal{Q}$ be an optimization problem on finite structures of type $(r_1, \ldots, r_t)$ that is contained in MAX SNP. Let $\mathcal{S} = (S_1, \ldots, S_u)$ be a tuple of relation symbols of arities $s_1, \ldots, s_u$. Finally let $\psi(\mathbf{x}, \mathcal{A}, \mathcal{S})$ be a formula in disjunctive normal form such that the optimum value of $\mathcal{Q}$ on finite structures $\mathcal{A}$ of type $(r_1, \ldots, r_t)$ can be expressed as

$$\max_{\mathcal{S}} |\{\mathbf{x} : (\mathcal{A}, \mathcal{S}) \models \psi(\mathbf{x}, \mathcal{A}, \mathcal{S})\}|.$$

Now, let $(\mathcal{A}, k)$ be an instance of p-$\mathcal{Q}$, with $\mathcal{A} = (A, R_1, \ldots, R_t)$. Similarly to Definition 3.16, we consider the set $X_{\mathcal{A}}$ of all tuples $\mathbf{x}$ such that $\psi_{\mathbf{x}}(\mathcal{A}, \mathcal{S})$ holds for some $\mathcal{S}$:

$$X_{\mathcal{A}} = \{\mathbf{x} : (\exists \mathcal{S}) : (\mathcal{A}, \mathcal{S}) \models \psi_{\mathbf{x}}(\mathcal{A}, \mathcal{S})\}.$$

By Lemma 3.17 (easily adapted to MAX SNP), if $|X_{\mathcal{A}}| \geq k \cdot 2^d$ then $\text{opt}_{\mathcal{Q}}(\mathcal{A}) \geq k$ and we may accept $\mathcal{A}$ as a yes-instance. Otherwise $|X_{\mathcal{A}}| \in \mathcal{O}(k)$ and by restricting A to those elements that occur in elements of $X_{\mathcal{A}}$ we obtain A' with $|A'| \in \mathcal{O}(k)$. Also restricting the relations R_i to A' we obtain an equivalent instance $\mathcal{A}' = (A', R_1', \ldots, R_t')$ of total size $\mathcal{O}(k^m)$ where $m = \max\{r_1, \ldots, r_t\}$. □

When considering the weighted version, restricted to weight zero and positive weights of at least one, we can get a stronger kernelization result.

Corollary 3.25. *Let $\mathcal{Q}$ be a* WEIGHTED MAX SNP *problem. The standard parameterization p-$\mathcal{Q}$ of $\mathcal{Q}$ admits a kernelization with a linear bound on size of the base set, with a linear number of tuples in each relation, and such that the weight function maps only a linear number of tuples to a non-zero value.*

Proof. Let $(\mathcal{A}, w, k)$ be an instance of p-$\mathcal{Q}$. Following the proof of Corollary 3.23, it suffices to argue the case that $|X_{\mathcal{A}}| < k \cdot 2^d \in \mathcal{O}(k)$. Let A' be the set of elements of A that occur in tuples of $X_{\mathcal{A}}$.

We make a new weight function $w' : A'^{c_x} \to \{0\} \cup \mathbb{Q}_{\geq 1}$ as

$$w'(\mathbf{x}) = \begin{cases} w(\mathbf{x}) & \text{if } \mathbf{x} \in X_{\mathcal{A}}, \\ 0 & \text{else.} \end{cases}$$

Now we create relations R'_i by deleting all tuples from the relations R_i which do not occur in $\psi(\mathbf{x}, \mathcal{A}, \mathcal{S})$ for any $\mathbf{x} \in X_{\mathcal{A}}$, i.e.,

$$R'_i = \{\mathbf{z} \in R_i : (R_i(\mathbf{z})) \text{ or } (\neg R_i(\mathbf{z})) \text{ is a literal of } \psi(\mathbf{x}, \mathcal{A}, \mathcal{S}) \text{ for some } \mathbf{x} \in X_{\mathcal{A}}\}\,.$$

Let $(\mathcal{A}', w', k)$ denote the kernelized instance, with $\mathcal{A}' = (A', R'_1, \ldots, R'_t)$. We argue that $(\mathcal{A}', w', k)$ is a yes-instance if and only if $(\mathcal{A}, w, k)$ is a yes-instance.

Let $\mathcal{S}$ be a tuple of relations such that $(\mathcal{A}, \mathcal{S}) \models \psi(\mathbf{x}, \mathcal{A}, \mathcal{S})$ for tuples $\mathbf{x}$ with a total weight of at least k. Let $\mathcal{S}'$ denote the restriction of $\mathcal{S}$ to A'. Let $\mathbf{x}$ be a tuple of non-zero weight (i.e., of weight at least one) for which $(\mathcal{A}, \mathcal{S}) \models \psi(\mathbf{x}, \mathcal{A}, \mathcal{S})$. Hence we have that $\mathbf{x} \in X_{\mathcal{A}}$. We observe that by definition of the relations R'_i, we have $\mathbf{z} \in R_i$ if and only if $\mathbf{z} \in R'_i$, for all literals $(R_i(\mathbf{z}))$ or $(\neg R_i(\mathbf{z}))$ of $\psi(\mathbf{x}, \mathcal{A}, \mathcal{S})$. Thus $(\mathcal{A}', \mathcal{S}') \models \psi(\mathbf{x}, \mathcal{A}', \mathcal{S}')$. Hence $\mathcal{S}'$ satisfies $\psi(\mathbf{x}, \mathcal{A}', \mathcal{S}')$ for tuples $\mathbf{x}$ with a total weight of at least k too.

For the converse we may restrict our attention to those tuples with positive weight. For these tuples and the corresponding formulas we have preserved the relations R_i, thus if they are satisfied by $(\mathcal{A}', \mathcal{S}')$ then the same is true for $(\mathcal{A}, \mathcal{S}')$. □

Remark. The basic idea is that we only need to keep those tuples of relations R_i that occur in $\psi(\mathbf{x}, \mathcal{A}, \mathcal{S})$ for $\mathbf{x} \in X_{\mathcal{A}}$, since the other tuples $\mathbf{x} \in A^{c_x} \setminus X_{\mathcal{A}}$ cannot contribute to the solution: Their weight is zero, or $\psi(\mathbf{x}, \mathcal{A}, \mathcal{S})$ cannot be satisfied. However, changing the relations R_i (other than by projection) may make $\psi(\mathbf{x}, \mathcal{A}, \mathcal{S})$ satisfiable. This side-effect is handled by setting the weight of all tuples from $A^{c_x} \setminus X_{\mathcal{A}}$ to zero.

3.6. Summary

We showed that the standard parameterizations of MIN $\mathrm{F}^{+}\Pi_1$ and MAX NP problems admit polynomial kernelizations, also extending to the weighted versions. For the subclass MAX SNP of MAX NP we even obtained kernelizations with a linear-sized base set for both the weighted and the unweighted case. Thus for a syntactically defined part of the class APX of constant-factor approximable problems we have established a strong connection to polynomial kernelization.

Possible extensions to larger subclasses of APX may be based on approximability preserving reductions; recall that APX and APX-PB are the closures of MAX SNP under PTAS- and E-reductions, respectively. However, APX-PB $\subseteq$ APX already contains CONNECTED VERTEX COVER and BIN PACKING; the former does not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly [46], the latter is not in XP unless P $=$ NP (packing into $k = 2$ bins is equivalent to the NP-complete PARTITION problem). Thus more restricted reductions would be necessary for this approach.

4. Lower bounds for edge modification problems

4.1. Introduction

In this chapter we consider the kernelizability of edge modification problems. Given a graph G and an integer k, the Π EDGE COMPLETION/EDITING/DELETION problem asks whether it is possible to add, edit, or delete at most k edges in G such that the resulting graph has the property Π:

> **Π EDGE COMPLETION/EDITING/DELETION**
>
> **Input:** A graph $G = (V, E)$ and an integer k.
>
> **Parameter:** k.
>
> **Task:** Decide whether adding, editing, respectively deleting at most k edges in G yields a graph with property Π.

We answer negatively an open problem posed by Cai at IWPEC 2006 (cf. [15]), namely whether the Π EDGE DELETION problem admits a polynomial kernelization when Π can be characterized by finitely many forbidden induced subgraphs. For this we use the recent framework for lower bounds for kernelization by Bodlaender et al. [16] (see Section 1.4). Our result complements the considerable number of polynomial kernelizations for edge modification problems, e.g., [23, 45, 60, 61, 65] (see Table 4.1).

Edge modification problems have a number of applications, including machine learning, numerical algebra, and molecular biology [26, 92, 105, 108]. In typical applications the input graphs arise from experiments and edge modification serves to correct the (hopefully) few errors. For similarity clustering, for example, the vertices represent entities that are linked by an edge if their similarity exceeds a certain threshold. Given a perfect similarity measure one would obtain a cluster graph, i.e., a disjoint union of cliques. In practice though, the obtained graph will at best be close to a cluster graph, i.e., within few edge modifications. Clearly a large number of modifications would yield a clustering of the entities that deviates strongly from the similarity measure, allowing the conclusion that the measure is probably faulty. Thus fpt-algorithms that solve the problem efficiently when the number of modifications is not too large are a good way of solving this and similar problems.

Characterization by finitely many forbidden induced subgraphs A graph property (or class of graphs) Π is *hereditary*, if it is closed under taking induced subgraphs. Thus any

Property/Class	Modification	Kernel size		Characterization
Chain	Deletion	$\mathcal{O}(k^2)$	[60]	$(2K_2, K_3, C_5)$-free
Split	Deletion	$\mathcal{O}(k^4)$	[60]	$(2K_2, C_4, C_5)$-free
Threshold	Deletion	$\mathcal{O}(k^3)$	[60]	$(2K_2, C_4, P_4)$-free
Co-Trivially Perfect	Deletion	$\mathcal{O}(k^3)$	[60]	$(2K_2, P_4)$-free
Triangle-Free	Deletion	$6k$	[23]	K_3-free
Cluster	Editing	$4k$	[61]	P_3-free
Chordal	Completion	$\mathcal{O}(k^2)$	[91]	C_i-free, $i \geq 4$

Table 4.1.: Kernelization bounds for some edge modification problems for hereditary properties. The kernel size refers to the number of vertices.

such property is characterized by the (possibly infinite) set of minimal graphs which do not have the property, e.g., *chordal graphs* are exactly the graphs without induced cycles of length at least four. Some properties admit a characterization by a finite number of forbidden induced subgraphs, e.g.:

Cluster graphs: *Cluster graphs* are disjoint unions of cliques. They are also exactly the graphs which do not contain an induced path P_3 (on three vertices), i.e., cluster = P_3-free.

Split graphs: A graph is *split* if its vertices can be partitioned into an independent set and a clique. Equivalently a graph is split if and only if it does not contain $2K_2$, C_4, or C_5. Thus split = $(2K_2, C_4, C_5)$-free.

Related work There exists rich literature on the complexity of edge modification problems; recent surveys were given by Natanzon et al. [92] and Burzyn et al. [26]. In 1996 Cai [27] showed that a very general version of this problem, also allowing vertex deletions, is fixed-parameter tractable when Π can be characterized by a finite set of forbidden induced subgraphs. In Table 4.1 we give an overview of kernelization results for a number of edge modification problems, along with their $\mathcal{H}$-free characterizations. Six of the properties listed in the table can be characterized by a finite set of forbidden induced subgraphs, leading to the question whether a finite characterization implies the existence of a polynomial kernelization. It is easy to see that the answer is yes for the vertex deletion setting since this translates directly to d-HITTING SET: Given an instance (G, k) we collect all vertex subsets that induce a forbidden subgraph, and ask for a hitting set of size at most k; the constant d is the largest number of vertices among forbidden induced subgraphs (see Section 2.2 for a d-HITTING SET kernelization). It is crucial that vertex deletion cannot create new copies of forbidden induced subgraphs.

Cai posed the question whether the same is true for the Π EDGE DELETION problem, i.e., whether it admits a polynomial kernelization when Π can be characterized by finitely many forbidden induced subgraphs. The kernelizability of the $\mathcal{H}$-free EDGE EDITING problem is open as well. We point out that $\mathcal{H}$-free EDGE COMPLETION is equivalent to $\bar{\mathcal{H}}$-free EDGE DELETION, where $\bar{\mathcal{H}}$ contains the complements of the graphs in $\mathcal{H}$.

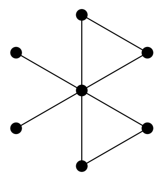

Figure 4.1.: A graph H such that H-free EDGE DELETION and H-free EDGE EDITING do not admit polynomial kernelizations (assuming NP $\nsubseteq$ co-NP/poly).

Our work Following joint work with Magnus Wahlström [80], we consider a simple parameterized satisfiability problem, called Not-1-in-3 SAT and show that it does not admit a polynomial kernelization, by proving compositionality. Then we reduce this problem to H-free EDGE DELETION and H-free EDGE EDITING for a graph H on seven vertices (see Figure 4.1). Thus Π EDGE DELETION and Π EDGE EDITING do not generally admit polynomial kernelizations, even when Π can be characterized by a single forbidden induced subgraph (assuming NP $\nsubseteq$ co-NP/poly). Our result builds upon the lower bound framework by Bodlaender et al. [16] (Section 1.4).

Structure of this chapter In Section 4.2 we introduce the Not-1-in-3 SAT problem and show that it is NP-complete as well as compositional. Thus, Not-1-in-3 SAT does not admit a polynomial kernelization (unless NP $\subseteq$ co-NP/poly), and we then extend this lower bound also to the case without repeated variables to simplify the later reduction. In Section 4.3 we give polynomial parameter transformations to H-free EDGE DELETION and H-free EDGE EDITING, proving the claimed lower bounds. We conclude with a short summary and open problems in Section 4.4.

4.2. Hardness of Not-1-In-3 SAT

In this section, we show that the Not-1-in-3 SAT problem does not admit a polynomial kernelization, unless NP $\subseteq$ co-NP/poly. A polynomial parameter transformation to an H-free EDGE DELETION problem and an H-free EDGE EDITING problem in Section 4.3 will then yield the desired lower bounds.

Not-1-In-3 SAT We define Not-1-in-3 SAT as a satisfiability problem asking for a satisfying assignment with at most k true variables to a conjunction of not-1-in-3-constraints $(x + y + z \neq 1)$, and constraints $(x \neq 0)$. Inputs to the problem consist of such a conjunction $\mathcal{F}$ and an integer k. An assignment ϕ to the variables of $\mathcal{F}$ is satisfying if $\phi(x) = 1$ for every variable x with a constraint $(x \neq 0)$ and

$$(\phi(x), \phi(y), \phi(z)) \in \{(0,0,0), (0,1,1), (1,0,1), (1,1,0), (1,1,1)\}$$

for every not-1-in-3-constraint $(x + y + z \neq 1)$ in $\mathcal{F}$.

Not-1-In-3 SAT

Input: A formula $\mathcal{F}$ on Boolean variables that is a conjunction of not-1-in-3-constraints as well as constraints $(x \neq 0)$; an integer k.

Parameter: k.

Task: Decide whether there exists a satisfying assignment of weight at most k, i.e., with at most k true variables.

Remark. In the context of the constraint satisfaction problems studied in the subsequent chapters Not-1-in-3 SAT would be called Min Ones SAT($\{(x+y+z \neq 1), (x)\}$).

To apply Theorem 1.4 we need to show that Not-1-in-3 SAT is compositional and that the unparameterized version is NP-hard. For the compositionality proof we need to be able to force any two variables to be equal; the following lemma addresses this issue.

Lemma 4.1. *Let* $(\mathcal{F}, k)$ *be an instance of* Not-1-in-3 SAT *and let* x *and* y *be two specified variables. Adding only not-1-in-3-constraints, we can create an instance* $(\mathcal{F}', k)$ *such that there is a satisfying assignment for* $\mathcal{F}'$ *of weight at most* k *if and only if there is a satisfying assignment* ϕ *for* $\mathcal{F}$ *of weight at most* k *such that* $\phi(x) = \phi(y)$*. Furthermore, every satisfying assignment* ϕ' *of weight at most* k *for* $\mathcal{F}'$ *has* $\phi'(x) = \phi'(y)$*.*

Proof. Let $(\mathcal{F}, k)$ be an instance of Not-1-in-3 SAT and let x and y be two variables of $\mathcal{F}$. Starting from $\mathcal{F}' = \mathcal{F}$, we add $k+1$ new variables $z_1, \ldots, z_{k+1}$. Then we add $k+1$ not-1-in-3-constraints $(x+y+z_i \neq 1)$, i.e., one not-1-in-3-constraint for each $i \in \{1, \ldots, k+1\}$.

Now, to argue correctness, let ϕ be a satisfying assignment for $\mathcal{F}$ with at most k true variables and such that $\phi(x) = \phi(y)$. We extend ϕ to a satisfying assignment ϕ' for $\mathcal{F}'$ by assigning false to all new variables z_i. It suffices to check that the added constraints are satisfied: We have $\phi'(x) = \phi'(y)$ and $\phi'(z_i) = 0$ for all $i \in \{1, \ldots, k+1\}$. Therefore all not-1-in-3-constraints $(x+y+z_i \neq 1)$ are satisfied.

For the converse, let ϕ' be a satisfying assignment for $\mathcal{F}'$ with at most k true variables. We show that $\phi'(x) = \phi'(y)$; then restricting ϕ' to the variables of $\mathcal{F}$ gives the claimed assignment ϕ. We observe that there must be a $j \in \{1, \ldots, k+1\}$ such that ϕ' assigns false to z_j. Therefore, the constraint $(x+y+z_j \neq 1)$ enforces equality of x and y. □

Now we will prove that Not-1-in-3 SAT is NP-complete and compositional. NP-completeness follows immediately from the classification of approximability of Boolean constraint satisfaction problems due to Khanna et al. [75]. Compositionality requires more work; observe that simply making disjoint copies of the given instances does not work, since the number of true variables will be counted over all parts of the joined instance. Our strategy is to "switch off" the instances by making them zero-valid, conditioned on one *activity variable* s_i per instance: We replace all $(x \neq 0)$ constraints by $(x = s_i)$. Thus if s_i is false then the corresponding instance is zero-valid. Otherwise, if s_i is true, then the $(x \neq 0)$ constraints are enforced by $(x = s_i)$ and the assignment (for the joined instance) must satisfy the corresponding instance. The additional and crucial component is a tree-like formula that forces at least one activity variable to be true, i.e., forcing at least one of the original instances to be active.

Lemma 4.2. Not-1-in-3 SAT *is NP-complete and compositional.*

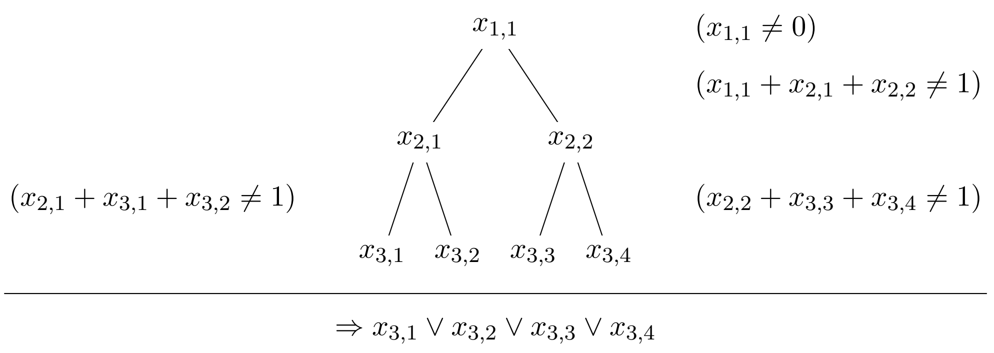
$x_{1,1}$
$(x_{1,1} \neq 0)$
$(x_{1,1} + x_{2,1} + x_{2,2} \neq 1)$
$x_{2,1}$
$x_{2,2}$
$(x_{2,1} + x_{3,1} + x_{3,2} \neq 1)$
$(x_{2,2} + x_{3,3} + x_{3,4} \neq 1)$
$x_{3,1}$
$x_{3,2}$
$x_{3,3}$
$x_{3,4}$
$\Rightarrow x_{3,1} \vee x_{3,2} \vee x_{3,3} \vee x_{3,4}$

Figure 4.2.: The first three levels of a composition tree. Every parent-node is in a not-1-in-3-constraint with its two sons, e.g., $(x_{2,1} + x_{3,1} + x_{3,2} \neq 1)$. If $x_{1,1}$ is true, then on every level i some variable $x_{i,j}$ must be true, which leads to $(x_{3,1} \vee x_{3,2} \vee x_{3,3} \vee x_{3,4})$ being satisfied.

Proof. NP-completeness follows from Khanna et al.'s [75] characterization of the approximability of min ones constraint satisfaction problems (the two constraints $(x+y+z \neq 1)$ and $(x \neq 0)$ are not both zero-valid, Horn, or width-2 affine; cf. Section 5).

The NP-completeness extends also to the variant where k is encoded in unary: If k is greater than the number of variables then we may reduce it to match that number without changing the outcome; adding the unary encoding of k can at most double the instance size. We will now show that Not-1-in-3 SAT is compositional.

Let $(\mathcal{F}_1, k)$, ..., $(\mathcal{F}_t, k)$ be t instances of Not-1-in-3 SAT. In time polynomial in their total size we will create a new instance $(\mathcal{F}', k')$ with $k' = \mathcal{O}(k)$ that is a yes-instance if and only if $(\mathcal{F}_i, k)$ is a yes-instance for some $1 \leq i \leq t$. If $t \geq 3^k$, then we have time to solve every instance exactly by branching, and output some dummy yes- or no-instance. We assume for the rest of the proof that $t < 3^k$. We also assume that t is a power of two (or else duplicate some instance $(\mathcal{F}_i, k)$): say $t = 2^\ell$.

We start building $\mathcal{F}'$ by creating variable-disjoint copies of all formulas $\mathcal{F}_i$. For every formula $\mathcal{F}_i$, we additionally create an activity variable s_i. Now for every variable x in $\mathcal{F}_i$ such that $\mathcal{F}_i$ contains a constraint $(x \neq 0)$, we replace this constraint by $(x = s_i)$ (according to Lemma 4.1). The variable s_i now decides whether $\mathcal{F}_i$ is active: if $s_i = 0$ then every variable of $\mathcal{F}_i$ can be set to zero, and if $s_i = 1$, then all constraints of the formula $\mathcal{F}_i$ are active. We will complete the construction by creating a formula that forces some s_i to be true, in a form of *composition tree.*

Figure 4.2 illustrates the construction used for this part. We use variables $x_{i,j}$ where i is the *level* of the variable, $i \in \{1, \ldots, \ell{+}1\}$, and $1 \leq j \leq 2^{i-1}$. The first-level variable $x_{1,1}$ is forced to be true by a constraint $(x_{1,1} \neq 0)$, and for every internal variable in the composition tree, we use a not-1-in-3-constraint $(x_{i,j} + x_{i+1,2j-1} + x_{i+1,2j} \neq 1)$. Finally, for every variable $x_{\ell+1,j}$ on the last level, we add a constraint $(s_j = x_{\ell+1,j})$.

Observe that a not-1-in-3-constraint $(x + y + z \neq 1)$ implies $(x \rightarrow (y \vee z))$. Thus, if some variable on level i is assigned true then one of its two sons on level $i + 1$ must be assigned true. Thus, we see inductively that at least one variable per level must be true. Furthermore, for every variable $x_{\ell+1,j}$ on the last level, there exists an assignment where $x_{\ell+1,j} = 1$ and exactly one variable per level is true: To construct it assign true to all variables on the path from $x_{\ell+1,j}$ to $x_{1,1}$ and false to the remaining variables. Thus, assuming a minimal assignment to the composed instance $\mathcal{F}'$, we can treat the composition tree as adding weight exactly $\ell + 1$ and encoding a disjunction over the activity variables s_j. We obtain the composed instance $(\mathcal{F}', k + \ell + 2)$.

Now if $\mathcal{F}'$ has an assignment ϕ' of weight at most $k + \ell + 2$ then $\phi'(s_j) = 1$ for some $j \in \{1, \ldots, t\}$. Thus the constraints of $\mathcal{F}_j$ are active and ϕ' can be restricted to a satisfying assignment for $\mathcal{F}_j$ (excluding s_j) of weight at most k. If some input formula $\mathcal{F}_j$ has an assignment of weight at most k then this can be extended to an assignment for $\mathcal{F}'$ of weight at most $k+\ell+2$ by assigning 1 to s_j and to each variable on the path from $x_{\ell+1,j}$ to $x_{1,1}$ in the composition tree, and 0 to all other variables of $\mathcal{F}'$. Finally, the parameter $k' = k + \ell + 2$ is $\mathcal{O}(k)$ since $t < 3^k$ and $\ell = \log t$, and the work performed is polynomial in the length of all input instances. □

Now by Theorem 1.4 we obtain the following result.

Theorem 4.3. *The* Not-1-in-3 SAT *problem does not admit a polynomial kernelization unless* NP $\subseteq$ co-NP/poly.

To simplify the reduction to H-free EDGE DELETION in the following section, we consider a variant of Not-1-in-3 SAT that does not allow repeated variables in the not-1-in-3-constraints, e.g., $(x + x + y \neq 1)$. The following lemma implicitly extends our lower bound to that variant.

Lemma 4.4. Not-1-in-3 SAT *reduces to* Not-1-in-3 SAT without repeated variables *by a polynomial parameter transformation.*

Proof. Observe that the construction of Lemma 4.1 to enforce an equality constraint works without repeating variables. Therefore, from an instance $(\mathcal{F}, k)$ of Not-1-in-3 SAT, we can create an instance $(\mathcal{F}', k')$ of Not-1-in-3 SAT without repeated variables where $k' = 2k$ in the following manner:

1. For every variable x in $\mathcal{F}$, create two variables x, x' in $\mathcal{F}'$. Force $(x = x')$ according to Lemma 4.1.

2. For every not-1-in-3-constraint $(x + x + y \neq 1)$ in $\mathcal{F}$, with $x \neq y$, place $(x + x' + y \neq 1)$ in $\mathcal{F}'$ (ditto for the other orderings of x, x', and y). Copy not-1-in-3-constraints $(x + y + z \neq 1)$ for distinct variables x, y, z and constraints $(x \neq 0)$ into $\mathcal{F}'$, and ignore any not-1-in-3-constraints $(x + x + x \neq 1)$.

The formula $\mathcal{F}'$ has a solution with at most $2k$ true variables iff $\mathcal{F}$ has a solution with at most k true variables. □

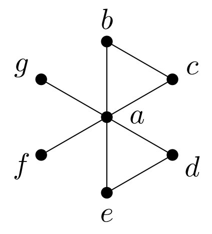

Figure 4.3.: The graph H for which we show that H-free EDGE DELETION and H-free EDGE EDITING do not admit polynomial kernelizations.

4.3. Lower bounds for two H-free modification problems

Throughout this section let $H = (V_H, E_H)$ with vertex set $V_H = \{a, b, c, d, e, f, g\}$ and edge set $E_H = \{\{a, b\}, \{a, c\}, \{a, d\}, \{a, e\}, \{a, f\}, \{a, g\}, \{b, c\}, \{d, e\}\}$, as depicted in Figure 4.3. We will show that Not-1-in-3 SAT without repeated variables reduces to H-free EDGE DELETION by a polynomial parameter transformation, thereby proving that the latter problem does not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly, according to Theorem 1.5. Consecutively we use a slightly more involved polynomial parameter transformation from Not-1-in-3 SAT without repeated variables to H-free EDGE EDITING.

Lemma 4.5. Not-1-in-3 SAT without repeated variables *reduces to* H-free EDGE DELETION *by a polynomial parameter transformation.*

Proof. Let $(\mathcal{F}, k)$ be an instance of Not-1-in-3 SAT without repeated variables. We will construct an equivalent instance (G, k') of H-free EDGE DELETION.

Transformation Let α be the number of variables x of $\mathcal{F}$ that have a constraint $(x \neq 0)$. Clearly, if $k < \alpha$ then $(\mathcal{F}, k)$ is a no-instance and we map it to a dummy no-instance of H-free EDGE DELETION, e.g., $(H, 0)$. Henceforth we assume that $k \geq \alpha$ and we let $k' = k - \alpha$. Starting from the empty graph we use the following two steps to construct G:

1. For each variable x of $\mathcal{F}$ we add two vertices p_x and q_x to G. We add the edge $\{p_x, q_x\}$ to G if there is no constraint $(x \neq 0)$ in the formula $\mathcal{F}$.
2. For each not-1-in-3-constraint $(x + y + z \neq 1)$ add $k' + 1$ new vertices $r_1, \ldots, r_{k'+1}$ to G. Connect each r_i to p_x, q_x, p_y, q_y, p_z, and q_z. Recall that x, y, and z are different by our restriction on the source problem.

See Figure 4.4 for an example of a formula $\mathcal{F}$ and the resulting graph. Observe that this construction can be accomplished in polynomial time and that $k' \in \mathcal{O}(k)$.

Correctness We will now show that $(\mathcal{F}, k)$ is a yes-instance of Not-1-in-3 SAT without repeated variables if and only if (G, k') is a yes-instance of H-free EDGE DELETION.

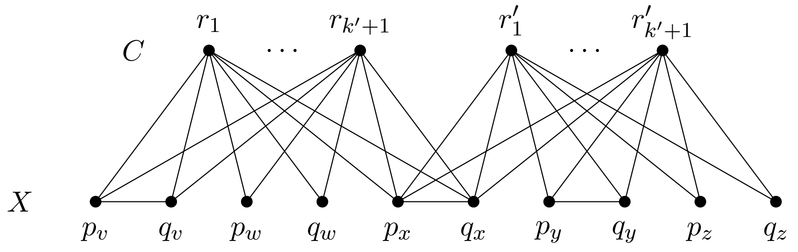

Figure 4.4.: The graph obtained by applying the reduction from Lemma 4.5 on the formula $\mathcal{F} = (v + w + x \neq 1) \wedge (x + y + z \neq 1) \wedge (w \neq 0) \wedge (z \neq 0)$.

We begin with two simple observations; let X denote the set of vertices created in Step 1 and let C denote the set of vertices created in Step 2:

(1) Each vertex of X has at most one neighbor in X.

(2) C is an independent set in G.

Suppose that $(\mathcal{F}, k)$ is a yes-instance of Not-1-in-3 SAT without repeated variables and let ϕ be a satisfying assignment for $\mathcal{F}$ with at most k true variables. We define a subset $D \subseteq E$ as those edges $\{p_x, q_x\}$ of G with $\phi(x) = 1$. Recall that $\{p_x, q_x\} \in E$ if and only if $\mathcal{F}$ contains no constraint $(x \neq 0)$. Therefore $|D| \leq k - \alpha = k'$ since at most k variables x have $\phi(x) = 1$ but α of those variables have a constraint $(x \neq 0)$.

We assume for contradiction that $G - D$ is not H-free. Let $a, \dots, g$ be vertices of G that induce a subgraph H. For simplicity let the adjacencies be the same as in Figure 4.3.

It is easy to see that $a \in C$: Otherwise, if $a \in X$, then by Observation (1) at most one of its neighbors can be in X. This would imply that $b, c \in C$ or $d, e \in C$ contradicting Observation (2), namely, that C is an independent set.

Since $a \in C$ the other vertices $b, \dots, g$ must be in X by Observation (2). Recall that in $G[X]$, i.e., among vertices of X, there are only edges of the form $\{p_x, q_x\}$ where x is a variable of $\mathcal{F}$. Since $\{b, c\}$ and $\{d, e\}$ are edges of $G[X]$ there must be variables x and y of $\mathcal{F}$ such that w.l.o.g. $b = p_x$, $c = q_x$, $d = p_y$, and $e = q_y$. By Step 2 of the construction there must be a third variable z of $\mathcal{F}$ such that w.l.o.g. $f = p_z$ and $g = q_z$. Thus the vertex a corresponds to the constraint $(x + y + z \neq 1)$ of $\mathcal{F}$.

From $\{p_x, q_x\}, \{p_y, q_y\} \in E \setminus D$ (the edge set of $G - D$) it follows that $\phi(x) = \phi(y) = 0$. Since $\{p_z, q_z\} \notin E \setminus D$ it follows that $\{p_z, q_z\} \in D$ or $\{p_z, q_z\} \notin E$. In the former case we have that $\phi(z) = 1$. In the latter case there must be a constraint $(z \neq 0)$ in $\mathcal{F}$, which also implies that $\phi(z) = 1$ since ϕ is satisfying for $\mathcal{F}$. However, $\phi(x) = \phi(y) = 0$ and $\phi(z) = 1$ contradict the feasibility of ϕ since $(x + y + z \neq 1)$ is a not-1-in-3-constraint of $\mathcal{F}$. Hence $G - D$ is an H-free graph, implying that (G, k') is a yes-instance of H-free EDGE DELETION, since $|D| \leq k'$.

Now, suppose that (G, k') is a yes-instance of H-free EDGE DELETION. We will construct a satisfying assignment ϕ with at most k true variables for $\mathcal{F}$. Let $D \subseteq E$

with $|D| \leq k'$ such that $G - D$ is H-free. We define ϕ by

$$\phi(x) = \begin{cases} 1 & \text{if } \{p_x, q_x\} \in D, \\ 1 & \text{if } (x \neq 0) \text{ is a constraint of } \mathcal{F}, \\ 0 & \text{otherwise.} \end{cases}$$

Thus the number of true variables that ϕ assigns is at most the cardinality of D plus the number of variables x of $\mathcal{F}$ with an $(x \neq 0)$-constraint, i.e., at most $|D| + \alpha \leq k$.

We assume for contradiction that ϕ is not satisfying for $\mathcal{F}$. This implies that there is a not-1-in-3-constraint, say $(x + y + z \neq 1)$, in $\mathcal{F}$ that is not satisfied by ϕ. The reason is that, by definition, ϕ satisfies all $(x \neq 0)$-constraints. Not satisfying $(x + y + z \neq 1)$ implies that ϕ sets exactly one of the three variables to 1. Recall that x, y, and z must be pairwise different by problem definition. W.l.o.g. we assume that $\phi(x) = 1$ and $\phi(y) = \phi(z) = 0$. This immediately implies that there are no constraints $(y \neq 0)$ or $(z \neq 0)$ in $\mathcal{F}$ and that $\{p_y, q_y\}$ and $\{p_z, q_z\}$ are not contained in D, by definition of ϕ. Hence $\{p_y, q_y\}$ and $\{p_z, q_z\}$ are contained in the edge set $E \setminus D$ of $G - D$.

Since $\phi(x) = 1$ we conclude that $\{p_x, q_x\} \in D$ or that $(x \neq 0)$ is a constraint of $\mathcal{F}$. In the latter case, by Step 1 in the construction of G, the edge $\{p_x, q_x\}$ does not occur in G. Thus, in both cases, $\{p_x, q_x\}$ is not an edge of $G - D$, i.e., $\{p_x, q_x\} \notin E \setminus D$.

We will now consider the vertices that were added for $(x + y + z \neq 1)$ in Step 2 of the construction, say $r_1, \ldots, r_{k'+1}$. Since $|D| \leq k'$ at least one r_i is adjacent to all six vertices p_x, q_x, p_y, q_y, p_z, and q_z. Together with r_i those vertices induce a subgraph H. This contradicts the choice of D, implying that ϕ is satisfying for $\mathcal{F}$.

Thus $(\mathcal{F}, k)$ is a yes-instance of Not-1-in-3 SAT without repeated variables if and only if (G, k') is a yes-instance of H-free EDGE DELETION. □

By Theorem 1.5 we obtain the desired result.

Theorem 4.6. *For the graph $H = (V_H, E_H)$ with vertices $V_H = \{a, b, c, d, e, f, g\}$ and edges $E_H = \{\{a, b\}, \{a, c\}, \{a, d\}, \{a, e\}, \{a, f\}, \{a, g\}, \{b, c\}, \{d, e\}\})$ the H-free* EDGE DELETION *problem does not admit a polynomial kernelization unless* NP $\subseteq$ co-NP/poly.

The following corollary proves that the H-free EDGE EDITING problem does not admit a polynomial kernelization either.

Corollary 4.7. *For the same graph H as in Theorem* 4.6*, the H-free* EDGE EDITING *problem does not admit a polynomial kernelization unless* NP $\subseteq$ co-NP/poly.

Proof. We extend the construction used in Lemma 4.5 to obtain a polynomial parameter transformation from Not-1-in-3 SAT without repeated variables to H-free EDGE EDITING.

Transformation Let $(\mathcal{F}, k)$ be an instance of Not-1-in-3 SAT without repeated variables. We create an instance (G, k') according to the construction in Lemma 4.5, except for using $3k' + 1$ vertices r_i per not-1-in-3-constraint in Step 2.

Essentially we only need one additional gadget to ensure that no edges are added between vertices of X in G (recall that X contains vertices p_x and q_x for every variable of $\mathcal{F}$). Adding this gadget for every non-edge of $G[X]$ will yield a graph G' and we will show that (G', k') and $(\mathcal{F}, k)$ are equivalent.

Let us recall that the only edges in $G[X]$ are of the form $\{p_x, q_x\}$ for some variable x of $\mathcal{F}$ that has no constraint ($x \neq 0$). For every other pair of vertices, say (p_x, p_y) with $x \neq y$, we add $k' + 1$ vertices s_i, t_i, u_i, v_i, and w_i, and for each i connect s_i to p_x, p_y, t_i, u_i, v_i, and w_i, and connect t_i to u_i. That is, create constructions which will induce $k'+1$ subgraphs H if the edge $\{p_x, p_y\}$ is added. Note that these subgraphs have only one pair of vertices in common. The same construction applies also for pairs (p_x, q_y) and (q_x, q_y) as well as for pairs (p_x, q_x) where x has a constraint ($x \neq 0$) in $\mathcal{F}$. Let G' denote the graph that is obtained by adding these vertices and edges to G.

Correctness Now, for proving equivalence of (G', k') and $(\mathcal{F}, k)$, let us first assume that (G', k') is a yes-instance of H-free EDGE EDITING and let D be a set of edges, with $|D| \leq k'$ such that $G' \triangle D := (V(G'), E(G') \triangle D)$ is H-free.

We claim that D adds no edges between vertices of X. Otherwise, if for example $\{p_x, p_y\} \in D$ then, referring to the new gadget, there would be $k' + 1$ induced subgraphs H that share only p_x and p_y. Since removal of these $k' + 1$ subgraphs would demand at least $k'+1$ modifications, this would imply that $G' \triangle D$ is not H-free. Thus D restricted to $G'[X]$ contains only deletions; let D' be these deletions. Clearly $|D'| \leq k'$.

Let us consider the r-vertices that are added in Step 2. Clearly, since $|D| \leq k'$, for every not-1-in-3-constraint of $\mathcal{F}$ at least $k' + 1$ of its corresponding vertices r_i are not incident with edges of D (recall that we started with $3k' + 1$ vertices r_i per constraint). We select a set R of vertices, picking, for every not-1-in-3-constraint of $\mathcal{F}$, $k' + 1$ of its corresponding r-vertices which are not incident with edges of D. Let G'' be the graph induced by $R \cup X$.

Now let (G_0, k') be the H-free EDGE DELETION-instance created from $(\mathcal{F}, k)$ according to Lemma 4.5. It is easy to see that G_0 and G'' are isomorphic, and that the only modifications made to G'' by D are the deletions D'. Since $G' \triangle D$ is H-free, so is the graph $G'' \triangle D' = (G' \triangle D)[R \cup X]$. Thus D' constitutes a solution to the H-free EDGE DELETION instance (G'', k), which by Lemma 4.5 maps to a solution for $(\mathcal{F}, k)$.

Let us now assume that $(\mathcal{F}, k)$ is a yes-instance and let ϕ be a satisfying assignment of weight at most k. Let D be obtained as in Lemma 4.5, i.e., D will only delete some $\{p_x, q_x\}$-edges from G'. We show that $G' \triangle D$ is H-free.

Consider the vertex which would form the vertex a of H, i.e., the vertex of degree six in H. It cannot be a p-vertex, since the neighborhood of a p-vertex contains only edges incident to the corresponding q-vertex (all r- and s-vertices are independent of one another, and a p-vertex has only such neighbors, in addition to its q-vertex). By symmetry, this excludes q-vertices as well. The r-vertices have only six neighbors, so any induced H would correspond to a contradicted constraint R which by assumption does not exist (same argument as in Lemma 4.5). Furthermore, in the solution constructed, the neighborhood of any s-vertex contains only one edge, as D only deletes edges. The remaining types of vertices, i.e., the t, u, v, and w vertices of the gadget, have degree less than six. □

4.4. Summary

We showed that the H-free EDGE DELETION problem and the H-free EDGE EDITING problem do not generally admit polynomial kernelizations (assuming NP $\not\subseteq$ co-NP/poly). This answers an open question by Cai, namely, whether the $\mathcal{H}$-free EDGE DELETION problem admits a polynomial kernelization for every finite set $\mathcal{H}$ of graphs. We point out that any $\mathcal{H}$-free EDGE DELETION problem can be expressed as a min ones constraint satisfaction problem (introduced in Chapter 5 and studied in Chapter 6), but the problems obtained in this way are overly general. The reason is that constraints may be freely placed but globally forbidding a subgraph leads to constraints emerging from the edges of the graph (including possible new copies of forbidden subgraphs, created by edge modifications).

It is an interesting open problem to further characterize kernelizability of $\mathcal{H}$-free edge modification problems. Considering the structure of the known positive examples (see introduction), one might first ask whether the $\mathcal{H}$-free EDGE DELETION problem always admits a polynomial kernelization when $\mathcal{H}$ consists only of paths, cycles, and cliques (or sums thereof, such as $2K_2$). On the other hand, the graph H used for the two lower bound proofs suggests a wide range of similar graphs such that H-free EDGE DELETION or H-free EDGE EDITING do not admit polynomial kernelizations.

Part III.

Kernelizability of Boolean constraint satisfaction problems

5. Boolean constraint satisfaction problems

5.1. Introduction

This chapter serves the purpose of providing an introduction to the Boolean constraint satisfaction problem (Boolean CSP). Constraint satisfaction problems are a framework that allows us to express, in a natural way, many combinatorial problems.

Let us begin by recalling the d-HITTING SET problem. There, given a hypergraph, i.e., hyperedges over a set of vertices, the task is to find a smallest set of vertices that intersects every edge. Let us abstract a little: We choose to consider Boolean variables, instead of vertices, and our task will be to set few variables to true (i.e., to 1), instead of selecting few vertices. If the assignment to the variables shall satisfy some set of positive clauses, then clearly, we have obtained an equivalent formulation of d-HITTING SET. Generalizing from d-HITTING SET we will now say that the positive clauses are one possible choice of *constraints*; each constraint restricts the possible assignments to its variables (i.e., those that occur in the constraint). If we allow other constraints, then we obtain a wide range of problems, called Min Ones SAT(Γ) problems (where Γ denotes the allowed constraints); we will later formally introduce and study these and similar problems. Note that already deciding satisfiability becomes an interesting task once the constraints are more complicated than positive clauses. Further, there are many other goals to optimize for, all of them encompass well-known and much-studied problems.

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 2-SAT and 3-SAT by allowing that constraints are given by arbitrary relations, not necessarily by clauses. Clearly, each relation can be expressed as a certain number of clauses, but focusing on the relations we can make more careful distinctions than restricting the length or the number of negations of clauses.

As Boolean CSP problems are NP-hard in general, there have been intensive efforts at finding efficiently solvable special cases. One approach is to consider the complexity of the graph (or hypergraph) underlying the formula, e.g., giving a fast algorithm for when the graph has bounded treewidth (i.e., if the graph is rather tree-like). Another 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 2-SAT. On the other hand, if Γ contains all the ternary relations, then SAT(Γ) is more general than 3-SAT, and hence it is NP-hard. In this thesis, we take the second approach.

A celebrated classical result of T.J. Schaefer [107] from 1978 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 [82] 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 none of the SAT(Γ) problems are intermediate. 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.

Schaefer's result has been generalized in various directions. Bulatov [24] 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 [25, 50]). Feder and Vardi [50] conjectured that CSP(Γ) is either in P or NP-hard for any Γ over finite domain; this is known as the *CSP Dichotomy Conjecture*. They arrived at this conjecture while pursuing the task of finding a large syntactic subclass of NP that exhibits a dichotomy, i.e., such that the problems are either in P or NP-hard, contrasting Ladner's Theorem [82].

Furthermore, there are numerous results for Boolean CSPs: Creignou et al. [39] classified the polynomial-time solvable cases of the problem Exact Ones SAT(Γ), where the task is to find a satisfying assignment with exactly k true variables, for some integer k given in the input. Marx [88] classified the parameterized complexity of Exact Ones SAT(Γ) problems into fixed-parameter tractable and W[1]-complete cases. Natural optimization variants of SAT(Γ) were considered in [36, 42, 75], one of the goals being to classify the approximability of the different problems: In Max SAT(Γ) we have to find an assignment maximizing the number of satisfied constraints (generalizing MAX CUT), while in Min UnSAT(Γ) we have to find an assignment minimizing the number of dissatisfied constraints (generalizing EDGE BIPARTIZATION). 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 (generalizing VERTEX COVER and CLIQUE).

We have seen that (Boolean) constraint satisfaction problems generalize many natural and well-studied problems. Thus they are a good framework for obtaining more general results, e.g., to what constraint types can we generalize the polynomial kernelization for d-HITTING SET? Additionally, there are many dichotomy results for SAT(Γ) and variants thereof. Such complete classifications into tractable and (likely) intractable cases provide a more profound insight into the nature of problems, e.g., compare Schaefer's [107] dichotomy for the SAT(Γ) problem (see Theorem 5.3) to the folklore result that 2-SAT is in P whereas 3-SAT is NP-complete. The latter hides many positive cases of essentially unbounded arity, e.g., Horn clauses or affine clauses (see the following section for definitions); uncovering such examples is one of the virtues of dichotomies.

Structure of this part In Chapters 6, 7, and 8 we will classify Min Ones SAT(Γ), Max Ones SAT(Γ), and Exact Ones SAT(Γ) problems according to when they admit a polynomial kernelization (depending on Γ). In the remaining part of this chapter we will introduce the necessary definitions as well as some background knowledge about constraint satisfaction problems and Boolean relations.

5.2. Definitions

In this section we present basic definitions as well as some background knowledge on Boolean constraint satisfaction problems. We emphasize that parts of this section are intended to provide an overview and a high-level concept of the more involved connections between the notions. A brief summary of the most frequently used connections can be found at the end of this section.

Notation

A *literal* is a (Boolean) variable or the negation thereof, we also call it a *positive* or *negative literal*, respectively. *Positive clauses* and *negative clauses* are disjunctions of positive and negative literals, respectively. We use 1 and 0 to denote true and false values, respectively.

For two tuples $\alpha = (\alpha_1, \dots, \alpha_r)$, $\beta = (\beta_1, \dots, \beta_r)$, we let $\alpha \wedge \beta = (\alpha_1 \wedge \beta_1, \dots, \alpha_r \wedge \beta_r)$, and likewise for other Boolean functions. We write $\alpha \leq \beta$ meaning that $\alpha_i \leq \beta_i$ for every $1 \leq i \leq r$; and $\alpha < \beta$ stands for $\alpha \leq \beta$ and $\alpha \neq \beta$.

Boolean constraint satisfaction problems

A *Boolean relation* R of arity r is a subset of $\{0,1\}^r$. A *constraint* $R(x_1, \dots, x_r)$ is an application of the relation R to a tuple $(x_1, \dots, x_r)$ of not necessarily distinct (Boolean) variables, we also call it an *R-constraint.* A *constraint language* Γ is a set of relations; here all constraint languages are finite and contain only Boolean relations. A formula $\mathcal{F}$ over Γ is a conjunction of constraint applications of relations from Γ. We use $V(\mathcal{F})$ to denote the set of variables of a formula $\mathcal{F}$. A *(truth) assignment* is a mapping $\phi : V \to \{0,1\}$ where V is a set of Boolean variables. An assignment ϕ satisfies a constraint $R(x_1, \dots, x_r)$ if $(\phi(x_1), \dots, \phi(x_r)) \in R$. An assignment ϕ to the variables of a formula $\mathcal{F}$ satisfies the formula if it satisfies each constraint of $\mathcal{F}$. A formula is *satisfiable* if it has at least one satisfying assignment. The *weight* of an assignment is the number of true variables, i.e., the number of variables that are assigned 1.

Remark. Formulas over Γ can be defined as pairs (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 $\overline{s}_i = (x_{i,1}, \dots, x_{i,r_i})$ is an r_i-tuple of variables from V (the *constraint scope*) and $R_i \subseteq \{0,1\}^{r_i}$ is the *constraint relation* ($R_i \in \Gamma$). A minor difference with our chosen definition is that this one implicitly allows free variables that do not occur in any constraint. All our results work also with free variables, but we do not require them.

The classic constraint satisfaction problem SAT(Γ) is that of deciding satisfiability of formulas over the constraint language Γ:

SAT(Γ)

Input: A formula $\mathcal{F}$ over Γ.

Task: Decide whether $\mathcal{F}$ is satisfiable.

Apart from SAT(Γ) a number of optimization variants are studied, most commonly

Max SAT(Γ) maximizing the number of satisfied constraints,

Min UnSAT(Γ) minimizing the number of dissatisfied constraints, and

Max Ones SAT(Γ) maximizing the number of true variables, or

Min Ones SAT(Γ) minimizing the number of true variables in a satisfying assignment.

The approximation properties of these four types of problems have been classified by Khanna et al. [75]; we will present results from [75] in Chapters 6 and 7, when considering Min Ones SAT(Γ) and Max Ones SAT(Γ), respectively.

Examples 5.1. Max SAT($\{(x \neq y)\}$) expresses MAX CUT, Min UnSAT($\{(x \neq y)\}$) expresses EDGE BIPARTIZATION (i.e., deleting edges to make a graph bipartite), Max Ones SAT($\{(\neg x \vee \neg y)\}$) expresses INDEPENDENT SET, and Min Ones SAT($\{(x \vee y)\}$) expresses VERTEX COVER.

In 1978 Schaefer gave a complexity dichotomy into polynomial and NP-complete cases of SAT(Γ) problems over all finite constraint languages Γ. Before stating the dichotomy result, let us recall various standard properties of Boolean relations (cf. [36]):

Definition 5.2. Let R be a Boolean relation.

Zero-valid R is *zero-valid* if $(0, \dots, 0) \in R$.

One-valid R is *one-valid* if $(1, \dots, 1) \in R$.

Strongly zero-valid R is *strongly zero-valid* if it contains all tuples (of matching arity) that have at most one 1.

Horn R is *Horn* (or *weakly negative*) if it can be expressed as a conjunction of clauses each containing at most one positive literal.

Anti-Horn R is *anti-Horn* (or *weakly positive*) if it can be expressed as a conjunction of clauses each containing at most one negated literal.

Bijunctive R is *bijunctive* if it can be expressed as a conjunction of constraints each being the disjunction of at most two literals.

Affine R is *affine* if it can be expressed as a conjunction of constraints of the form $(x_1 \oplus \ldots \oplus x_t = b)$, where $b \in \{0, 1\}$ and addition is modulo 2. The number of tuples in any affine relation is an integer power of 2.

Width-2 affine R is *width-2 affine* if it can be expressed as a conjunction of equality $(x = y)$ and disequality $(x \neq y)$ constraints as well as positive and negative assignments, i.e., (x) respectively $(\neg x)$.

IHS-B− R is *IHS-B−* (*implicative hitting set bounded−*) if it can be expressed as a conjunction of positive assignments (x), implications $(x \to y)$, and negative clauses $(\neg x_1 \vee \ldots \vee \neg x_n)$, for $n \geq 1$.

IHS-B+ R is *IHS-B+* (*implicative hitting set bounded+*) if it can be expressed by negative assignments ($\neg x$), implications ($x \to y$), and positive clauses ($x_1 \vee \ldots \vee x_n$), for $n \geq 1$.

IHS-B R is *IHS-B* if it is either IHS-B+ or IHS-B–.

These and other properties are extended to properties of constraint languages Γ, by saying that Γ is zero-valid (one-valid, Horn, ...) if this holds for every $R \in \Gamma$; one exception: A constraint language Γ is IHS-B if all relations in Γ are IHS-B+ or all relations in Γ are IHS-B–, i.e., if Γ is IHS-B+ or IHS-B–.

Remark. Note that the properties defined above only require that the relation *can* be implemented in the described way, not that they are given in this form. E.g., bijunctive relations are not necessarily binary, but their expressive power is limited to that of 2-SAT formulas: $(w \wedge x) \vee (y \wedge z)$ is bijunctive but $(w \wedge x) \vee (y = z)$ is not bijunctive.

Schaefer's dichotomy theorem

Now we can state Schaefer's dichotomy theorem.

Theorem 5.3 (Schaefer [107]). *Let Γ be a set of Boolean relations. Then* SAT(Γ) *is in* P *if Γ is zero-valid, one-valid, Horn, anti-Horn, bijunctive, or affine. Otherwise,* SAT(Γ) *is* NP*-complete.*

The polynomial cases for SAT(Γ) are well-known and mostly straightforward, it is the proof for the NP-completeness of the remaining cases which is interesting. It is shown that relations which are Horn, anti-Horn, bijunctive, or affine have certain so-called *closure properties*. As an example, each relation R is Horn if and only if it is closed under conjunction (AND), i.e., for any two tuples $\alpha, \beta \in R$ the relation must also contain $\alpha \wedge \beta$. Thus, if the constraint language Γ is not Horn, it must contain a relation R which is not Horn and therefore not closed under conjunction. Hence, R must contain tuples α and β such that $\alpha \wedge \beta \notin R$, we call them *witness tuples* or *witnesses.* Using witnesses against all six known polynomial cases (including two relations which are not zero-valid and not one-valid, respectively) permits a reduction from 3-SAT by giving implementations of the four different 3-SAT clauses, proving NP-completeness (membership in NP is trivial).

The following example shows how to implement relations from witness tuples. Afterwards we will introduce the concepts of closure properties and witnesses in more detail.

Example 5.4. We consider the constraint language $\Gamma = \{(x), (\neg x), R\}$ where R is an 8-ary relation that is not Horn. Let us first pick any two tuples $\alpha, \beta \in R$ witnessing that R is not closed under conjunction, e.g., $\alpha = (1,0,0,0,1,1,0,1)$ and $\beta = (1,1,0,1,0,1,0,0)$ with $\alpha \wedge \beta \notin R$. Thus we have

$$\begin{array}{lccccccccl} \alpha & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \in R \\ \beta & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & \in R \\ \hline \alpha \wedge \beta & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \notin R \end{array}$$

We implement a new relation R' on two variables, by

$$R'(x,y) = (\exists w,z) : (\neg w) \wedge (z) \wedge R(z,x,w,x,y,z,w,y).$$

Thus we get that $(0,1),(1,0) \in R'$ whereas $(0,0) \notin R'$; these tuples correspond to α, β, and $\alpha \wedge \beta$, respectively. Hence, depending on whether $(1,1,0,1,1,1,0,1)$ is contained in R, the relation R' includes or excludes $(1,1)$; we implemented $(x \vee y)$ or $(x \neq y)$, respectively, depending on R.

The underlying strategy is to place variables according to the entries in the witness tuples, e.g., we have placed x where α is 0 and β is 1. In that way, we transfer the knowledge about the witness tuples to inclusion or exclusion of tuples in the new relation. Note that this approach does not require to actually know the relation or the tuples, however, some case distinctions may be required: For example, when we place x in all positions where α is zero and β is one without knowing α and β, we have to argue that such positions exist or discuss also the case when there are no such positions.

Closure properties

A *Boolean function* of arity r is a mapping $f : \{0,1\}^r \to \{0,1\}$. A *clone* is a class C of functions which is closed under superposition (also called composition) and which contains all projections; here clones are classes of Boolean functions:

Projections The class C contains all projections $\pi_i^n : \{0,1\}^n \to \{0,1\}$, which are defined by $\pi_i^n(x_1,\dots,x_n) = x_i$.

Superposition For any functions $f, g_1, \dots, g_m \in C$, where f is m-ary and the g_i are n-ary, C contains also the n-ary function h given by

$$h(x_1,\dots,x_n) = f(g_1(x_1,\dots,x_n),\dots,g_m(x_1,\dots,x_n)).$$

The clones of Boolean functions form a lattice under set inclusion; a complete description of this lattice was given by Emil Post in 1941 [101], it is therefore named *Post's lattice.*

A function f *preserves* a relation R (or R is *invariant* or *closed* under f) if coordinate-wise application of f to any tuples of R gives a tuple in R: For any t tuples $\alpha_1,\dots,\alpha_t \in R$ we have

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

We also say that f is a *polymorphism* of R. Consider the following example:

Example 5.5. Let $f(a,b,c) = (a \wedge b) \vee (a \wedge c) \vee (b \wedge c)$ and let a relation R be defined by $R(w,x,y,z) = (w \wedge x) \vee (y = z)$. Applying f to the tuples $(0,0,0,0)$, $(0,1,1,1)$, and $(1,1,1,0) \in R$ creates the tuple $(0,1,1,0) \notin R$:

a	0	0	0	0	1	1	1	1
b	0	0	1	1	0	0	1	1
c	0	1	0	1	0	1	0	1
$f(a,b,c)$	0	0	0	1	0	1	1	1

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

Intuitively, the application of f to the tuples is done by writing the tuples as a matrix and applying f on each column. Observe that since $(0,1,1,0) \notin R$ we know that R is not invariant under f.

For any set of relations Γ, the set $\mathrm{Pol}(\Gamma)$ contains all functions that preserve all relations in Γ, i.e., all polymorphisms of Γ. For each Γ, its set of polymorphisms $\mathrm{Pol}(\Gamma)$ is a clone and each clone can be defined as $\mathrm{Pol}(\Gamma)$ for some Γ (see [84, 100]).

Let us mention the defining closure properties for some of the Boolean relations. A relation R is Horn if and only if it is invariant under conjunction (AND), e.g., for any two tuples $\alpha, \beta \in R$ also $\alpha \wedge \beta$ is contained in R. A relation is anti-Horn if and only if it is invariant under disjunction. A relation R is affine if and only if it is invariant under ternary XOR, i.e., $\alpha \oplus \beta \oplus \gamma \in R$ for any tuples $\alpha, \beta, \gamma \in R$. A relation R is bijunctive if and only if it is invariant under majority, i.e., for any tuples $\alpha, \beta, \gamma \in R$ the tuple $((\alpha \wedge \beta) \vee (\alpha \wedge \gamma) \vee (\beta \wedge \gamma))$ is contained in R (see [21] or [38] for a characterization of Boolean co-clones).

Given a relation R which is not invariant under the function $f : \{0,1\}^r \to \{0,1\}$ there must be tuples $\alpha_1, \ldots, \alpha_r \in R$ such that f applied to these tuples creates a tuple which is not in R; we call such tuples *witness tuples*. Consider again Example 5.5, it shows that the relation given by $(w \wedge x) \vee (y = z)$ is not closed under majority (i.e., $f(a,b,c) = (a \wedge b) \vee (a \wedge c) \vee (b \wedge c)$) and therefore that it is not bijunctive. Similarly, a constraint language which does not have a certain closure property must contain a relation for which we find the corresponding witness tuples. Given such tuples, one may aim to implement relations which permit a reduction from some hard problem.

Implementation of relations

A constraint language Γ (existentially) implements a relation R if

$$R(x_1, \ldots, x_r) \equiv \exists X : \bigwedge_i R_i(x_{i1}, \ldots, x_{it}),$$

where each $R_i \in \Gamma$ and using variables $X \cup \{x_1, \ldots, x_r\}$. Sets of relations closed under existential implementation are called *co-clones*. The smallest co-clone containing Γ is denoted by $\langle\Gamma\rangle$; we also say that Γ *generates* $\langle\Gamma\rangle$. The types of Boolean relations, introduced in Definition 5.2, are such co-clones.

The set $\mathrm{Inv}(F)$ contains all relations that are invariant under each function in F. For each set of Boolean functions F, the set $\mathrm{Inv}(F)$ is a co-clone and each co-clone can be defined as $\mathrm{Inv}(F)$ for some set F of Boolean functions. Like the clones of Boolean functions, co-clones form a lattice under set inclusion; infimum and supremum of two co-clones A and B are given by $A \cap B$ and $\langle A \cup B\rangle$, respectively (see [84, 100]).

We will later introduce restricted forms of implementations more suitable for the study of problems addressing the number of true variables, which we will use in this thesis. For now, let us mention the strong correspondence between co-clones of Boolean relations and the clones of Boolean functions (their polymorphisms).

The Galois connection

There is a well investigated *Galois connection* between clones and co-clones (cf. [84, 100]). In particular it is known that for any sets of relations Γ_1 and Γ_2 we have

$$\langle\Gamma_1\rangle \subseteq \langle\Gamma_2\rangle \Leftrightarrow \mathrm{Pol}(\Gamma_2) \subseteq \mathrm{Pol}(\Gamma_1)$$

and that $\langle\Gamma\rangle = \mathrm{Inv}(\mathrm{Pol}(\Gamma))$ for any set of relations Γ, i.e., $\langle\cdot\rangle = \mathrm{Inv}(\mathrm{Pol}(\cdot))$. It implies that the lattices of clones and co-clones are dually isomorphic, with Inv and Pol being the dual isomorphisms. Let us briefly mention its impact on the polynomial-time reducibility between SAT(Γ) problems, starting with a theorem due to Jeavons [71].

Theorem 5.6 ([71])**.** *Let Γ_1 and Γ_2 be finite sets of relations such that $\langle\Gamma_1\rangle \subseteq \langle\Gamma_2\rangle$. Then* SAT($\Gamma_1$) *is polynomial-time reducible to* SAT(Γ_2)*.*

Using the Galois connection one obtains the statement that $\mathrm{Pol}(\Gamma_2) \subseteq \mathrm{Pol}(\Gamma_1)$ implies SAT(Γ_1) being polynomial-time reducible to SAT(Γ_2) (thus $\mathrm{Pol}(\Gamma_2) = \mathrm{Pol}(\Gamma_1)$ implies that SAT(Γ_1) and SAT(Γ_2) are polynomial-time equivalent). Using Post's classification of Boolean clones [101] (*Post's lattice*) this already implies Schaefer's dichotomy (cf. [40]).

The Galois connection is an invaluable tool for proving properties and dichotomies of Boolean relations. We will mainly use it for the one-to-one correspondence between co-clones and the matching clones of polymorphisms. Characterizations of the Boolean co-clones were given by Creignou et al. [38] as well as by Böhler et al. [21].

To prove upper bounds we may use the implementation and closure properties, e.g., to design a polynomial kernelization. We will frequently use results of Creignou et al. [38], namely so-called *plain bases* for some of the co-clones, generating the co-clone without requiring extra (i.e., existentially quantified) variables.

For the lower bounds we will use witness tuples against the closure properties, e.g., to construct gadgets for reductions. Another frequent use is to first consider special cases, e.g., that all relations are closed under conjunction (Horn) or disjunction (anti-Horn). The remaining (and possibly most involved) case may then be made easier by using witness tuples, e.g., of relations that are not closed under conjunction or disjunction, respectively.

Many of the results that are presented in Chapters 6, 7, and 8 of this thesis do not follow the lattice of Boolean co-clones, i.e., there are constraint languages which generate the same co-clone (under existential implementation) but which have different kernelization properties. The reason is that the considered problems address the number of true variables in satisfying assignments, but existential implementation of relations does not preserve this. Thus we use restricted notions of polymorphisms and implementations, which are more fine-grained than the co-clone lattice. We conclude the chapter on Boolean CSPs by introducing so-called *partial polymorphisms* and *frozen implementations*.

Partial polymorphisms

Let R be a Boolean relation of arity r. A t-ary *partial polymorphism* of R is a partially defined function $f : \{0,1\}^t \to \{0,1\}$ such that for any t tuples $\alpha_1, \ldots, \alpha_t \in R$, such that $f(\alpha_1(i), \dots, \alpha_t(i))$ is defined for every $i \in [r]$, we have

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

Note the crucial difference compared to the regular polymorphisms: The partial polymorphism must be defined on all tuples $(\alpha_1(i), \dots, \alpha_t(i))$, obtained by taking (in order) the ith entry of each tuple $\alpha_1, \dots, \alpha_t$; we then say that f *applies to* $\alpha_1, \dots, \alpha_t$. Let us stress again that for R to be *invariant under* f, it must contain all tuples created from application of f to tuples from R *to which* f *applies*. Thus a relation may be invariant under some partial polymorphism by not having a choice of tuples to which the partial polymorphism applies.

Where convenient we present 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. (Similar to the complete list of function values for a Boolean function of the same arity.)

Example 5.7. Let f and f' be ternary partial polymorphisms given by the following matrices:

$$\begin{array}{c|ccccc} a & 0 & 0 & 0 & 1 & 1 \\ b & 0 & 0 & 1 & 0 & 1 \\ c & 0 & 1 & 0 & 0 & 1 \\ \hline f(a,b,c) & 0 & 0 & 0 & 0 & 1 \end{array} \qquad \begin{array}{c|cccccc} a & 0 & 0 & 0 & 0 & 1 & 1 \\ b & 0 & 0 & 1 & 1 & 0 & 1 \\ c & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline f'(a,b,c) & 0 & 0 & 0 & 1 & 0 & 1 \end{array}$$

Let a relation R be defined by $R(w,x,y,z) = (w \wedge x) \vee (y = z)$ and consider the tuples $(0,0,0,0)$, $(0,1,1,1)$, and $(1,1,1,0) \in R$. The partial polymorphism f does not apply to these tuples, but f' applies and creates the tuple $(0,1,1,0) \notin R$:

$$\begin{array}{ccccc} 0 & 0 & 0 & 0 & \in R \\ 0 & 1 & 1 & 1 & \in R \\ 1 & 1 & 1 & 0 & \in R \\ \hline 0 & 1 & 1 & 0 & \notin R \end{array}$$

Thus R is not invariant under f'. Note that the three considered tuples to which f does not apply do not imply that R is invariant or not invariant under f. To check whether R is preserved by f one needs to check for all possible choices of three tuples from R whether f applies and whether it generates a tuple that is not in R.

Restricted implementations

Another way to obtain and describe more fine-grained results is to consider weaker notions of implementations. One example are the *partial co-clones* generated by implementations without allowing existential quantification, i.e., $R(x_1, \dots, x_r) = \mathcal{F}$ where $\mathcal{F}$ is a formula over Γ on variables $\{x_1, \dots, x_r\}$. Partial co-clones give a refinement of the co-clone lattice. There is a Galois connection between partial co-clones and the clones of partial polymorphisms; however, there are uncountably many partial-clones and their structure is very complicated (see [84]).

An intermediate concept of implementation was introduced by Nordh and Zanuttini [96]. So-called *frozen (Boolean) co-clones* are generated by permitting existential quantification only for constants: Let $\mathcal{F}$ be a formula and x a variable of $\mathcal{F}$. Then x is said to be *frozen* in $\mathcal{F}$ if x takes the same value in every satisfying assignment of $\mathcal{F}$. Further, let Γ be a set of relations, and R an r-ary relation. Then Γ *freezingly implements* R if there is a formula $\mathcal{F}$ over $\Gamma \cup \{=\}$ such that $R(x_1, \dots, x_r) \equiv \exists X \mathcal{F}$, where $\mathcal{F}$ uses variables $X \cup \{x_1, \dots, x_r\}$ only, and all variables in X are frozen in $\mathcal{F}$. Sets of relations closed under frozen implementation are called *frozen co-clones* and $\langle \Gamma \rangle_{fr}$ denotes the smallest frozen co-clone containing Γ. Frozen co-clones refine co-clones and are refined by partial co-clones. Nordh and Zanuttini [96] also introduce a corresponding notion of polymorphisms, so-called *frozen partial polymorphisms*, and prove a Galois connection between frozen co-clones and clones of frozen partial polymorphisms.

If only relations of Γ are used, then we have a *frozen implementation without equality.* This will be our standard notion of implementation, and as such is shortened to simply "implements". Such an implementation is used in Example 5.4; it uses existentially quantified variables, but their assignments are frozen to true or false.

Summary

Let us repeat some of the crucial aspects of this introduction, which will appear frequently in the following chapters, and add some remarks on their usage:

- Sets of Boolean relations can be characterized by polymorphisms and partial polymorphisms (functions and partial functions). There is a one-to-one correspondence between closed sets of Boolean functions (clones) and closed sets of Boolean relations (co-clones), which are invariant under these functions.

- If a relation is invariant under some (partial) polymorphism, then this may be used to prove other positive properties. For example, zero-valid relations that are invariant under a certain partial polymorphism may be invariant under some other polymorphism, e.g., invariant under conjunction. Thus such relations would also be Horn and we would conclude that they can be implemented in a simple way and construct an algorithm (e.g., a kernelization) using this simpler implementation.

- If a constraint language does not have a certain closure property, then it must contain a relation and corresponding witness tuples that prove this fact. Based on the witness tuples we may then implement other relations in order to construct gadgets for a reduction from some hard problem.

Further reading

For an indepth introduction to Boolean constraint satisfaction problems see the monograph of Creignou, Khanna, and Sudan [36]. An introduction to Post's Lattice and its implications for Constraint Satisfaction Problems was given by Böhler et al. [19, 20]. A current and extensive overview on the field of CSPs can be found in [37]. See also Chen's [30] recent presentation of Schaefer's theorem and of the necessary tools.

6. Min ones constraint satisfaction problems

6.1. Introduction

In this chapter, we classify all Min Ones SAT(Γ) problems into admitting or not admitting a polynomial kernelization depending on the constraint language Γ. Given a formula over Γ, the Min Ones SAT(Γ) problem asks whether there exists a satisfying assignment with at most k true variables. Thus Min Ones SAT(Γ) generalizes well-studied problems such as VERTEX COVER, d-HITTING SET, and graph modification problems.

> **Min Ones SAT(Γ)**
>
> **Input:** A formula $\mathcal{F}$ over Γ and an integer k.
>
> **Parameter:** k.
>
> **Task:** Decide whether $\mathcal{F}$ has a satisfying assignment with at most k true variables.

Depending on the constraint language Γ, Khanna et al. [75] gave a characterization into polynomial cases as well as five further degrees of approximability; we require only the classification into polynomial-time solvable and APX-hard (and NP-complete) cases.

Theorem 6.1 ([75])**.** *Let Γ be a finite set of Boolean relations. If Γ is zero-valid, Horn, or width-2 affine, then* Min Ones SAT(Γ) *is in* P*; otherwise* Min Ones SAT(Γ) *is* APX-*hard.*

It is furthermore known that the problem is fixed-parameter tractable for every finite constraint language Γ, by means of a simple branching algorithm (see Algorithm 2). The algorithm starts with the zero assignment and checks whether this assignment satisfies the formula. If not then any satisfying assignment must set an extra variable to true, in order to satisfy some dissatisfied constraint (possibly implying further necessary changes). Picking any dissatisfied constraint the algorithm branches on all choices of setting an additional variable to true. Clearly one of the branches in the search tree sets the same variables to true as some optimal assignment (if one exists). Otherwise the search will never set more than k variables to true and the number of recursive calls is bounded by the maximum arity of relations in Γ, say d; thus the runtime is $d^k \cdot n^{\mathcal{O}(1)}$.

Our work Following joint work with Magnus Wahlström [81], we give a complete classification of Min Ones SAT(Γ) problems with respect to admittance of polynomial kernelizations. Apart from the hardness dichotomy due to Khanna et al. [75], we distinguish

Algorithm 2 *MinOnesSAT*(Γ)-Solver

Input: A formula $\mathcal{F}$ over Γ, an integer k, and an assignment $\phi : V(\mathcal{F}) \to \{0,1\}$.
Output: A satisfying assignment with at most k true variables if one exists or **false**.

```
if $\mathcal{F}$ is satisfied by $\phi$ then
  return $\phi$
end if
if $|\phi| \geq k$ then
  return false
end if
Pick any constraint of $\mathcal{F}$ that is not satisfied by $\phi$, say $R(x_1, \ldots, x_d)$.
for all $x_i$ with $\phi(x_i) = 0$ do
  Let $\phi' = \phi$ except for $\phi'(x_i) = 1$.
  $MinOnesSAT(\Gamma)$-Solver($\mathcal{F}$, $k$, $\phi'$)
  if the recursive call returned an assignment $\phi''$ then
    return $\phi''$
  end if
end for
return false
```

constraint languages Γ based on mergeability, i.e., a closure property (or partial polymorphism) which we introduce. If Γ is mergeable then we provide a sunflower-based polynomial kernelization; if Γ contains at least one relation which is not mergeable we show that Min Ones SAT(Γ) is either polynomial-time solvable or it does not admit a polynomial kernelization, unless NP ⊆ co-NP/poly.

Structure of this chapter We continue by introducing mergeability in Section 6.2. In Section 6.3 we present the polynomial kernelization for the case that Γ is mergeable. In Section 6.4 we prove the matching lower bound for all constraint languages that contain at least one relation that is not mergeable. We give a summary in Section 6.5.

6.2. Mergeability

The characterization of the dichotomy of the polynomial kernelizability of Min Ones SAT(Γ) given in this chapter centers around a newly introduced property we refer to as *mergeability.* Specifically, we will see that for any finite set Γ of Boolean relations, Min Ones SAT(Γ) admits a polynomial kernelization if either Min Ones SAT(Γ) is in P or every relation $R \in \Gamma$ is mergeable; in every other case, Min Ones SAT(Γ) admits no polynomial kernelization unless NP ⊆ co-NP/poly. In this section, we define this property and give some basic results about it.

Definition 6.2. Let R be a Boolean relation. Given four (not necessarily distinct) tuples $\alpha, \beta, \gamma, \delta \in R$, we say that the *merge operation* applies if $(\alpha \wedge \delta) \leq \beta \leq \alpha$

and $(\beta \wedge \gamma) \leq \delta \leq \gamma$. If so, then *applying* the merge operation to $\alpha, \beta, \gamma, \delta$ produces the tuple $\alpha \wedge (\beta \vee \gamma)$. We say that R is *mergeable* if for any four tuples $\alpha, \beta, \gamma, \delta \in R$ for which the merge operation applies, we have $\alpha \wedge (\beta \vee \gamma) \in R$.

We show some basic results about mergeability.

Proposition 6.3. *Let R be a relation of arity r on the Boolean domain. Partition the positions of R into two sets, called the* core *and the* petals*; w.l.o.g. assume that positions 1 through c are the core, and the rest the petals. Let (α_C, α_P), where α_C is a c-ary tuple and α_P an $(r-c)$-ary tuple, denote the tuple whose first c positions are given by α_C, and whose subsequent positions are given by α_P. Consider then the following four tuples:*

$$\begin{aligned} \alpha &= (\alpha_C, \alpha_P) \\ \beta &= (\alpha_C, 0) \\ \gamma &= (\gamma_C, \gamma_P) \\ \delta &= (\gamma_C, 0) \end{aligned}$$

If α through δ are in R, then the merge operation applies, giving us

$$(\alpha_C, \alpha_P \wedge \gamma_P) \in R.$$

Furthermore, for any four tuples to which the merge operation applies, there is a partitioning of the positions into core and petals such that the tuples can be written in the above form.

Proof. It is easy to see that the merge operation applies to tuples α, β, γ, and δ as given in the proposition, and that it creates $(\alpha_C, \alpha_P \wedge \gamma_P)$.

For the converse, let us consider tuples α, β, γ, and δ to which the merge operation applies. Using $(\alpha \wedge \delta) \leq \beta \leq \alpha$ and $(\beta \wedge \gamma) \leq \delta \leq \gamma$ it can be checked that only the following types of positions can occur:

$$\begin{array}{l|ccccccc} \alpha & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ \beta & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \gamma & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ \delta & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline \alpha \wedge (\beta \vee \gamma) & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{array}$$

Taking positions of the first three types for the core C and the remaining positions for the petals P completes the proof. □

As a conclusion of Proposition 6.3 we may equivalently express mergeability as the following partial polymorphism (in matrix form):

$$\begin{array}{ccccccc} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{array}$$

The following proposition follows immediately from standard properties of partial polymorphisms.

Proposition 6.4. *Mergeability is preserved by assignment and identification of variables, i.e., if R is mergeable, then so is any relation produced from R by these operations. Further, any relation implementable by mergeable relations is mergeable.*

Next, we show what mergeability implies for a zero-valid relation.

Lemma 6.5. *Any zero-valid relation R which is mergeable is also IHS-B–, and can therefore be implemented using negative clauses and implications.*

Proof. We show that $\alpha \wedge (\beta \vee \gamma) \in R$ for all tuples $\alpha, \beta, \gamma \in R$. First, for any two tuples $\alpha, \beta \in R$, we can apply the merge operation to the tuples α, 0, β, 0, to show that $\alpha \wedge \beta \in R$ (as R is zero-valid). It can then be checked that the operation applies to the tuples α, $(\alpha \wedge \beta)$, $(\alpha \wedge \gamma)$, and $(\alpha \wedge \beta \wedge \gamma)$, and that this implies $\alpha \wedge (\beta \vee \gamma) \in R$. Thus R is IHS-B– and zero-valid, which implies that it can be implemented by negative clauses and implications, according to [38]. $\square$

Note that by Proposition 6.4, this shows that the only zero-valid relations that can be produced from a mergeable relation by assigning or identifying variables are IHS-B–. However, this still leaves room for other positive examples.

Examples 6.6. As basic examples, positive and negative clauses, as well as implications are mergeable. Thus, the same holds for anything implementable by such relations, such as monotone or antimonotone relations (i.e., $\alpha \in R$ implies $\beta \in R$ for all $\beta \geq \alpha$ resp. $\beta \leq \alpha$). A further positive example is the ternary ODD relation ($x \oplus y \oplus z = 1$), but not the ternary EVEN or 4-ary ODD relations.

6.3. Characterization I: Polynomial kernelization

In this section we present a polynomial kernelization for the Min Ones SAT(Γ) problem when all relations in Γ are mergeable. At the core of our kernelization we will apply the sunflower lemma for tuples to find sunflowers among the constraints of a given formula. Mergeability of the relations in Γ will then ensure that we can simplify the constraints of the sunflower. The according replacements will gradually permit us to set some of the variables to zero (false), until only relatively few variables remain. For some intuition it may be helpful to remember the sunflower-based kernelization for d-HITTING SET from Section 2.2; note however, that general mergeable relations are much harder to handle than positive clauses.

Before using these more involved reductions, however, we are interested in identifying variables that we can set to false without restricting the assignments to other variables. Note that, since the task is to find an assignment with at most k true variables, this is a safe reduction rule.

Zero-safe variables A relation R (or an R-constraint) is *zero-closed on position i*, if for each $(\alpha_1, \dots, \alpha_{i-1}, \alpha_i, \alpha_{i+1}, \dots, \alpha_r) \in R$ we have $(\alpha_1, \dots, \alpha_{i-1}, 0, \alpha_{i+1}, \dots, \alpha_r) \in R$. This exactly captures the fact that setting the variable in position i to false does not make the constraint unsatisfied (if it was satisfied before). A variable is said to be *zero-safe*, if it only occurs in zero-closed positions of constraints; these variables can be set to false without harm.

Non-zero-closed cores A relation R (or an R-constraint) is *non-zero-closed* if it has no zero-closed positions. Similarly, the *non-zero-closed core* is the projection of R onto all its non-zero-closed positions. Equivalently it can be obtained by forcing $x_i = 0$ for all zero-closed positions i.

Non-zero-closed cores are the central concept for making many variables zero-safe: Any variable is zero-safe if and only if it does not occur in a non-zero-closed core. In general, relations do not contain zero-closed positions. However, our later sunflower-based reductions will be aimed at replacing constraints by ones that have smaller non-zero-closed cores. This will increase the number of zero-safe variables. In order to do so we define the *zero-closure* and the *sunflower restriction* of a constraint or relation.

Definition 6.7. Let R be an r-ary relation. We define two functions Δ and ∇:

$\boldsymbol{\Delta_P(R)}$**:** the *zero-closure of R on positions* $P \subseteq \{1, \dots, r\}$ is the smallest superset of R that is zero-closed on all positions $i \in P$.

$\boldsymbol{\nabla_C(R)}$**:** the *sunflower restriction of R with core* $C \subseteq \{1, \dots, r\}$ is the relation given by $R(x_1, \dots, x_r) \wedge R(x'_1, \dots, x'_r)$ where

$$x'_i = \begin{cases} x_i & \text{if } i \in C, \\ 0 & \text{if } i \notin C. \end{cases}$$

Remark. The zero-closure $\Delta_P(R)$ of a relation R can be easily obtained: For each tuple $(\alpha_1, \dots, \alpha_r)$ and any subset $P' \subseteq P$ add the tuple $(\alpha'_1, \dots, \alpha'_r)$ to $\Delta_P(R)$, where

$$\alpha'_i = \begin{cases} \alpha_i & \text{if } i \notin P', \\ 0 & \text{if } i \in P'. \end{cases}$$

This can be easily seen to satisfy the definition for the tuples that are in R, and we observe that no further tuples need to be added.

Remark. A sunflower restriction with core C forces all variables in the core positions to take values such that all variables in the other (the petal) positions *can* be set to false. This restriction will occur when we find sunflowers among the constraints of a formula.

Note that when we are able to replace some R-constraint by its zero-closure $\Delta_P(R)$ and P contains at least one non-zero-closed position of R, then $\Delta_P(R)$ has a smaller non-zero-closed core (i.e., less non-zero-closed positions). In fact, we will be able to replace (carefully chosen) sets of R-constraints by the corresponding zero-closures, such that the introduced constraints will have identical non-zero-closed cores (i.e., they have the same variables in their non-zero-closed positions). Before showing how to arrive at such replacements we first introduce a measure that will capture our progress.

Definition 6.8. Let $\mathcal{F}$ be a formula and let R be a relation of arity r that is zero-closed on positions P. We define $\mathcal{C}(\mathcal{F}, R)$ as the set of all tuples $(x_{i_1}, \ldots, x_{i_t})$ such that $R'(x_{i_1}, \ldots, x_{i_t})$ is the non-zero-closed core of an R-constraint in $\mathcal{F}$; the relation R' is R projected to its non-zero-closed positions.

Equivalently, let $\{i_1, \ldots, i_t\} = \{1, \ldots, r\} \setminus P$ and $i_1 < \ldots < i_t$ then $\mathcal{C}(\mathcal{F}, R)$ contains the tuple $(x_{i_1}, \ldots, x_{i_t})$ for each R-constraint $R(x_1, \ldots, x_r)$ in $\mathcal{F}$.

As a last preliminary step towards our kernelization we require two technical lemmas that address the issue of how to simplify constraints which form a sunflower. Notice that when $k + 1$ R-constraints of a formula $\mathcal{F}$ form a sunflower with core C and petal positions P then any satisfying assignment for $\mathcal{F}$, that has at most k true variables, must assign false to all variables in the petal positions of at least one constraint. Thus any such assignment satisfies the sunflower restrictions of the constraints (this will be made more formal in the proof of Theorem 6.12). Hence the constraints of the sunflower may be replaced by the corresponding sunflower restrictions with core C. The following two lemmas permit us to use a better implementation (in the sense of possibly smaller non-zero-closed cores) when the relation is mergeable and ensure as well that the introduced constraints are mergeable.

Lemma 6.9. *Let R be a mergeable relation and let $C \cup P$ be a partition of its positions into core and petals. There is an implementation of the corresponding sunflower restriction $\nabla_C(R)$ using its zero-closure $\Delta_P(\nabla_C(R))$ (on the petal positions) and implications.*

Proof. By Proposition 6.4, $\nabla_C(R)$ must be mergeable. The same is true for the zero-valid constraint on the variables in the petal positions induced by assigning any set of values to the variables in the core positions. Thus by Lemma 6.5, such petal constraints each have an implementation using negative clauses and implications. For any tuple σ in $\nabla_C(R)$ or $\Delta_P(\nabla_C(R))$, let its *core assignment* be the values it assigns to the positions in C. By definition, we have $\nabla_C(R) \subseteq \Delta_P(\nabla_C(R))$.

Consider now a tuple $\sigma \in \Delta_P(\nabla_C(R)) \setminus \nabla_C(R)$. Assume that σ makes core assignment α_C and petal assignment σ_P; thus there is a matching $\alpha \in \nabla_C(R)$, $\alpha > \sigma$, with an identical core assignment, by definition of the zero-closure. As in Proposition 6.3, write $\alpha = (\alpha_C, \alpha_P)$, and consider the constraint on the petals that is induced by the core assignment α_C. This constraint, say on relation R', contains α_P but not σ_P and we have $\alpha_P > \sigma_P$. The relation R' has an implementation through negative clauses and implications, as discussed above. Note that any negative clause which excludes σ_P also excludes α_P (if the corresponding positions of σ_P are all true then so are the positions of α_P). Thus there is an implication $(y_i \rightarrow y_j)$ which is consistent with R' and which excludes σ_P. Hence $\sigma_i = 1$ and $\sigma_j = 0$ implying $\alpha_i = 1$ and $\alpha_j = 1$.

Let us assume for contradiction that $(y_i \rightarrow y_j)$ does not hold in $\nabla_C(R)$ in general. Accordingly let $\gamma = (\gamma_C, \gamma_P) \in \nabla_C(R)$ be a tuple which assigns $y_i = 1$, $y_j = 0$. Since $\nabla_C(R)$

is a sunflower restriction, we have that $\beta = (\alpha_C, 0)$ and $\delta = (\gamma_C, 0)$ are contained in $\nabla_C(R)$. By Proposition 6.3, we can now apply the merge operation to tuples α through δ, showing that $(\alpha_C, \alpha_P \wedge \gamma_P)$ is contained in $\nabla_C(R)$. However, this is a tuple with core assignment α_C which assigns $y_i = 1$, $y_j = 0$. This contradicts our choice of $(y_i \to y_j)$ as an implication consistent with the relation R' induced by core assignment α_C. Thus by contradiction, the implication $(y_i \to y_j)$ holds in $\nabla_C(R)$ regardless of the core assignment, and the constraint $(y_i \to y_j)$ can be added to our implementation of $\nabla_C(R)$, removing the tuple σ.

Adding all implications between petal positions which hold in $\nabla_C(R)$ removes from our implementation all tuples which are in $\Delta_P(\nabla_C(R))$ but not in $\nabla_C(R)$. Thus the conjunction of $\Delta_P(\nabla_C(R))$ with all valid implications implements $\nabla_C(R)$. □

Lemma 6.10. *Let R be a mergeable relation and let $C \cup P$ be a partition of its positions into core and petals. Then $\Delta_P(\nabla_C(R))$ is mergeable.*

Proof. Recall that $\Delta_P(\nabla_C(R))$ is the zero-closure on the petal positions of the sunflower restriction of R with core C. We assume for contradiction that $\Delta_P(\nabla_C(R))$ is not mergeable. Then there are four tuples in $\Delta_P(\nabla_C(R))$ such that applying the merge operation on the tuples creates a tuple not in $\Delta_P(\nabla_C(R))$. Let $C' \cup P'$ be the partition of the positions of $\Delta_P(\nabla_C(R))$ into core and petals that is used in this counterexample. Grouping the positions of $\Delta_P(\nabla_C(R))$ in four groups, written in the order $(C' \cap C, C' \cap P, P' \cap C, P' \cap P)$, naming the groups W through Z, the counterexample can be written as follows.

$$\begin{aligned}
(W_1, X_1, Y_1, Z_1) &\in \Delta_P(\nabla_C(R)) && (6.1)\\
(W_1, X_1, 0, 0) &\in \Delta_P(\nabla_C(R)) && (6.2)\\
(W_2, X_2, Y_2, Z_2) &\in \Delta_P(\nabla_C(R)) && (6.3)\\
(W_2, X_2, 0, 0) &\in \Delta_P(\nabla_C(R)) && (6.4)\\
(W_1, X_1, Y_1 \wedge Y_2, Z_1 \wedge Z_2) &\notin \Delta_P(\nabla_C(R)) && (6.5)
\end{aligned}$$

We will derive a contradiction. First, we note that for each equation (6.1)–(6.4), there is a corresponding tuple in $\nabla_C(R)$.

$$\begin{aligned}
(W_1, X_{1a}, Y_1, Z_{1a}) &\in \nabla_C(R) && (6.6)\\
(W_1, X_{1b}, 0, Z') &\in \nabla_C(R) && (6.7)\\
(W_2, X_{2a}, Y_2, Z_{2a}) &\in \nabla_C(R) && (6.8)\\
(W_2, X_{2b}, 0, Z'') &\in \nabla_C(R) && (6.9)
\end{aligned}$$

Here, $X_1 \leq X_{1a}$ and $X_1 \leq X_{1b}$, and likewise for X_2, Z_1, and Z_2. The tuples Z' and Z'' are arbitrary. Using that $\nabla_C(R)$ is a sunflower restriction and mergeable, we can conclude the following.

$$\begin{aligned}
(W_1, 0, 0, 0) &\in \nabla_C(R) && (6.10)\\
(W_2, 0, 0, 0) &\in \nabla_C(R) && (6.11)\\
(W_1, X_{1a} \wedge X_{2a}, Y_1 \wedge Y_2, Z_{1a} \wedge Z_{2a}) &\in \nabla_C(R) && (6.12)
\end{aligned}$$

The first two come from (6.7) and (6.9); the third is produced by a merge operation on (6.6) and (6.8) using these two. Now, the tuples which match W_1 on the W-variables form a zero-valid relation. By Lemma 6.5, this relation is closed under an operation $(\alpha \wedge (\beta \vee \gamma))$. Applying this on the tuples of equations (6.6), (6.7), and (6.12) gives us the following conclusion:

$$(W_1, X_{1a} \wedge (X_{1b} \vee X_{2a}), Y_1 \wedge Y_2, Z_{1a} \wedge (Z' \vee Z_{2a})) \in \nabla_C(R)$$

In particular, this tuple matches $(W_1, X_1, Y_1 \wedge Y_2, Z_1 \wedge Z_2)$ on the W and Y positions, and is greater or equal on each X and Z position. Since R' is the zero-closure of $\nabla_C(R)$ on the X- and Z-variables, we have that R' contains $(W_1, X_1, Y_1 \wedge Y_2, Z_1 \wedge Z_2)$, a contradiction. □

Lemmas 6.9 and 6.10 are the foundation for a sunflower-based kernelization for Min Ones SAT(Γ). They show that the sunflower restriction $\nabla_C(R)$ of some mergeable R-constraint can be implemented using its mergeable zero-closure on the petal positions as well as implications. There are, however, still some obstacles that need to be overcome: Firstly the needed relations $\Delta_P(\nabla_C(R))$ are not necessarily contained in Γ and, secondly, we still need to show that such a replacement does lead to a simplification of the formula. As to the latter point, consider the following example.

Example 6.11. Consider the following mergeable relation:

$$R = \{(0,0,1,0),(0,1,0,0),(0,1,0,1),(1,0,0,0),(1,0,0,1),(1,1,1,0),(1,1,1,1)\}.$$

Observe that its sunflower restriction with core $\{1,2,3\}$ and its matching zero-closure on position 4 are the same relation, i.e., $R = \nabla_{\{1,2,3\}}(R) = \Delta_{\{4\}}(\nabla_{\{1,2,3\}}(R))$ (in fact whenever there is only one petal position then the zero-closure adds no tuples to the sunflower restriction). Thus even in terms of $|\mathcal{C}(\mathcal{F}, R)|$, i.e., the number of non-zero-closed cores of R-constraints, there would be no improvement when replacing a sunflower of R-constraints with core $C = \{1,2,3\}$.

We have seen that there are mergeable relations for which certain sunflower constellations do not lead to an improvement, even in terms of the number of non-zero-closed cores of R-constraints in $\mathcal{F}$, i.e., $|\mathcal{C}(\mathcal{F}, R)|$; indeed we would not even change the formula at all. However, when we ensure that the petal positions contain at least one non-zero-closed position of R, then we will see that we do get fewer and smaller non-zero-closed cores through each replacement. This is facilitated by searching for sunflowers among the tuples of $\mathcal{C}(\mathcal{F}, R)$. Those are then leveraged into a replacement of the R-constraints that contributed these tuples. The following theorem shows this approach in detail.

We point out that for the purpose of this theorem we will avoid the fact that the relations needed for our sunflower replacements are not necessarily contained in Γ, by using a larger constraint language Γ′. We will handle this syntactical problem later, focusing first on obtaining a shorter equivalent formula over Γ′.

Theorem 6.12. *Let Γ be a mergeable constraint language with maximum arity d. Let Γ′ be the constraint language consisting of all mergeable relations of arity at most d.*

Let $\mathcal{F}$ be a formula over Γ and let k be an integer. In polynomial time one can compute a formula $\mathcal{F}'$ over Γ' such that every assignment with at most k true variables satisfies $\mathcal{F}$ if and only if it satisfies $\mathcal{F}'$ and, furthermore, such that $|\mathcal{C}(\mathcal{F}', R)| \leq (d!)^2 k^d \in \mathcal{O}(k^d)$, for every non-zero-valid relation $R \in \Gamma'$.

Proof. We begin with constructing $\mathcal{F}'$, starting from $\mathcal{F}' = \mathcal{F}$. While $|\mathcal{C}(\mathcal{F}', R)| > k^d (d!)^2$ for any non-zero-valid relation R in Γ', search for a sunflower of cardinality $k + 1$ in $\mathcal{C}(\mathcal{F}', R)$, according to Lemma 2.7. Let C denote the core of the sunflower and apply the following replacement. Remove each R-constraint whose non-zero-closed core matches a tuple of the sunflower and add its sunflower restriction with core C using an implementation according to Lemma 6.9. Repeating this step until $|\mathcal{C}(\mathcal{F}', R)| \leq k^d (d!)^2$ for all non-zero-valid relations R in $\mathcal{F}'$ completes the construction.

Correctness Now, to prove correctness, let us consider a single replacement. We denote the tuples of the sunflower by $(x_1, \ldots, x_c, y_{i1}, \ldots, y_{ip})$, with $i \in \{1, \ldots, k+1\}$, i.e., w.l.o.g. with core $C = \{1, \ldots, c\}$ and petals $P = \{c + 1, \ldots, c + p\}$. Let ϕ be any satisfying assignment with at most k true variables.

We consider an arbitrary tuple $(x_1, \ldots, x_c, y_{i1}, \ldots, y_{ip})$ of the sunflower. There must be a constraint $R(x_1, \ldots, x_c, y_{i1}, \ldots, y_{ip}, z_1, \ldots, z_t)$ whose non-zero-closed core matches the tuple, w.l.o.g. we take the last positions of R to be zero-closed, let Z be those positions. Thus ϕ must satisfy $R(x_1, \ldots, x_c, y_{i1}, \ldots, y_{ip}, 0, \ldots, 0)$, since the z_i are in zero-closed positions. Observe that, by having at most k true variables, ϕ assigns false to all variables $y_{i1}, \ldots, y_{ip}$ for some $i \in \{1, \ldots, k+1\}$. Thus ϕ satisfies also $R(x_1, \ldots, x_c, 0, \ldots, 0)$.

Hence for any constraint $R(x_1, \ldots, x_c, y_{i1}, \ldots, y_{ip}, z_1, \ldots, z_t)$, the assignment ϕ satisfies

$$\begin{aligned}&\nabla_C(R(x_1, \ldots, x_c, y_{i1}, \ldots, y_{ip}, z_1, \ldots, z_t)) = \\ &R(x_1, \ldots, x_c, y_{i1}, \ldots, y_{ip}, z_1, \ldots, z_t) \wedge R(x_1, \ldots, x_c, 0, \ldots, 0)\end{aligned}$$

too. This permits us to replace each R-constraint, whose non-zero-closed core matches a tuple of the sunflower, by an implementation of its sunflower restriction with core C. The implementation uses $\Delta_{P \cup Z}(\nabla_C(R(x_1, \ldots, x_c, y_{i1}, \ldots, y_{ip}, z_1, \ldots, z_t)))$ and implications, according to Lemma 6.9. By Lemma 6.10 the added $\Delta_{P \cup Z}(\nabla_C(R))$-constraints are mergeable, maintaining the invariant that all constraints in $\mathcal{F}'$ are mergeable.

Efficiency To establish that the construction can be performed efficiently, i.e., in time polynomial in the size of $\mathcal{F}$, we use as a measure of $\mathcal{F}'$ the sum of $|\mathcal{C}(\mathcal{F}', R)|$ over all relations R occurring in $\mathcal{F}'$. First, let us observe that, initially, this measure is bounded by the size of $\mathcal{F}$ since each R-constraint of $\mathcal{F}'$ contributes at most one tuple to the corresponding set $\mathcal{C}(\mathcal{F}', R)$ (recall that we start with $\mathcal{F}' = \mathcal{F}$). Consider again the replacement made in each step: All R-constraints matching one of the tuples of the sunflower are replaced by an implementation using $\hat{R} = \Delta_{P \cup Z}(\nabla_C(R))$ and implications. It is crucial to observe that all added constraints contribute the same tuple to $\mathcal{C}(\mathcal{F}', \hat{R})$, consisting only of variables with positions in C. This is caused by the application of the zero closure Δ on all positions but those in C. Hence the $k+1$ tuples of the sunflower are removed, as all matching R-constraints are replaced, and only one new tuple is added to the set $\mathcal{C}(\mathcal{F}', \hat{R})$. This decreases the measure, implying that the modification step is applied at most a number of times which is polynomial in the size of $\mathcal{F}$.

Finally let us express the fact that each iteration of the replacement can be done efficiently. The set $\mathcal{C}(\mathcal{F}', R)$ can be generated in one pass over the formula and since the arity is bounded by d there are only a constant number of relations. The applications of Lemma 2.7 to find a sunflower among the tuples of the sets $\mathcal{C}(\mathcal{F}', R)$ take time polynomial in $|\mathcal{F}|$, since the size of $\mathcal{C}(\mathcal{F}', R)$ is bounded by $\mathcal{O}(|\mathcal{F}|)$. Observe that the size of $\mathcal{F}'$ is bounded by a polynomial in $|\mathcal{F}|$ at all times, since there are only a polynomial number of possible constraints of arity at most d on the variables of $\mathcal{F}$. □

Now we are able to derive a polynomial kernelization for Min Ones SAT(Γ). For a given instance $(\mathcal{F}, k)$, it will first generate an equivalent formula $\mathcal{F}'$ according to Theorem 6.12. However, $\mathcal{F}'$ will not replace $\mathcal{F}$, rather, it allows us to remove variables from $\mathcal{F}$ based on conclusions drawn from $\mathcal{F}'$. This approach avoids the obstacle of a possible lack of expressibility from using only the language Γ, and requires no additional assumptions or annotations to be made.

Theorem 6.13. *Let Γ be a mergeable constraint language. Then* Min Ones SAT(Γ) *admits a polynomial kernelization.*

Proof. Let $(\mathcal{F}, k)$ be an instance of Min Ones SAT(Γ) and let d be the maximum arity of relations in Γ. According to Theorem 6.12, we generate a formula $\mathcal{F}'$, such that assignments with at most k true variables are satisfying for $\mathcal{F}$ if and only if they are satisfying for $\mathcal{F}'$. Moreover, for each non-zero-valid relation R, we have that $|\mathcal{C}(\mathcal{F}', R)|$ is bounded by $\mathcal{O}(k^d)$. Note that constraints of $\mathcal{F}'$ have maximum arity d. We allow the constant 0 (false) to be used for replacing variables; an implementation for this follows at the end of the proof.

First, according to Lemma 6.5, we replace each zero-valid constraint of $\mathcal{F}'$ by an implementation through negative clauses and implications. Next, we address variables that occur only in zero-closed positions of constraints in $\mathcal{F}'$. By definition of zero-closed positions it is immediate that setting such a variable to 0, does not affect the possible assignments for the other variables. By equivalence of $\mathcal{F}$ and $\mathcal{F}'$ with respect to assignments of weight at most k, the same is true for $\mathcal{F}$. We replace all such variables by the constant 0 in $\mathcal{F}$ and $\mathcal{F}'$, maintaining the equivalence with respect to assignments with at most k true variables.

Now, let X be the set of variables that occur in a non-zero-closed position of some non-zero-valid constraint of $\mathcal{F}'$. For each variable $x \in X$ count the number of variables y that are implied by x, i.e., that have to take value 1 if $x = 1$, by implication constraints in $\mathcal{F}'$. If the number of those variables is at least k, then there is no satisfying assignment with at most k true variables for $\mathcal{F}'$ that assigns 1 to x. By equivalence of $\mathcal{F}$ and $\mathcal{F}'$ with respect to such assignments, we replace all occurrences of such a variable x by the constant 0, again maintaining the equivalence property.

Finally we replace all variables $y \in V(\mathcal{F}') \setminus X$, which are not implied by a variable from X in $\mathcal{F}'$, by the constant 0 in $\mathcal{F}$ and $\mathcal{F}'$. Note that such variables y occur only in zero-closed positions and in implications. It can be easily verified that this does not affect satisfiability with respect to assignments of weight at most k. For efficiency of this modification consider the fact that the number of implications in $\mathcal{F}'$ is polynomial in the initial size of $\mathcal{F}$, since there are at most two implications per pair of variables of $\mathcal{F}$. This completes the kernelization.

Bounding the kernel size Now we prove a bound of $\mathcal{O}(k^{d+1})$ on the number of variables in $\mathcal{F}$. First, we observe that all remaining variables of $\mathcal{F}$ must occur in a non-zero-closed position of some constraint of $\mathcal{F}'$. We begin by bounding the number of variables that occur in a non-zero-closed position of some non-zero-valid R-constraint, i.e., the remaining variables of the set X. Observe that such a variable must occur in the corresponding tuple of $\mathcal{C}(\mathcal{F}', R)$. Since there is only a constant number of relations of arity at most d and since $|\mathcal{C}(\mathcal{F}', R)| \in \mathcal{O}(k^d)$, this limits the number of such variables to $\mathcal{O}(k^d)$. For all other variables, their non-zero-closed occurrences must be in implications, since negative clauses are zero-closed on all positions. Thus, these variables must be implied by a variable of X. Since each variable implies at most $k-1$ other variables, we get an overall bound of $\mathcal{O}(k^{d+1})$. Finally, the total size of $\mathcal{F}$ is polynomial for fixed d, since the number of variables is polynomial and the arity of the constraints is bounded by d.

Implementing the constant zero To express the 0-constant, we add $k+1$ new variables $z_1, \ldots, z_{k+1}$. Every constraint with at least one 0 is replaced by $k+1$ copies, each time replacing 0 with a different z_i. Clearly one of the z_i takes value 0 in any assignment with at most k true variables. Hence the original constraints with constant 0 are enforced. Conversely, given a satisfying assignment of weight at most k for the formula before making this replacement, we can easily extend it by assigning 0 to each z_i. This construction does not affect our upper bound on the number of variables. $\square$

We have showed that Min Ones SAT(Γ) admits a polynomial kernelization for any mergeable constraint language Γ. In the following section we will see that, if Min Ones SAT(Γ) is NP-hard, then any non-mergeable relation in Γ suffices to exclude the existence of a polynomial kernelization (assuming NP $\nsubseteq$ co-NP/poly).

6.4. Characterization II: Lower bound

In this section we complete the dichotomy by showing that if Γ is not mergeable, then Min Ones SAT(Γ) is either in P or it is NP-complete and does not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly. Let us recall Khanna et al.'s [75] classification of the polynomial-cases: Min Ones SAT(Γ) is polynomial when Γ is zero-valid, weakly negative (Horn), or width-2 affine; otherwise it is NP-hard.

Our strategy for proving the lower bound is to give a polynomial parameter transformation from a problem that is shown not to admit a polynomial kernelization to Min Ones SAT(Γ). As a source problem we will use a parameterized version of EXACT HITTING SET; the (possibly unusual) parameter is the number m of sets. EXACT HITTING SET has relatively simple constraints but they have unbounded arity.

The first part of this section will establish a kernel lower bound for this problem, following an approach by Dom et al. [46]. Then we will introduce so-called *log-cost selection formulas*, which are the central tool for our lower bound and whose construction requires a non-mergeable relation. These formulas will cope with the fact that we need to reduce constraints of arbitrary arity to constant-arity constraints. Then we show that any non-mergeable Γ (such that Min Ones SAT(Γ) is NP-complete) can implement such selection formulas. We complete the lower bound proof with a polynomial parameter transformation from EXACT HITTING SET to Min Ones SAT(Γ).

Lower bound for Exact Hitting Set

We will now show that EXACT HITTING SET parameterized by the number of edges does not admit a polynomial kernelization unless NP ⊆ co-NP/poly. Given a hypergraph over some universe U the problem asks for a subset S of U that contains exactly one element of each edge:

> **EXACT HITTING SET(m)**
>
> **Input:** A hypergraph $\mathcal{H}$ consisting of m subsets of a universe U of size n.
>
> **Parameter:** m.
>
> **Task:** Decide whether there is a set $S \subseteq U$ such that $|E \cap S| = 1$ for every $E \in \mathcal{H}$.

We follow an approach of Dom et al. [46] who showed that UNIQUE COVERAGE does not admit a polynomial kernel by using a *colored version* of the problem. In a similar way we introduce a problem called Exact CSP($m + n$):

> **Exact CSP($m + n$)**
>
> **Input:** A CSP instance with n variables of arbitrary finite domain, and m constraints Exactly-One($v_{i_1}{=}b_{i_1}, \ldots, v_{i_r}{=}b_{i_r}$) of arbitrary arity, where each v_i is a variable and b_i a value from the respective variable domain.
>
> **Parameter:** $m + n$.
>
> **Task:** Decide whether there is an assignment of a value to every variable that satisfies each constraint (i.e., for each constraint, exactly one statement $v = b$ is true).

We show that Exact CSP($m + n$) admits no polynomial kernelization and give a straightforward reduction to EXACT HITTING SET(m). The proof follows the same lines as the lower bound for UNIQUE COVERAGE in [46, Sec. 4.2], but the construction is somewhat simplified, and the lower bound somewhat stronger (as EXACT HITTING SET(m) is equivalent to a special case of UNIQUE COVERAGE).

Lemma 6.14. Exact CSP($m+n$) *and* EXACT HITTING SET(m) *do not admit polynomial kernelizations unless* NP ⊆ co-NP/poly.

Proof. First, Exact CSP$(m+n)$ is NP-complete (even if all domains have cardinality 2, in which case it is the EXACT SATISFIABILITY problem). Also, the problem can be solved in time $n^m \cdot m^{\mathcal{O}(1)}$: Decide for each constraint the identity of the variable which will satisfy it (but not yet its value). Assuming that a variable v is chosen for a particular constraint, for every statement $(v_i = b_j)$ in the constraint with $v \neq v_i$, remove the value b_j from the domain of the variable v_i, and restrict the domain of v to those values which would satisfy the constraint. Repeat for all constraints, backtracking if necessary; the size of the search tree is at most n^m. Thus, we may assume in our composition algorithm that the number of input instances is bounded by n^m (or else we solve all instances in time polynomial in the total input size).

Assume, then, that there are t input instances. Let n be the maximum number of variables and m the maximum number of constraints; for simplicity of the argument, assume that all input instances have the same numbers of variables and constraints (or else do trivial padding with unary-domain variables or trivially true constraints, such as "variable 1 has exactly one value"). Number the variables from 1 to n and the constraints from 1 to m in each input instance.

Now create the composed instance. First collect all the values of variables numbered i into the domain of a single variable v'_i, say with values (j_1, j_2) signifying "value j_2 in the domain of input instance j_1". Similarly concatenate all constraints numbered i into a single constraint, over these new domain values. Note that values stemming from different instances are different, so that a constraint is hit only once in an intended solution (where all values come from the same instance). We finally need to add constraints to ensure that all variables take values stemming from the same input instance.

For this, assign to each input instance a number from 1 to t as its ID, and write this in binary form. Let $\ell = \lceil \log_2 t \rceil = \mathcal{O}(m \log n)$ be the number of bit levels needed. For each pair of values $(i, i+1)$, $1 \leq i < n$, and each bit level j, $1 \leq j \leq \ell$, add a testing constraint consisting of all values of v_i for which the jth digit of the ID of the originating instance is 1, and all values of v_{i+1} for which the jth digit of the ID is 0. This makes $\mathcal{O}(mn \log n)$ extra edges. If variables v_i and v_j take values from different input instances, then their IDs will differ in some position, which will lead to one of these testing edges being hit twice or not at all. Otherwise, each testing edge is hit exactly once. By transitivity, this forces all variables in the composed instance to take values from the same input instance. This completes the compositionality proof, showing that Exact CSP with parameter $m+n$ admits no polynomial kernelization.

The result for Exact Hitting Set with parameter m follows by a simple reduction. Let the vertices of the hitting set be the individual variable values, create for each variable an edge containing all its values, and retain all constraints as edges. We get an equivalent instance with $m+n$ edges. □

Log-cost selection formulas

Now we need to find a reduction from EXACT HITTING SET(m) to Min Ones SAT(Γ). A central issue here is that EXACT HITTING SET(m) permits constraints of arbitrary arity (i.e., size of the underlying universe) while our constraint languages are finite. Let us briefly recall the classic reduction from SATISFIABILITY to 3-SAT:

Example 6.15 (Reduction from SATISFIABILITY to 3-SAT)**.** The central idea of the reduction is that clauses of arbitrary length, say on n literals: $(\ell_1 \vee \ell_2 \vee \ldots \vee \ell_n)$, can be reduced to a conjunction of 3-CNF clauses by using auxiliary variables:

$$\begin{aligned}(\ell_1 \vee \ell_2 \vee \ldots \vee \ell_n) \quad \Leftrightarrow \quad & (\ell_1 \vee \ell_2 \vee s_1) \wedge (\neg s_1 \vee \ell_3 \vee s_2) \wedge \ldots \wedge \\ & \wedge (\neg s_{n-4} \vee \ell_{n-2} \vee s_{n-3}) \wedge (\neg s_{n-3} \vee \ell_{n-1} \vee \ell_n).\end{aligned}$$

The reduction increases the input size, i.e., the length of the formula, by a linear factor and introduces $n-3$ auxiliary variables.

Using a similar technique for our desired reduction from EXACT HITTING SET(m) to Min Ones SAT(Γ) would introduce two problems: First, when introducing auxiliary variables in this fashion, we have to control the total number of true variables in satisfying assignments. Note that, in the above example, the assignment to the s_i-variables varies greatly depending on the assignment to the literals (e.g., if only ℓ_1 is true, then all s_i must be set to false; if only ℓ_n is true then all s_i must be set to true).

Remark. We could fix the first problem by introducing further auxiliary variables $\tilde{s}_i$ and constraints $(s_i \vee \tilde{s}_i) \wedge (\neg s_i \vee \neg \tilde{s}_i)$ for all i (or directly $(s_i \neq \tilde{s}_i)$). In this way any satisfying assignment would assign true to exactly $n-3$ of the auxiliary variables.

The second problem is more severe: For EXACT HITTING SET(m) we will have to encode, for each of the m edges, the constraint of selecting (exactly) one of its at most $n = |U|$ vertices. However, with the above reduction satisfying assignments may require $\Theta(n)$ true variables. The desired polynomial parameter transformation must make the new parameter have value at most $m^{\mathcal{O}(1)}$, whereas n may be super-polynomial in m. Thus we have to find a more involved way of encoding the edges.

Let us recall the tree-like construction used in Lemma 4.2. There we used auxiliary variables, for intuition arranged like a binary tree, to create a selection of one of the variables in the leaf positions to be set to true. We repeat a similar construction to express a disjunction on n literals:

Example 6.16 (Alternate reduction from SATISFIABILITY to 3-SAT)**.** Consider again a disjunction of n literals: $(\ell_1 \vee \ldots \vee \ell_n)$. Using a tree-like arrangement of auxiliary variables, with variables $s_{i,1}, \ldots, s_{i,2^i}$ on the i-th level, we can replace the clause in the following way:

$$\begin{aligned}(\ell_1 \vee \ell_2 \vee \ldots \vee \ell_n) \quad \Leftrightarrow \quad & (s_{0,1} = 1) \wedge (s_{0,1} \rightarrow (s_{1,1} \vee s_{1,2})) \wedge \\ & \wedge (s_{1,1} \rightarrow (s_{2,1} \vee s_{2,2})) \wedge (s_{1,2} \rightarrow (s_{2,3} \vee s_{2,4})) \wedge \\ & \wedge \ldots \wedge \\ & \wedge (s_{t,1} \rightarrow (\ell_1 \vee \ell_2)) \wedge \ldots \wedge (s_{t,u} \rightarrow (\ell_{n-1} \vee \ell_n)).\end{aligned}$$

In this way we again use $\mathcal{O}(n)$ auxiliary variables, however, every satisfying assignment for $(\ell_1 \vee \ldots \vee \ell_n)$ can be extended to the auxiliary variables by setting exactly one $s_{i,j}$ to true for each $i \in \{0, \ldots, t\}$, where $t \in \mathcal{O}(\log n)$.

We distill this example into a definition, namely that of a *log-cost selection formula.* We need to abstract from the example since we are not guaranteed that $(x \to (y \vee z))$ is contained in Γ. The definition captures the selection process in such a way, that we will be able to reproduce it using any non-mergeable relation R from any constraint language Γ such that Min Ones SAT(Γ) is NP-complete.

Definition 6.17. A *log-cost selection formula* of arity n is a formula on variable sets X and Y, with $|Y| = n$ and $|X| = n^{\mathcal{O}(1)}$, such that there is no solution where $Y = 0$, but for any $y_i \in Y$ there is a solution where $y_i = 1$, $y_j = 0$ for $j \neq i$, and where a fix number $w_n = \mathcal{O}(\log n)$ variables among X are true. Furthermore, there is no solution where fewer than w_n variables among X are true.

Implementing log-cost selection formulas

Over the course of the following three lemmas we will establish that any constraint language Γ, such that Min Ones SAT(Γ) is NP-complete and that contains at least one relation that is not mergeable, can implement a log-cost selection formula. We will use NP-completeness to argue that we may assume that Γ contains (x) and $(\neg x)$, i.e., simple assignment constraints that force a variable to take value one or zero (true or false) as well as the equality constraint. Any non-mergeable relation from Γ can then be used to implement log-cost selection formulas. We begin by showing two fundamental ways of constructing log-cost selection formulas.

Lemma 6.18. *The following types of relations can implement log-cost selection formulas of any arity:*

1. *A ternary relation R_3 such that*

$$\begin{aligned}(0,0,0) &\in R_3\\ (1,1,0) &\in R_3\\ (1,0,1) &\in R_3\\ (1,0,0) &\notin R_3\end{aligned}$$

   together with relations (x) and $(\neg x)$.

2. *A 5-ary relation R_5 such that*

$$\begin{aligned}(1,0,1,1,0) &\in R_5\\ (1,0,0,0,0) &\in R_5\\ (0,1,1,0,1) &\in R_5\\ (0,1,0,0,0) &\in R_5\\ (1,0,1,0,0) &\notin R_5\\ (0,1,1,0,0) &\notin R_5\end{aligned}$$

   together with relations $(x \neq y)$, (x), and $(\neg x)$.

Proof. Let $Y = \{y_1, \ldots, y_n\}$ be the variables over which a log-cost selection formula is requested. We will create "branching trees" over variables $x_{i,j}$ for $0 \leq i \leq \log_2 n$, $1 \leq j \leq 2^i$, as variants of the composition trees used in Chapter 4. Assume that $n = 2^h$ for some integer h; otherwise pad Y with variables forced to be false, as assumed to be possible in both constructions. There are two different constructions:

(1) The first construction is straightforward. Create the variables $x_{i,j}$ and add a constraint $(x_{0,1})$ forcing $x_{0,1}$ to be assigned true. Further, for all i, j with $0 \leq i < h$ and $1 \leq j \leq 2^i$, add a constraint $R_3(x_{i,j}, x_{i+1,2j-1}, x_{i+1,2j})$. Finally, replace variables $x_{h,j}$ by y_j. By the requirements on R_3, for every internal variable $x_{i,j}$, if $x_{i,j}$ is true then one of its children $x_{i+1,2j-1}$ and $x_{i+1,2j}$ must be true. Thus by transitivity, some variable on each level of the branching tree must be true, making $Y = 0$ impossible. Conversely, for any variable y_i on the leaf level, there is a solution where exactly the variables along the path from the root node to y_i are true. Thus $w_n = h = \log_2 n$, i.e., in each solution there are at least $\log_2 n$ true variables, and for each leaf variable there is a solution with $\log_2 n$ true variables that sets it to true (recall Definition 6.17).

(2) The second construction uses the same principle, but the construction is somewhat more involved. Create variables $x_{i,j}$ and a constraint $(x_{0,1})$ as before. In addition, introduce for every $0 \leq i \leq h-1$ two variables l_i, r_i and a constraint $(l_i \neq r_i)$. Now the intuition is that (l_i, r_i) decides whether the path of true variables from the root to a leaf should take a left or a right turn after level i (i.e., the choice made by l_i and r_i is always possible, and the path can only stop in a leaf). Concretely, add for every i, j with $0 \leq i \leq h-1$ and $1 \leq j \leq 2^i$ a constraint $R_5(l_i, r_i, x_{i,j}, x_{i+1,2j-1}, x_{i+1,2j})$. Now for every true variable $x_{i,j}$, it is not allowed that $x_{i+1,2j-1} = x_{i+1,2j} = 0$, while depending on l_i and r_i (and R_5), $(x_{i+1,2j-1} = 1, x_{i+1,2j} = 0)$ or $(x_{i+1,2j-1} = 0, x_{i+1,2j} = 1)$ is allowed. This rules out the case $Y = 0$, while for each set of values of l_i, r_i it is allowed to set among variables $x_{i,j}$ exactly the variables along a path from the root to a leaf y_i to true, and other variables to false. In total, exactly two variables not among Y are true per level in such an assignment, making $w_n = 2h = 2\log_2 n$. □

We will now show that any relation which is not mergeable can be used to construct a relation as in Lemma 6.18. The constructions are based on the concept of a *witness* that some relation R lacks a certain closure property. Recall that a witness is a set of tuples to which a certain closure property would apply but such that the resulting tuple is not contained in the relation, e.g., if some relation R is not closed under conjunction then there are tuples $\alpha, \beta \in R$ such that $\alpha \wedge \beta \notin R$. Using the knowledge that such witnesses exist, we use the approach of Schaefer [107], identifying variables according to their occurrence in the tuples of the witness, to build relations with the properties we need. We begin by showing that we may assume to have assignments (x) and $(\neg x)$ as well as equality available in Γ.

Lemma 6.19. *Let* Γ *be a set of relations such that* Min Ones SAT(Γ) *is* NP*-complete and some* $R \in \Gamma$ *is not mergeable. Then* Min Ones SAT($\Gamma \cup \{(x), (\neg x), (x = y)\}$) *reduces to* Min Ones SAT(Γ) *by a polynomial parameter transformation.*

Proof. Let $(\mathcal{F}, k)$ be an instance of Min Ones SAT$(\Gamma \cup \{(x), (\neg x), (x = y)\})$. It suffices to implement the constraints (x), $(\neg x)$, and $(x = y)$ using only constraints of Γ:

(x) Since Min Ones SAT(Γ) is NP-complete, it contains some relation that is not zero-valid; let $R \in \Gamma$ be such a relation. If R is one-valid, then $R(x, \ldots, x)$ is equivalent to (x) and we are done. Else, let r be the arity of R and let I be a maximal set such that $R(x_1, \ldots, x_r)$ holds for $x_i = 1$ for $i \in I$, $x_i = 0$ else. Identify all x_i, $i \in I$, to a single variable x, and all x_i, $i \notin I$, to a single variable y. This forms a new constraint $R'(x, y)$, where $(1, 0) \in R'(x, y)$ and $(0, 0), (1, 1) \notin R'(x, y)$. Thus R' is either $(x \wedge \neg y)$ or $(x \neq y)$. In the former case we are done; in the latter case, constraints $(x \neq y_i)$ for $1 \leq i \leq k+1$ force $x = 1$ and all $y_i = 0$ in any solution with at most k true variables and thus implement (x).

$(\neg x)$ and $(x = y)$ Let α through δ be tuples that witness that R is not mergeable, i.e., $\alpha, \ldots, \delta \in R$ such that the merge operation applies and $\sigma := \alpha \wedge (\beta \vee \gamma) \notin R$. Notice that $\beta < \sigma < \alpha$, meaning that the positions of R are of four types: those where $\beta < \sigma$, those where $\sigma < \alpha$, and optionally positions which are constant among these tuples, i.e., true in β or false in α. Call these positions C_x, C_y, C_1, and C_0, in the order they were introduced. Note that positions C_x and C_y must exist to distinguish σ from β and α, respectively.

Add a variable c_1 and a constraint (c_1) forcing c_1 to always be assigned true. Place c_1 in all positions C_1, if any, and variables x and y in all positions C_x respectively C_y. Now there are two cases:

(a) If there are no positions C_0, then this creates a constraint $R'(x, y)$ with $(1, 0) \notin R'$ and $(0, 0), (1, 1) \in R'$. Thus $R'(x, y) \wedge R'(y, x)$ implements $(x = y)$. This can be used to implement $(\neg x)$: create k variables y_i and add $(x = y_i)$ for every i. In any solution with at most k true variables, all these variables are false.

(b) Otherwise, if there are positions C_0, then place the variable y in these positions as well, and apply $R'(x, y) \wedge R'(y, x)$ again; the result is either $(x = y)$ or $(x = y = 0)$. In the latter case we place a variable c_0, forced to zero by $(c_0 = c_0 = 0)$, in positions C_0 and implement $(x = y)$ as above. In both cases we then implement $(\neg x)$ as before.

Using these implementations we can create an equivalent formula $\mathcal{F}'$ over Γ on variables $V(\mathcal{F}) \cup \{c_0, c_1\}$. Any satisfying assignment for $\mathcal{F}$ with at most k true variables gives a satisfying assignment for $\mathcal{F}'$ with at most $k+1$ true variables by assigning zero (false) to c_0 and one (true) to c_1. The reverse holds also by simply restricting any satisfying assignment for $\mathcal{F}'$ to the variables of $\mathcal{F}$, reducing the number of true variables by one (since c_1 is always assigned true). □

Thus, for the purpose of proving that Min Ones SAT(Γ) does not admit a polynomial kernel, we may assume Γ to contain (x), $(\neg x)$, and equality. Lower bounds for this case can then be transferred to languages without these constraints. We will now prove the central piece of our lower bound, namely how to construct log-cost selection formulas from any suitable constraint language Γ.

Lemma 6.20. *Let Γ be a non-mergeable constraint language that contains (x) and $(\neg x)$. Then Γ can express a log-cost selection formula of any arity.*

Proof. Let $R \in \Gamma$ be a relation that is not mergeable, and let α through δ be a witness of this, i.e., α through δ are contained in R, the merge operation applies to them, and $\sigma := \alpha \wedge (\beta \vee \gamma) \notin R$. By Proposition 6.3, partition the positions of R into core and petals in a way that agrees with the witness tuples. Group the variables with respect to their values in these four tuples into constant variables Z_1, Z_0, non-constant core variables C_{10} and C_{01}, and non-constant petal variables P_{11}, P_{10}, P_{01} (where the indices indicate membership in α and γ, as β and δ are now determined by this). Identify variables according to type, and order them in the order of the previous sentence. We now have a relation whose arity depends on which variable types are represented in the witness. In the case that all seven types are present, we have implemented a 7-ary relation R_7 about which we know the following (the final tuple is produced on the witness tuples by the merge operation); the positions are in order Z_1, Z_0, C_{10}, C_{01}, P_{11}, P_{10}, and P_{01}:

$$\begin{aligned}
(1,0, \quad 1,0, \quad 1,1,0) &\in R_7 \\
(1,0, \quad 1,0, \quad 0,0,0) &\in R_7 \\
(1,0, \quad 0,1, \quad 1,0,1) &\in R_7 \\
(1,0, \quad 0,1, \quad 0,0,0) &\in R_7 \\
(1,0, \quad 1,0, \quad 1,0,0) &\notin R_7
\end{aligned}$$

To distinguish the final tuple from the witness tuples, we can observe that variable types P_{11}, P_{10}, and one further non-constant variable type must be represented by the witness; else $\sigma \in \{\alpha, \ldots, \delta\}$. The constant positions can be ignored by putting the constant variables z_1 and z_0 in these positions; we create them through constraints $(\neg z_0)$ and (z_1). Thus we implement a relation of arity between three and five. There are two main cases:

***R* is anti-Horn** In this case R is closed under disjunction and the tuple $\beta \vee \gamma$ is in R, implying $(1,0,1,1,1,0,1) \in R_7$. Thus the variable type C_{01} or P_{01} must occur; else $\sigma = \beta \vee \gamma$, a contradiction. Identify C_{01} and P_{01} if both occur, and set $C_{10} = 1$ if this type occurs, implementing a ternary relation R' which matches R_3 of Lemma 6.18, with the variable types being P_{11}, P_{10}, and $(P_{01} = C_{01})$ in the order used in Lemma 6.18. Indeed, $\{\alpha, \beta \vee \gamma, \beta\} \subseteq R$, representing the positive requirement, while there can be no tuple $(1,0,0) \in R'$, whether C_{10} occurs or not. This fulfills the conditions of Lemma 6.18, part 1.

***R* is not anti-Horn** In this case R is not closed under disjunction. Using a witness for this, we can implement a binary relation R_2 which is either $(x \neq y)$ or $(\neg x \vee \neg y)$. Likewise, by NP-completeness we have a relation which is not Horn, which can implement $(x \neq y)$ or $(x \vee y)$. Combining them, we find that we can always implement $(x \neq y)$, and thus are free to use R_5 of Lemma 6.18. We implement a relation R' as before, again letting the variable types appear in the order $(C_{10}, C_{01}, P_{11}, P_{10}, P_{01})$. We go through the cases of non-empty non-constant variable types, and show that our relation R' can implement a relation matching R_3 or R_5 of Lemma 6.18.

(a) If R' has arity three, with the third variable type being P_{01} or C_{01}, then we implement a relation matching R_3 with $\{(1,1,0),(1,0,1),(0,0,0)\} \subseteq R'$ and $(1,0,0) \notin R'$.

(b) If R' has arity three, and the third type is C_{10}, then we implement a relation R' with $\{(1,1,1),(1,0,0),(0,1,0),(0,0,0)\} \subseteq R'$ and $(1,1,0) \notin R'$. Use $R'(v,x,y) \wedge R'(w,x,z)$ to implement a relation matching R_5.

(c) If the core type C_{10} is not present, then identify C_{01} with P_{01}. This implements a relation R' matching R_3.

(d) If the core type C_{01} is not present, we need two cases. If $(0,1,0,0) \notin R'$, then identify C_{10} with P_{10} to produce a ternary relation matching R_3. Otherwise, we force $P_{01} = 0$ to produce a ternary relation as in case b.

(e) If the petal type P_{01} is not present, then $R'(v,w,x,y) \wedge R'(w,v,x,z)$ implements a relation matching R_5.

(f) If all five types are present, then $R'(v,w,x,y,z) \wedge R'(w,v,x,z,y)$ implements a relation matching R_5.

Thus in every case, we meet the conditions of part 1 or 2 of Lemma 6.18. □

Lower bound for Min Ones SAT

We can now show the main result of this section by giving a polynomial parameter transformation from EXACT HITTING SET(m) to Min Ones SAT(Γ), for suitable Γ.

Theorem 6.21. *Let Γ be a constraint language which is not mergeable. Then* Min Ones SAT(Γ) *is either polynomial-time solvable, or it does not admit a polynomial kernelization unless* NP $\subseteq$ co-NP/poly.

Proof. By Theorem 6.1, Min Ones SAT(Γ) is either polynomial-time solvable or NP-complete; assume that it is NP-complete. By Lemma 6.19 we may then assume that Γ contains the assignment constraints (x) and $(\neg x)$ as well as equality. Furthermore, by Lemma 6.20, we can implement log-cost selection formulas. It remains only to describe the polynomial parameter transformation from EXACT HITTING SET(m) to Min Ones SAT(Γ):

Transformation Let $(\mathcal{H}, m)$ be an instance of EXACT HITTING SET(m). If $\mathcal{H}$ contains more than 2^m vertices, then it can be solved in time polynomial in the input length [13]; otherwise, we create a formula $\mathcal{F}$ and fix a weight k so that $(\mathcal{F}, k)$ is positive if and only if $\mathcal{H}$ has an exact hitting set: Create one variable $y_{i,j}$ in $\mathcal{F}$ for every occurrence of a vertex v_i in an edge E_j in $\mathcal{H}$. For each edge $E \in \mathcal{H}$, create a log-cost selection formula over the variables representing the occurrences in E. Then, for all pairs of occurrences of each vertex v_i in any two edges E_j and $E_{j'}$, add constraints $(y_{i,j} = y_{i,j'})$. Finally fix $k = m + \sum_{E \in \mathcal{H}} w_{|E|}$, where w_i is the weight of a log-cost selection formula of arity i. We have an upper bound on the value of k of $\mathcal{O}(m \log n) = \mathcal{O}(m^2)$.

Correctness Now solutions with weight exactly k correspond to exact hitting sets of $\mathcal{H}$. Note that k is the minimum possible weight of the selection formulas, which is taken if exactly one occurrence in each edge is picked. By the definition of log-cost selection formulas, any solution where more than one occurrence has been picked (if such a solution is possible at all) will have a total weight which is larger than this, if the weight of the y-variables is counted as well, and thus such a solution to $\mathcal{F}$ of weight at most k is not possible.

As Exact Hitting Set(m) is NP-complete, it follows from [18] that a polynomial kernelization for Min Ones SAT(Γ) would imply the same for Exact Hitting Set(m), giving our result. □

Remark. Lemma 6.19 can be adjusted to provide (x), $(\neg x)$, and $(x = y)$ without the use of repeated variables and, using standard techniques (as in Chapter 4), we can show that the lower bound still applies under the restriction that constraints contain no repeated variables. Such a restriction can be useful in showing hardness of other problems, e.g., as for H-free EDGE DELETION (presented in Chapter 4).

6.5. Summary

We presented a dichotomy for Min Ones SAT(Γ) for finite sets of relations Γ into admittance or non-admittance of polynomial kernelizations, assuming that the polynomial hierarchy does not collapse. The characterization of the dichotomy is a closure property we call mergeability. We obtained the following result, with the polynomial cases due to Khanna et al. [75]:

Corollary 6.22. *Let* Γ *be a finite set of Boolean relations. 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.
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.

It might be interesting to compare our kernelization dichotomy to the approximation properties of Min Ones SAT(Γ), as characterized by Khanna et al. [75]. The mergeability property cuts through the classification of Khanna et al. as follows (we use the terms from B.4 of [75]). For every Γ such that Min Ones SAT(Γ) is known to be in APX, Min Ones SAT(Γ) admits a polynomial kernelization, while no problem identified as being

MIN HORN DELETION-complete is mergeable (every mergeable problem closed under disjunction is IHS-B+, i.e., APX-complete). The remaining classes are cut through (e.g., among the affine relations, the relation $(x \oplus y \oplus z = 1)$ is mergeable, while $(x \oplus y \oplus z = 0)$ is not). We also get kernelizations for some problems where the corresponding SATISFIABILITY problem is NP-complete, e.g., Min Ones Exact Hitting Set for sets of bounded arity, where no approximation is possible unless P = NP.

7. Max ones constraint satisfaction problems

7.1. Introduction

This chapter addresses the Max Ones SAT(Γ) problem, i.e., the problem of finding satisfying assignments with at least k true variables for formulas over Γ. Max Ones SAT(Γ) generalizes packing problems such as INDEPENDENT SET.

> **Max Ones SAT(Γ)**
>
> **Input:** A formula $\mathcal{F}$ over Γ and an integer k.
>
> **Parameter:** k.
>
> **Task:** Decide whether there is a satisfying assignment for $\mathcal{F}$ with at least k true variables.

We study the parameterized complexity of this problem, in particular, for which constraint languages Γ the problem becomes fixed-parameter tractable or (stronger) admits a polynomial kernelization. Khanna et al. [75] gave a characterization of Max Ones SAT(Γ) problems into polynomial cases as well as different degrees of approximability.

Theorem 7.1 ([75]). *Let Γ be a finite set of Boolean relations.*

- *If Γ is one-valid, anti-Horn, or width-2 affine, then* Max Ones SAT(Γ) *is in* P.
- *Else, if Γ is affine, then* Max Ones SAT(Γ) *is* APX-*complete.*
- *Else, if Γ is strongly zero-valid, Horn, or bijunctive, then* Max Ones SAT(Γ) *is poly*-APX-*complete.*
- *Else, if Γ is zero-valid, then* Max Ones SAT(Γ) *is not approximable.*
- *Else, feasibility is* NP-*hard (i.e.,* SAT(Γ) *is* NP-*complete).*

For our purpose of characterizing the parameterized complexity and admittance of polynomial kernelizations, we will use the polynomial cases as a starting point. However, it is also crucial to observe that Max Ones SAT(Γ) is not in XP, when deciding satisfiability of formulas over Γ is NP-complete (assuming P $\neq$ NP). To see this, consider the task of finding an assignment with at least zero true variables, which is equivalent to deciding feasibility; thus there cannot be an $n^{f(k)}$ time algorithm for Max Ones SAT(Γ), unless P = NP. Hence, we will focus on the six maximal constraint languages for which

SAT(Γ) is polynomial. Note also that, unlike Min Ones SAT(Γ), Max Ones SAT(Γ) is W[1]-hard already in the bijunctive, i.e., 2-CNF, case. The reason is that Max Ones SAT($\{(\neg x \vee \neg y)\}$) is equivalent to the W[1]-complete INDEPENDENT SET problem.

Our work Depending on the constraint language Γ we characterize Max Ones SAT(Γ) problems as having a polynomial kernelization; being FPT but admitting no polynomial kernelization unless NP $\subseteq$ co-NP/poly; being W[1]-hard and in XP; and not being in XP unless P = NP. We follow joint work with Dániel Marx and Magnus Wahlström [79].

Structure of this chapter We arrive at our characterization (Section 7.2) by considering the six maximal constraint languages for which satisfiability is known to be in P, namely zero-valid, one-valid, Horn, anti-Horn, affine, and bijunctive constraint languages. We will start from the known polynomial cases for Max Ones SAT(Γ) (i.e., one-valid, anti-Horn, and width-2 affine) and then address affine, Horn, and zero-valid constraint languages. Finally, using Nordh and Zanuttini's [96] notion of frozen co-clones, we classify the bijunctive cases, which do not follow the co-clone lattice (e.g., there are constraint languages which generate the co-clone of bijunctive relations but with different kernelizability of Max Ones SAT(Γ)). We give a short summary in Section 7.3.

7.2. 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 $\mathcal{F}$ there is a satisfying assignment with at least zero true variables if and only if $\mathcal{F}$ is satisfiable. By Schaefer (Theorem 5.3), 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. In the following we will consider Max Ones SAT(Γ) for the six maximal constraint languages for which SAT(Γ) is in P.

Polynomial cases

We start with those cases for which Max Ones SAT(Γ) can be solved in polynomial time, as characterized by Khanna et al. [75] (Theorem 7.1): Max Ones SAT(Γ) is in P if Γ is one-valid, anti-Horn, or width-2 affine, and APX-hard otherwise.

It remains to consider affine, Horn, zero-valid, and bijunctive constraint languages.

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 7.2. *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 $(\mathcal{F}, k)$ be an instance of Max Ones SAT(Γ); if $\mathcal{F}$ 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 fewer 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 $(\mathcal{F}', k')$ be the instance obtained after replacing all but at most two fixed variables (one fixed to true and one fixed to false). If $\mathcal{F}'$ contains less than $2k' + 2 \in \mathcal{O}(k)$ variables, then we return $(\mathcal{F}', k')$ as a kernel. Otherwise, we show that $(\mathcal{F}', k')$ has a satisfying assignment having weight at least k', thus 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 F'. 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 F' to $\{x, y\}$ would be a binary relation having exactly 3 tuples. 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). Furthermore, 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 $(\mathcal{F}', k')$. □

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 7.3. *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 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 $c_0 = 0$, $c_1 = 1$ satisfies R, this would imply that $c_0 = c_1 = 0$ also satisfies 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 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	$\notin R'$

If R' is not zero-valid, then as above we create a variable $c_1 = 1$ and put it in place of z, implementing $(\neg x \vee \neg y)$ 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. $\square$

Zero-valid cases

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

Lemma 7.4. *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 will 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); see Table 7.1. Further, we put c_0 in R_ℓ where α_ℓ and β_ℓ are both 0. 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

	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	$\notin R_3$

Table 7.1.: Using witness tuples to implement a ternary relation $R'(x, y, z)$.

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 $R'(c_0, x, y)$ implements $(y \to x)$ 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 $R'(x, c_0, y)$ implements $(y \to x)$ 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 5.3, i.e., Schaefer's dichotomy theorem). We will reduce SAT(Γ') to Max Ones SAT(Γ) with parameter $k = 1$.

Let $\mathcal{F}'$ be a formula over Γ'. Assume that $\mathcal{F}'$ is false for the all-zero assignment, otherwise we may reduce to a dummy positive instance. We will create a formula $\mathcal{F}$ using only R, which has a non-zero solution if and only if $\mathcal{F}'$ is satisfiable: Copy all constraints using R from $\mathcal{F}'$ to $\mathcal{F}$, and add a single global variable $\hat{z}$. For every variable x in $\mathcal{F}'$, 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 $\mathcal{F}'$, create a constraint $R(x, y, \hat{z})$ in $\mathcal{F}$. Now every solution to $\mathcal{F}$ where any variable is true must have $\hat{z} = 1$, which "activates" all inequalities from $\mathcal{F}'$. Thus $\mathcal{F}'$ is satisfiable if and only if $\mathcal{F}$ has a satisfying assignment with at least one true variable, i.e., if $(\mathcal{F}, 1) \in$ Max Ones SAT(Γ). □

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 (or zero-valid, or one-valid, but this follows implicitly using invariance under majority). This corresponds to the constraint languages Γ which, using existentially quantified variables, can implement all 2-SAT clauses; see [96]. We are going to find that there is no unique answer for these languages (which are properly bijunctive). We will find that 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 will need the results of Nordh and Zanuttini [96], who studied co-clones under more limited implementations than existential implementations, namely frozen implementations. 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. We will use the characterization of [96] of the frozen partial co-clones that our Γ can generate. The free use of equality constraints is somewhat more general than what we wish to allow, but we will find that it causes no problems.

We need the following special cases.

1. $\Gamma_2^{p\neq} = \{(x \vee y), (x \neq y)\}$
2. $\Gamma_3^n = \{R_3^n\}$, where $R_3^n = (\neg x \vee \neg y) \wedge (x \neq z)$
3. $\Gamma_2^{p\neq i} = \{(x \vee y), (x \neq y), (x \rightarrow y)\}$

Finally, we need a technical lemma to show that we can assume that we have access to the constants.

Lemma 7.5. *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. Given such a constraint language Γ, if every relation is closed under negation then Γ is width-2 affine. Thus we can assume that there is a relation $R \in \Gamma$ which 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. 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 7.6. *Let Γ be a set of Boolean relations that generates the co-clone of bijunctive relations (i.e., such that Γ is bijunctive but neither Horn, anti-Horn, nor width-2 affine).*

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

Remark. In the proof of Lemma 7.6 we will use results for frozen implementations that allow equality in addition to relations from the constraint language Γ. In two places we will say that equality is not useful for implementing a certain constraint, e.g., for implementing $(\neg x \vee \neg y)$; the intuition behind that statement is as follows: Assume that we have some frozen implementation of $(\neg x \vee \neg y)$ that contains at least one equality and let X be the set of existentially quantified variables. Clearly, the implementation cannot contain $(x = y)$, since that would exclude for example $x = 0$ and $y = 1$. If the equality constraint contains at least one variable from X, say $(z = z')$, with $z \in X$ and $z' \in X \cup \{x, y\}$, then we may replace all occurrences of z by z' and discard $(z = z')$. Thus equality is not useful for implementing $(\neg x \vee \neg y)$, since it must contain variables from X, but then we can simplify the implementation and use one less existentially quantified variable.

Proof of Lemma 7.6. Let $(\mathcal{F}, k)$ be a Max Ones SAT(Γ) instance. Assume throughout that the instance is feasible (as otherwise, the problem is trivial). By assumption Γ is not Horn, anti-Horn, or width-2 affine, else it would not generate the co-clone of bijunctive relations. 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 7.5. We split the proof into four parts.

Polynomial kernelization If $\Gamma \subseteq \langle \Gamma_2^{p\neq} \rangle_{fr}$, then, by [96], every relation in Γ, and thus all of $\mathcal{F}$, has a frozen implementation over $\Gamma_2^{p\neq} \cup \{=\}$, i.e., using positive clauses, equality, and disequality. We will refer to this implementation when inferring a kernel, but the kernelization will apply for the original Γ as well. Let a 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 any solution will contain at least k true variables, and can be found in polynomial time. 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 we can find 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 [96], there is an implementation of R_3^n over $\Gamma \cup \{=\}$. As the equality constraint will not be useful in such an implementation, there is also an implementation directly over Γ, showing that Max Ones SAT(Γ_3^n) reduces to Max Ones SAT(Γ) by a polynomial parameter transformation. We will in turn show that the MULTIPLE COMPATIBLE PATTERNS (MCP) problem reduces to Max Ones SAT(Γ_3^n) by a polynomial parameter transformation.

Observe that R_3^n can be written as $(x \neq y) \wedge (z \to x)$. Let (I, k) be an instance of MCP, with string length r. Create variables $(x_i \neq y_i)$ for $1 \leq i \leq r$, coding the entries of the solution string; these variables contribute weight exactly r to any solution. The intuition is that $x_i = 1$ codifies that the solution string has value 0 at position i, and that $y_i = 1$ codifies value 1. Now for every pattern i, create a variable z_i, and for every position j of pattern i containing 0, add a constraint $(x_j \neq y_j) \wedge (z_i \to x_j)$. For positions containing 1, create the same constraint with an implication instead to 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 Theorem 1.5 and Lemma 1.8, 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 $\mathcal{F}$ 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. 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 [96], there is an implementation of $(\neg x \vee \neg y)$ over $\Gamma \cup \{=\}$, and again the equality constraint would not be useful. Thus there is an fpt-reduction from INDEPENDENT SET to Max Ones SAT(Γ). □

7.3. Summary

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

Corollary 7.7. *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 $\Gamma \subseteq \langle (x \vee y), (x \neq y), (x \to y) \rangle_{fr}$, then* Max Ones SAT($\Gamma$) *is in* FPT*, with a running time of $2^k \cdot n^{\mathcal{O}(1)}$, but if the previous case does not apply, then there is no polynomial kernelization unless* NP $\subseteq$ co-NP/poly*.*

4. *If none of these cases applies, then* Max Ones SAT(Γ) *is* W[1]*-hard; if* Γ *is Horn or bijunctive, then* Max Ones SAT(Γ) *is in* XP*.*

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

Remark. 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 $(\mathcal{F}, k)$ 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. It is crucial that setting a variable to true still leaves a formula that is bijunctive; this can be seen by adding the bijunctive constraint (x) to the formula for each selected variable x. Note that this does not work for general zero-valid constraint languages.

For Max Ones SAT(Γ) the FPT cases almost coincide with the cases that admit polynomial kernelizations. This contrasts Min Ones SAT(Γ), which is FPT for every Γ but which does not always admit a polynomial kernelization. Same as for Min Ones SAT(Γ) the constant-factor approximable cases (i.e., for affine constraint languages) are subsumed by the cases that admit polynomial kernelizations.

8. Exact ones constraint satisfaction problems

8.1. Introduction

In this chapter we consider the kernelizability of Exact Ones SAT(Γ) problems, i.e., the problem of deciding whether a given formula over Γ has a satisfying assignment with a specified number of true variables:

> **Exact Ones SAT(Γ)**
>
> **Input:** A formula $\mathcal{F}$ over Γ and an integer k.
>
> **Parameter:** k.
>
> **Task:** Decide whether there is a satisfying assignment for $\mathcal{F}$ with exactly k true variables.

Creignou et al. [39] 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 [88]; the result is a dichotomy into FPT and W[1]-complete cases based on two partial polymorphisms (see Theorem 8.2).

In comparison with the min ones and the max ones variants consider the following: For every constraint language Γ, Min Ones SAT(Γ) reduces to Exact Ones SAT(Γ) by a polynomial parameter transformation (see Lemma 8.3). Furthermore, some hard max ones problems like CLIQUE, i.e., Max Ones SAT($\{(\neg x \vee \neg y)\}$), are also hard when asking for exactly k ones. However, while Max Ones SAT(Γ) is not contained in XP for some Γ (assuming P ≠ NP), Exact Ones SAT(Γ) is in W[1] for every choice of Γ (as observed by Marx [88]). Thus, while it does not have an optimization variant, it is interesting as a refinement of WEIGHTED q-SAT, which is complete for W[1] (cf. [47]).

> **WEIGHTED q-SAT**
>
> **Input:** A q-CNF formula $\mathcal{F}$, i.e., with clauses of at most q literals, and an integer k.
>
> **Parameter:** k.
>
> **Task:** Decide whether $\mathcal{F}$ has a satisfying assignment with exactly k true variables.

Our work Following joint work with Dániel Marx and Magnus Wahlström [79] we characterize the kernelizability of Exact Ones SAT(Γ) problems. The characterization is by mergeability as well as by semi-separability (i.e., a combination of the two partial polymorphisms which characterize fixed-parameter tractability): If Γ is mergeable as well as semi-separable then Exact Ones SAT(Γ) admits a polynomial kernelization, otherwise it does not admit a polynomial kernelization unless NP ⊆ co-NP/poly.

Structure of this chapter In the following section, we recall some definitions and mention some related results. Sections 8.3 and 8.4 present the classification of Exact Ones SAT(Γ); the former covers choices of Γ for which Min Ones SAT(Γ) is polynomial, the latter addresses those for which Min Ones SAT(Γ) is NP-complete. We summarize and conclude in Section 8.5.

8.2. Definitions and related results

In this section we recall some definitions and results from previous chapters as well as from Marx's [88] characterization of the parameterized complexity of Exact Ones SAT(Γ). We first define weak separability (introduced in [88]) as well as a stronger, joined version of the two partial polymorphisms defining it.

Definition 8.1 ([88])**.** 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	1	0	1
0	1	1	1
0	0	0	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).

Now we can state Marx's characterization of the fixed-parameter tractability of Exact Ones SAT(Γ), which is as follows:

Theorem 8.2 ([88])**.** *Let Γ be any finite Boolean constraint language. The problem* Exact Ones SAT(Γ) *is fixed-parameter tractable if every relation $R \in \Gamma$ is weakly separable. In the remaining cases* Exact Ones SAT(Γ) *is* W[1]*-complete.*

Since any kernelization for a problem also implies fixed-parameter tractability, we will only need to further classify the fixed-parameter tractable cases, i.e., weakly separable constraint languages. Furthermore, by a polynomial parameter transformation from Min Ones SAT(Γ) to Exact Ones SAT(Γ), we may conclude that lower bounds from the min ones setting may be transferred to the exact ones setting.

Lemma 8.3. Min Ones SAT(Γ) *reduces to* Exact Ones SAT(Γ) *by a polynomial parameter transformation.*

Remark. The essential idea in the reduction is that adding k dummy variables to $\mathcal{F}$ would give a formula $\mathcal{F}'$ that has a satisfying assignment with exactly k true variables if and only if $\mathcal{F}$ has a satisfying assignment with at most k true variables. The actual reduction is a bit more complicated to avoid introducing free variables that do not occur in any constraint.

Proof of Lemma 8.3. Let $(\mathcal{F}, k)$ 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 and creating an equivalent dummy instance of Exact Ones SAT(Γ) depending on the outcome. Otherwise, let R be a relation from Γ that is not zero-valid. We show that we are able to create a formula $\mathcal{F}_0$ 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, $(\mathcal{F} \wedge \mathcal{F}_0, k + k_{\min})$ is in Exact Ones SAT(Γ) if and only if $(\mathcal{F}, k)$ is in Min Ones SAT(Γ).

We assume first that R contains two tuples α and β such that $\alpha < \beta$. 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 $\mathcal{F}_0$ 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 $\mathcal{F}_0$ 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 $\alpha \not< \beta$ and $\beta \not< \alpha$, 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 $\mathcal{F}_0$ consist of k variable-disjoint copies of R'''-constraints. Again $\mathcal{F}_0$ 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. $\square$

Thus, using the kernelization dichotomy for Min Ones SAT(Γ) (see Chapter 6), 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) was 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

Using Corollary 6.22 (about the kernelizability of Min Ones SAT(Γ)) and Lemma 8.3 we immediately obtain the following corollary:

Corollary 8.4. *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*.*

According to Khanna et al. [75] 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. For the remaining cases we may then assume Γ to be mergeable since Min Ones SAT(Γ) will be NP-hard; see Section 8.4. Both sections combined constitute a proof for the following theorem, characterizing Exact Ones SAT(Γ).

Theorem 8.5. *Let Γ be a 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 Horn respectively zero-valid and mergeable is also width-2 affine.

8.3. 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.

Width-2 affine cases

If all relations in Γ are width-2 affine then, by Creignou et al. [39], Exact Ones SAT(Γ) is in P. To see this, consider the fact 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; 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). For each such pair of classes, say C_1 and C_2, at least $\min(|C_1|, |C_2|)$ variables must be assigned true. Thus, conceptually, we may shrink both classes by $\min(|C_1|, |C_2|)$ and reduce k by that value; one of the two classes is empty. Hence the only remaining constraint for finding a satisfying assignment is a partition of the remaining variables into independent equivalence classes; a satisfying assignment with the right number of true variables can then be found via dynamic programming (if one exists).

Horn cases

We show that every weakly separable relation which 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 8.6. *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 (ii), α, (iii)	

Thus, by [38], 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 8.7. *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.

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 8.8. *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 $\alpha \wedge \beta$, α, 0	(i)
$\neg\alpha \wedge \beta$	0	1	0	0	FPT(2) on $\alpha \wedge \beta$, β, 0	(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 [38]. □

We will now prove that we may assume to have positive and negative assignments available in Γ. To this end, as in the previous chapters, 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) by Theorem 1.5. 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 8.9. *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. Given an instance $(\mathcal{F}, k)$ of Exact Ones SAT($\Gamma \cup \{(x), (\neg x)\}$) it clearly suffices to show how to implement the (x) and $(\neg x)$ constraints. First, we make a copy x' of every variable x of $\mathcal{F}$ and add an implementation of the equality 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 an implementation of $(x = c_1)$. 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 implement all $(\neg x)$ constraints as $(x = c_0)$.

The formula $\mathcal{F}'$ obtained in this way has a satisfying assignment with exactly k' true variables if and only if $\mathcal{F}$ has a satisfying assignment with k true variables. □

Now, we can show that for constraint languages which are zero-valid and weakly separable but not Horn we may assume that they contain assignments with respect to admitting or not admitting a polynomial kernelization.

Lemma 8.10. *Let Γ be a constraint language that is zero-valid and weakly separable 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 $(\mathcal{F}, k)$ of Exact Ones SAT$(\Gamma \cup \{(x), (\neg x)\})$ we will show how to express (x) and $(\neg x)$ to obtain an equivalent instance $(\mathcal{F}', k')$ 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 8.9 to obtain an equivalent instance $(\mathcal{F}', k')$ 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 $R(x, \dots, x)$ 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 8.9. $\square$

Remark. In the proof of Lemma 8.10 we implement a relation R' with $(0,0),(1,1) \in R'$ and $(1,0) \notin R'$, we then have an implementation of equality as $R'(x,y) \wedge R'(y,x)$. Note that equivalently, by using that Γ must be weakly separable, we may conclude that R' *is* equality. The reason is that it must be invariant under FPT(2); this partial polymorphism in particular excludes implications (i.e., $\{(0,0),(0,1),(1,1)\}$).

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 8.11. *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 8.6. Then we see that Γ is not mergeable, since mergeability applied to $\alpha, 0, \beta, 0 \in R$ (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 8.10, 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 [75], Min Ones SAT(Γ') is NP-hard, and by Corollary 8.4, 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 8.12. *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.

8.4. Characterization II: Remaining cases

Having handled the cases for which Min Ones SAT(Γ) is in P, we may now by Corollary 8.4 restrict our attention to sets of relations Γ which are mergeable as well as weakly separable. We begin with a special case (for which we do not require mergeability).

Lemma 8.13. *Let R be a weakly separable relation. If R is anti-Horn, then R can be implemented by equality, positive clauses, and assignment.*

Proof. We will 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 (i), (ii), α	

Thus R is invariant under $a \vee (b \wedge \bar{c})$ implying by [38] that it can be implemented using equality, negative assignments, and positive clauses all of which are semi-separable and mergeable. □

Lemma 8.14. *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 the previous lemma, Γ is mergeable; thus by Lemma 8.8 we may assume that Γ contains at least one relation which 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 8.9. 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 $(x \neq y)$ 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 8.10 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 $R \in \Gamma$). $\square$

Thus when Γ is anti-Horn, we either have a reduction of 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 8.18. In the former case, we get a polynomial kernelization for Exact Ones SAT(Γ) as a conclusion of Theorem 1.5.

We will now consider the case that Γ is mergeable and contains at least one relation which is not anti-Horn.

Lemma 8.15. *Let Γ be a weakly separable constraint language which 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 witness tuples $\alpha, \beta \in R$ for some $R \in \Gamma$ such that $\alpha \vee \beta \notin R$. 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.

	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	$\notin R$

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.

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 $(\mathcal{F}, k)$ of Exact Ones SAT(Γ) as well as a constraint $R'(c_0, c_1)$ obtaining $\mathcal{F}'$. Clearly $(\mathcal{F}, k)$ and $(\mathcal{F}', k+1)$ 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 $R''(c_0, x, y)$. $\square$

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

Lemma 8.16. *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 $(\mathcal{F}, k)$ be an instance of Exact Ones SAT($\Gamma \cup \{(x), (\neg x)\}$). We show how to obtain an equivalent instance $(\mathcal{F}', k')$ of Exact Ones SAT(Γ) by implementing (x) and $(\neg x)$: We start with $\mathcal{F}' = \mathcal{F}$ 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 $\mathcal{F}'$ 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 $\mathcal{F}$ with k true variables. The converse is straightforward too. $\square$

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 8.17. *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 7.6. The definition of MCP can be found in Section 1.4 and Lemma 1.8 shows that MCP does not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly. W.l.o.g. Γ contains 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 an implementation of $(y \neq z)$.

Now we can use the same reduction from MCP as in Lemma 7.6, 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) and is thus stronger (i.e., it implies the other two).

Theorem 8.18. *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 8.14 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 by Theorem 1.5 (and using the fact that d-HITTING SET admits a polynomial kernelization).

If Γ is not anti-Horn, then by Lemmas 8.15 and 8.16 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 $(\mathcal{F}, k)$ 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 $(\mathcal{F}, k)$ 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 $(\mathcal{F}, k)$.

Non zero-valid constraints First we will 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 $\mathcal{F}$ is greater than $(d!)^2 \cdot k^d$ we can find a sunflower consisting of $k+1$ R-constraints by applying Lemma 2.7 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 that $(\alpha, 0)$ 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	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):

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

  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 8.8. 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 $\mathcal{F}$.

Zero-valid constraints Now we simplify zero-valid constraints. We replace each zero-valid constraint by an implementation using negative assignments and equality constraints, according to Lemma 8.8. 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 which have 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. 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 $(\mathcal{F}, k)$. □

8.5. Summary

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

Corollary 8.19. *Let Γ be a 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 $\subseteq$ 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., $((x \rightarrow y) \land (y \neq z))$ is weakly separable and mergeable but not semi-separable.

Part IV.

Conclusion and open problems

9. Conclusion

In this chapter we review some of the results presented in this thesis. In the following chapter we will present and discuss some open problems and directions of further research.

Motivating our work and explaining the general interest in kernelization, we have argued that data reduction is a natural approach for solving problems and for arriving at decisions, especially for combinatorially hard problems. We considered two ways of giving a formal definition for data reduction, in order to allow a rigorous study. On the one hand, we observed that a straightforward definition of data reduction, i.e., as a polynomial-time algorithm that shrinks all instances of a given problem, gives a concept that can be ruled out for NP-hard problems. On the other hand, we saw that (polynomial) kernelization is a robust way of formalizing data reduction. Moving the question of analyzing data reductions to the field of parameterized complexity allows a rigorous study, and parameterized complexity provides a decent amount of tools.

We presented polynomial kernelizations for the standard parameterizations of all problems from MIN $F^+\Pi_1$ and MAX NP (which includes MAX SNP). These results apply also to the weighted versions if the weights are non-negative rationals and there are no arbitrary small weights (we have argued that less restricted weights make fixed-parameter tractability impossible unless P = NP). Thus, by showing that a problem can be expressed as a MIN $F^+\Pi_1$ or MAX NP problem, one immediately obtains a polynomial kernelization from our result.

For edge modification problems we exhibited a small graph H, such that H-free EDGE DELETION and H-free EDGE EDITING do not admit polynomial kernelizations unless NP $\subseteq$ co-NP/poly. This answered an open question of Cai (posed at IWPEC 2006, see [15]), namely whether the Π EDGE DELETION problem admits a polynomial kernel when Π can be characterized by a finite set of forbidden induced subgraphs. Our results show that there is no general positive answer, but that giving a characterization of classes of positive or negative cases, or possibly a complete classification, is an interesting problem.

Finally, we completely characterized the admittance of polynomial kernelizations for Min Ones SAT(Γ), Max Ones SAT(Γ), and Exact Ones SAT(Γ) problems, depending on the constraint language Γ. For Max Ones SAT(Γ) we also characterized the parameterized complexity into admitting a polynomial kernelization, being FPT but not admitting a polynomial kernelization unless NP $\subseteq$ co-NP/poly, being W[1]-hard but in XP, and not being contained in XP unless P = NP. Apart from giving complete characterizations and providing general polynomial kernelizations (including some new cases like maximizing the number of true variables in an affine formula), we showed that a wide range of these problems do not admit polynomial kernelizations. As we have seen in Chapter 4 such problems, like the Not-1-in-3 SAT problem, make good source problems to extend

Γ	Min Ones	Exact Ones	Max Ones
width-2 affine	P	P	P
$\{(x \oplus y \oplus z = 1)\}$	PK	PK	P
$\{(x \oplus y \oplus z = 0)\}$	P	FPT	PK
$\{(x \oplus y \oplus z = 0), (x)\}$	FPT	FPT	PK
$\{(w \oplus x \oplus y \oplus z = 1)\}$ / 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
$\{(x + y + z = 1)\}$	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 9.1.: Examples of constraint languages 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; problems marked not in XP are not contained in XP unless P = NP.

the lower bounds via polynomial parameter transformations. See Table 9.1 for a number of constraint languages as well as the corresponding results for the three CSP variants.

Structure of the chapter In Section 9.1 we discuss possible extensions of the results for MIN $F^+\Pi_1$ and MAX NP within the optimization classes defined by Kolaitis and Thakur [76, 77]. In Section 9.2 we compare the admittance of polynomial kernelizations to that of having constant-factor approximation algorithms for some of the problems considered in this thesis (i.e., for those which are the decision version of some optimization problem).

9.1. Beyond MIN $F^+\Pi_1$ and MAX NP

In this section we briefly discuss the possibility of extending the polynomial kernelizations for MIN $F^+\Pi_1$ and MAX NP to larger classes within the framework of Kolaitis and Thakur [76, 77]. We will omit formal definitions of the encountered classes and only give high-level descriptions. Let us begin with the minimization problems, i.e., generalizing beyond MIN $F^+\Pi_1$:

MIN $F^+\Pi_1$ The next obvious relaxation of MIN $F^+\Pi_1$ is the class MIN $F\Pi_1$, i.e., MIN $F^+\Pi_1$ without the restriction that the formula $\psi(\mathbf{x}, \mathcal{A}, S)$ be positive in S (the relation which we minimize over). We remark that the fact that we needed only to minimize over a single relation instead of a tuple of relations relied heavily on this positivity property. However, even with only a single relation, we are able to express Min Ones SAT($\{(x+y+z \neq 1), (x)\}$) as a MIN $F\Pi_1$ problem:

Example 9.1. We express formulas $\mathcal{F}$ over $\{(x+y+z \neq 1), (x)\}$ as finite structures $\mathcal{A} = (A, P, R)$ where A contains the variables, P is a unary relation, and R is a ternary relation. We let $P(x) = 1$ if $\mathcal{F}$ contains a constraint (x) and we let $R(x, y, z) = 1$ if $\mathcal{F}$ contains a constraint $(x + y + z \neq 1)$. The unary relation S is intended to encode a truth assignment to the variables in A, i.e., $S(x) = 1$ if x is assigned true. Thus, given a correct formulation, $|S|$ will be the number of true variables. We have the following expression:

$$\begin{aligned}\min_{S\subseteq A}\{|S| : (\mathcal{A}, S) \models (\forall \mathbf{x} \in A^3) :&(\neg R(x_1, x_2, x_3) \vee \neg S(x_1) \vee S(x_2) \vee S(x_3))\\ &\wedge (\neg R(x_1, x_2, x_3) \vee S(x_1) \vee \neg S(x_2) \vee S(x_3))\\ &\wedge (\neg R(x_1, x_2, x_3) \vee S(x_1) \vee S(x_2) \vee \neg S(x_3))\\ &\wedge (\neg P(x_1) \vee S(x_1))\}\end{aligned}$$

The quantifier-free first-order formula in this expression is equivalent to

$$(R(x_1, x_2, x_3) \to (S(x_1) + S(x_2) + S(x_3) \neq 1)) \wedge (P(x_1) \to S(x_1)).$$

We see that this formulation forces S to correspond immediately to a satisfying assignment for $\mathcal{F}$. Clearly, we will obtain an assignment with a minimum number of true variables.

Thus MIN $F\Pi_1$ already contains problems which do not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly. Furthermore, given some instance of a MIN $F\Pi_1$ problem, we may proceed as in Section 3.3 and insert all the given relations (and simplify). Thus we obtain CNF formulas, i.e., conjunctions of clauses, on literals $S(.)$, and the task is to find a satisfying assignment that minimizes $|S|$. This is quite similar to Min Ones SAT(Γ) problems, and the study of their kernelizability is a natural conclusion to the observation about MIN $F\Pi_1$. (In fact, this is what motivated the work on Min Ones SAT(Γ) problems presented in Chapter 6.)

Two other generalizations of MIN $F^+\Pi_1$ are obtained by allowing *exists-forall* or *forall-exists* quantification, namely MIN $F^+\Sigma_2$ and MIN $F^+\Pi_2$, respectively. The latter already contains the W[2]-complete DOMINATING SET problem. The former class appears to contain compositional problems, e.g.:

OR d-HITTING SET

Input: A set of d-dimensional hypergraphs $\{\mathcal{H}_1, \ldots, \mathcal{H}_t\}$ for some integer t and an integer k.

Parameter: k.

Task: Decide whether at least one of the hypergraphs has a hitting set of size at most k.

Clearly, OR d-HITTING SET is NP-complete and it is immediate to give a composition algorithm for it: Given some number of instances with parameter value k, we make a union of the sets of hypergraphs. It is easy to see that the new instance (with parameter k) is yes if and only if one of the original instances is a yes-instance. Hence, OR d-HITTING SET does not admit a polynomial kernelization unless NP $\subseteq$ co-NP/poly (by Theorem 1.4).

Without formally introducing MIN $\mathrm{F}^+\Sigma_2$ let us sketch the idea of how to express OR d-HITTING SET: We minimize over a single relation S that is intended to encode the hitting set. In the existential quantification the number of a hypergraph is chosen. The remaining formula (universally quantified) expresses that each edge of the chosen hypergraph has a nonempty intersection with the hitting set encoded by S. To this end the edges are given as a $d{+}1$-ary relation R such that $(x_1,\ldots,x_d,i) \in R$ iff $\{x_1,\ldots,x_d\}$ is an edge of the ith hypergraph (repeating vertices to encode smaller edges). The following formula then expresses the smallest hitting set size over the given t hypergraphs:

$$\min_{S\subseteq A}\Big\{|S| : (\exists i \in A)(\forall \mathbf{x} \in A^d) : ((\neg R(x_1,\ldots,x_d,i) \vee S(x_1) \vee \ldots \vee S(x_d)) \wedge N(i))\Big\}$$

The base set A contains the disjoint union of the vertex sets of the t hypergraphs as well as the natural numbers from 1 to t. The relation $N(i)$ is true if $i \in \{1,\ldots,t\}$.

Thus MIN $\mathrm{F}^+\Sigma_2$ contains problems that do not admit a polynomial kernelization (unless NP $\subseteq$ co-NP/poly). However, the exists-forall quantification seems to allow *disjunctive* polynomial kernelizations: One reduces to a possibly large number of small instances, such that at least one of them is a yes-instance iff the original instance is a yes-instance (for an example see Fernau et al. [53]). The idea is to make one instance for each possible assignment to the existentially quantified variables (e.g., $\exists \mathbf{x} \subseteq A^{c_x}$); the kernelization can then focus on the remaining universally quantified formula. Note that this is trivial for OR d-HITTING SET: The disjunctive polynomial kernelization is obtained by turning each hypergraph into a single instance (with parameter k) and using a polynomial kernelization for d-HITTING SET.

MAX NP In the framework of Kolaitis and Thakur [76, 77] MAX NP is called MAX Σ_1. Further, less restricted, classes would be MAX $\mathrm{F}^-\Pi_1$ or its superclass MAX Π_1 = MAX $\mathrm{F}\Pi_1$ (note that many of the classes introduced in [76, 77] are also showed to have the same expressibility). However, both classes contain the W[1]-complete CLIQUE problem, for which even an fpt-algorithm is unlikely. Again, Boolean CSPs are a possible perspective. When restricted to maximizing over a single relation S, then MAX $\mathrm{F}\Pi_1$ problems are closely related to Max Ones SAT(Γ) problems (similar as for MIN $\mathrm{F}\Pi_1$).

Summary Within the framework of Kolaitis and Thakur our results cannot be extended to larger classes (i.e., with more quantification power or without the positivity requirement). Boolean constraint satisfaction problems provide a way of further analyzing some of the problems by restricting the formulas which are permitted.

9.2. Kernelization and approximation revisited

We will now relate some of our kernelization results to the approximability of the corresponding optimization problems; this applies for Chapters 3, 6, and 7. We are not aware of approximability results for the H-free EDGE DELETION and the H-free EDGE EDITING problem, considered in Chapter 4. The only result for $\mathcal{H}$-free EDGE DELETION problems is that a few of these problems, i.e., for some choices of $\mathcal{H}$, are contained in MIN $\text{F}^+\Pi_1$ (cf. [77]), implying they are constant-factor approximable and that they admit polynomial kernelizations.

The Exact Ones SAT(Γ) problem, considered in Chapter 8, does not have an optimization variant.

MIN $\mathbf{F^+\Pi_1}$ and MAX NP

In Chapter 3 we showed that the standard parameterizations of all MIN $\text{F}^+\Pi_1$ and MAX NP problems admit polynomial kernelizations. It is known that all problems from these classes can be approximated to within a constant factor of the optimum in polynomial time. Khanna, Motwani, Sudan, and Vazirani [74] showed that APX is the closure of MAX SNP under PTAS-preserving reductions, and APX-PB is its closure under so-called E-reductions. Thus, while PTAS-preserving reductions as well as E-reductions do not appear to preserve kernelizability or even parameterized complexity (both closures include CONNECTED VERTEX COVER and BIN PACKING), there is a syntactic core of APX on which the perceived relation holds true.

Min Ones SAT($\boldsymbol{\Gamma}$)

Let us begin by restating the characterization of approximability due to Khanna et al. [75].

Theorem 9.2 ([75]). *Let Γ be a finite set of Boolean relations.*

- *If Γ is zero-valid, Horn, or width-2 affine then* Min Ones SAT(Γ) *is in* P*.*
- *Else, if Γ is bijunctive or IHS-B then* Min Ones SAT(Γ) *is* APX*-complete.*
- *Else, if Γ is affine then* Min Ones SAT(Γ) *is* NEAREST CODEWORD*-complete.*
- *Else, if Γ is anti-Horn then* Min Ones SAT(Γ) *is* MIN HORN DELETION*-complete.*
- *Else, if Γ is one-valid then* Min Ones SAT(Γ) *is poly-*APX*-complete.*
- *Else,* SAT(Γ) *is* NP*-hard.*

Remark. Arora et al. [5] showed that NEAREST CODEWORD is hard to approximate to within a factor of $\Omega(2^{\log^{1-\epsilon} n})$ for any $\epsilon > 0$. The same bound holds for MIN HORN DELETION (cf. [75]). For this discussion it suffices that these results rule out constant-factor approximability (assuming P $\neq$ NP).

If Γ is bijunctive or IHS-B, then it can be implemented by positive clauses, negative clauses, and implications; all of these are mergeable. Thus in the known constant-factor approximable cases Min Ones SAT(Γ) does also admit a polynomial kernelization. Additionally, there are constraint languages that are mergeable, but which are neither bijunctive nor IHS-B. In fact already IHS-B allows only positive clauses *or* negative clauses, whereas any choice of positive and negative clauses gives a mergeable constraint language. Furthermore, $\{(x \oplus y \oplus z = 1)\}$ is affine but not contained in the APX or P cases, however Min Ones SAT($\{(x \oplus y \oplus z = 1)\}$) admits a polynomial kernelization.

Max Ones SAT(Γ)

Again, let us begin the discussion by recalling the characterization of approximability.

Theorem 9.3 ([75]). *Let Γ be a finite set of Boolean relations.*

- *If Γ is one-valid, anti-Horn, or width-2 affine then* Max Ones SAT(Γ) *is in* P*.*
- *Else, if Γ is affine* Max Ones SAT(Γ) *is* APX*-complete.*
- *Else, if Γ is strongly zero-valid, Horn, or bijunctive then* Max Ones SAT(Γ) *is poly-*APX*-complete.*
- *Else, if Γ is zero-valid then* Max Ones SAT(Γ) *is not approximable.*
- *Else,* SAT(Γ) *is* NP*-hard.*

We showed that Max Ones SAT(Γ) admits a polynomial kernelization when Γ is affine. Additionally, we obtained polynomial kernelizations for some of the frozen co-clones within the co-clone of bijunctive relations. Thus, again, we have polynomial kernelizations for all choices of Γ such that the optimization version is constant-factor approximable.

Summary All constant-factor approximable problems considered in this thesis have been shown to admit polynomial kernelizations. Additionally, for both Min Ones SAT(Γ) and Max Ones SAT(Γ) there are choices of Γ such that the respective problem is hard to approximate to within a constant factor of the optimum, but still admits a polynomial kernelization.

10. Open problems and further research

In this final chapter we discuss some open problems directly related to work presented in this thesis as well as some general issues from parameterized complexity and kernelization.

Kernelization lower bounds

Currently, lower bounds for the kernelizability of parameterized problems are one of the most interesting topics in parameterized complexity.

In their lower bound framework Bodlaender et al. [16] introduced the notions of AND- respectively OR-distillation algorithms and conjectured that NP-complete problems do not have such algorithms. They showed that a polynomial kernelization for any OR- or AND-compositional parameterized problem, whose unparameterized version is NP-complete, would give such an OR- respectively AND-distillation algorithm (for the NP-complete unparameterized version). Fortnow and Santhanam [56] proved the OR-distillation conjecture, assuming NP $\not\subseteq$ co-NP/poly (i.e., if the OR-distillation conjecture does not hold then NP $\subseteq$ co-NP/poly). It is an open problem whether the AND-distillation conjecture can be proven using a similar assumption in classical complexity. Such a result would strengthen the consequence of having polynomial kernelizations for some well-studied AND-compositional problems, e.g., TREEWIDTH and PATHWIDTH. Similarly, it is an interesting problem whether the implication of having an OR-distillation for some NP-complete problem can be further strengthened.

Dell and van Melkebeek [44] provided concrete lower bounds on the total size of any kernelization (or stronger on the cost of any oracle communication protocol) for a number of parameterized problems, e.g., d-HITTING SET has no kernelization with total size $\mathcal{O}(k^{d-\epsilon})$ for any $\epsilon > 0$ assuming NP $\not\subseteq$ co-NP/poly. Note that this strong result makes no statement about the number of vertices in the problem kernel. Both the sunflower-based kernelization (see Chapter 2) as well as Abu-Khzam's [1] kernelization (based on crown-decompositions) achieve a bound of $\mathcal{O}(k^d)$ on the number of edges; the degree of this bound cannot be improved by [44]. However, the latter kernelization has a bound of $\mathcal{O}(k^{d-1})$ on the number of vertices whereas the former only has the (then trivial) bound of $\mathcal{O}(k^d)$. It is an open problem to provide a kernelization with a smaller number of vertices, or to exclude such a kernelization. As to the latter, it is an open problem to derive methods that can prove concrete polynomial lower bounds on the number of vertices or variables.

It would also be interesting to have such concrete lower bounds for the Boolean CSPs that we considered. However, the work of Dell and van Melkebeek [44] seems to relate to a notion of arity in the problems. While fixing a (finite) constraint language Γ gives us a maximum arity, there may be an equivalent constraint language Γ' that has a smaller maximum arity; consider for example $\Gamma = \{(w \vee x) \wedge (y \vee z)\}$ and $\Gamma' = \{(x \vee y)\}$: Clearly both constraint languages can implement one another (without existential or frozen existential variables), but the maximum arities are different.

General upper and lower bounds for kernelization

Recently, we have seen a few general results providing polynomial kernelizations for certain problems, e.g., consider the nice results of Bodlaender et al. [17] for problems on planar or bounded-genus graphs as well as the subsequent paper of Fomin et al. [55] for problems on graph classes excluding some fixed minor. For our understanding of problems it seems important to obtain further results of this type as well as to strengthen and simplify the existing ones (if possible), making them more accessible.

The related issue of general lower bounds appears to be quite hard. For example, if we would focus on graph problems definable in some restriction of second-order logic and restricted to planar graphs, then certainly this will also contain problems that are (trivially) polynomial-time solvable. In the study of Boolean constraint satisfaction problems we were able to do a case analysis, which would allow to later consider constraint languages that are, say, Horn but not anti-Horn, allowing reductions from hard problems (and similarly when using partial polymorphisms). This also includes that we could use the known characterization of the polynomial cases, further helping to restrict the constraint languages. An easier but less general goal is to exhibit a matching lower bound for some problem inside a certain class or restricted setting.

Some open problems in kernelization

Apart from the general open problems relating to kernelization, let us mention a few problems whose kernelizability has received significant attention. We begin with EDGE BIPARTIZATION and DIRECTED FEEDBACK VERTEX SET:

EDGE BIPARTIZATION

Input: A graph $G = (V, E)$ and an integer k.

Parameter: k.

Task: Decide whether there is a set of edges $F \subseteq E$ of size at most k such that $G - F$ is bipartite.

DIRECTED FEEDBACK VERTEX SET

Input: A directed graph $G = (V, A)$ and an integer k.

Parameter: k.

Task: Decide whether there is a set of vertices $S \subseteq V$ of size at most k such that $G - S$ contains no directed cycles.

Both problems are known to be FPT by work of Reed, Smith, and Vetta [103] for the vertex-deletion variant called GRAPH BIPARTIZATION (as observed by Guo et al. [62]) and Chen et al. [35] respectively. It is open whether they admit polynomial kernelizations.

Moreover, there are a number of graph problems which are known to admit quadratic kernelizations (i.e., with $\mathcal{O}(k^2)$ vertices) and it would be interesting to know which of them can be improved to (vertex) linear, e.g., FEEDBACK VERTEX SET and MINIMUM FILL-IN:

MINIMUM FILL-IN

Input: A graph $G = (V, E)$ and an integer k.

Parameter: k.

Task: Decide whether there is a set of at most k additional edges $F \subseteq \binom{V}{2} \setminus E$ such that $G' = (V, E \cup F)$ is chordal, i.e., G' contains no induced cycle of length at least 4.

Let us recall that Dell and van Melkebeek [44] showed that FEEDBACK VERTEX SET does not admit a kernelization with total size $\mathcal{O}(k^{2-\epsilon})$; this does not rule out a kernelization with $\mathcal{O}(k)$ vertices, but the bound of $\mathcal{O}(k^2)$ edges, as achieved by Thomassé's [110] kernelization, is essentially optimal.

Kernelizability of edge modification problems

With the results presented in Chapter 4, i.e., an H-free EDGE DELETION and an H-free EDGE EDITING problem which do not admit polynomial kernelizations, it is clear that the kernelizability of these problems will not behave as simply as their parameterized complexity. The latter was characterized by Cai [27], who showed that the following problem is fixed-parameter tractable when Π can be characterized by finitely many forbidden induced subgraphs:

Π-GRAPH MODIFICATION

Input: A graph G and integers i, j, and k.

Parameter: $i + j + k$.

Task: Decide whether it is possible to obtain a graph with property Π by deleting at most i vertices of G, adding at most j edges, and deleting at most k edges.

A simple fpt-algorithm for this problem will repeatedly search for a forbidden induced subgraph and branch on one of the finitely many possible modifications, e.g., deleting an edge, that will "destroy" the induced subgraph; the depth of the search tree is obviously bounded by $i+j+k$. (This easily extends to edge editing problems using Cai's algorithm as a black box: As a first step we may guess one of the $\ell + 1$ ways of dividing ℓ edit operations into additions and deletions.)

Concerning the kernelizability let us observe again that, when only vertex deletions are allowed, there is a straightforward reduction to d-HITTING SET (where d equals the maximum number of vertices in any forbidden induced subgraph of the characterization): We make a family of sets, containing each vertex set that induces a forbidden subgraph. Then hitting sets for this family correspond directly to the same number of vertex deletions that would remove all forbidden induced subgraphs. Observe that this approach does not work for edge modification problems, since the modifications may create new copies of forbidden induced subgraphs (exceptions include, e.g., triangle-free EDGE DELETION: edge deletions cannot create new triangles).

The kernelizability of edge modification problems is open, even if we are only interested in making graphs H-free for some fixed graph H using k edge deletions. Let us recall that making a graph H-free by deleting k edges is equivalent to making the complement graph $\bar{H}$-free by adding k edges. Thus, it suffices to consider H-free EDGE DELETION and H-free EDGE EDITING problems.

Considering H-free EDGE DELETION (or the more general case of forbidding a set of graphs, $\mathcal{H}$-free EDGE DELETION) we quickly observe that it is somewhat related to Min Ones SAT(Γ). It is possible to reduce instances of H-free EDGE DELETION to instance of Min Ones SAT(Γ) for a suitable Γ, however, the following example shows that this is not necessarily helpful for resolving the kernelizability.

Example 10.1. We consider the CLUSTER DELETION problem, where the task is to delete at most k edges to obtain a cluster graph, i.e., a P_3-free graph (containing no induced path on three vertices). Let us try to translate a given graph into an equivalent formula, i.e., iff there are k edge deletions that make the graph P_3-free then the formula shall have a satisfying assignment with at most k true variables. We make a variable for each edge with the intention that assigning true is equivalent to deleting the edge:

- For any three vertices which have at most one edge between them we do not require any constraint since no edge deletions can create a P_3 between these vertices.
- For three vertices which induce a P_3 we know that at least one of the two edges must be deleted, thus we make disjunction $(x \vee y)$ on the variables corresponding to the two edges. If this constraint is satisfied then the three vertices cannot form a P_3 (there will be at most one edge between them).
- If three vertices of the graph form a triangle, then we want to avoid deleting exactly one of the three edges. Thus we make a not-1-in-3 constraint $(x+y+z \neq 1)$ for its three edges; it is feasible to make no deletions or to delete at least two of the edges. Recall that we have already used this constraint when showing a lower bound for kernelizations of the H-free EDGE DELETION problem in Chapter 4.

Thus P_3-free EDGE DELETION reduces to Min Ones SAT($\{(x \vee y), (x+y+z \neq 1)\}$) by a polynomial parameter transformation. However, the former problem admits a polynomial kernelization (in fact one with a linear number of vertices [61]) whereas the latter does not unless NP $\subseteq$ co-NP/poly. This follows from an immediate polynomial parameter transformation from Min Ones SAT($\{(x), (x+y+z \neq 1)\}$) for which we showed a lower bound in Chapter 4. (It is also easy to verify that $(x + y + z \neq 1)$ is not mergeable, and that Min Ones SAT($\{(x \vee y), (x + y + z \neq 1)\}$) is NP-complete.)

Thus the lower bounds obtained for Min Ones SAT(Γ) problems do not easily carry over to H-free EDGE DELETION problems (even though in the example we used a seemingly minimal selection of constraints). However, as seen in Chapter 4, they can be used by giving careful reductions from Min Ones SAT(Γ) to some H-free EDGE DELETION problem. The work presented in Chapter 6 provides a wide selection of possible source problems that do not admit polynomial kernelizations (unless NP $\subseteq$ co-NP/poly). Still, one should clearly not expect to prove all lower bounds for H-free EDGE DELETION in this way, the structure of graphs H obtained by similar reductions would likely be too restricted. Also, a general upper bound result for these problems, beyond the cases subsumed by MIN $F^{+}\Pi_1$, seems nowhere in sight.

Concerning the case of more than one forbidden subgraph, let us point out that, while Min Ones SAT(Γ') is always at least as hard as Min Ones SAT(Γ) for any $\Gamma' \supseteq \Gamma$, an edge deletion problem may actually become simpler: Compare the complexity of making a graph H-free to that of making it $\{H, K_2\}$-free (i.e., making it an independent set). Similarly, using the graph H as in Chapter 4, H-free EDGE DELETION does not admit a polynomial kernelization, whereas $\{H, P_3\}$-free admits a polynomial kernelization (trivially since $P_3 \leq H$). The crucial difference and difficulty is that constraints of a formula exist independently of one another whereas the edges of a graph and added modifications to the edge set give rise to forbidden subgraphs. A rigorous study of these problems may require to first express the forbidden induced subgraphs in a different way, e.g., at hand of the forbidden adjacency patterns of c vertices, where c equals the number vertices in the largest forbidden induced subgraph.

Parameterized complexity of Min UnSAT(Γ)

In Chapters 6 and 7 we have obtained complete characterizations of the polynomial kernelizability of Min Ones SAT(Γ) and Max Ones SAT(Γ) problems, respectively. Let us also recall that Max SAT(Γ) problems admit polynomial kernelizations for any Γ, as Max SAT(Γ) problems can be expressed as MAX SNP problems (cf. [75]). Of the four basic optimization problems for Boolean constraint satisfaction problems this leaves the kernelizability of Min UnSAT(Γ) as an open problem. Note however, that for these problems even the parameterized complexity (say a classification into FPT and W[1]-hard cases) is open. So far we know only a few positive results for certain languages Γ, e.g., Min UnSAT($\{(x \neq y)\}$) is EDGE BIPARTIZATION, Min UnSAT($\{(x = y), (x \neq y)\}$) is BALANCED SUBGRAPH:

BALANCED SUBGRAPH

Input: A graph $G = (V, E)$ with edges labeled $=$ or $\neq$ and an integer k.

Parameter: k.

Task: Decide whether there is a two-coloring of the vertices that respects all but at most k of the edge labels.

EDGE BIPARTIZATION is FPT (Reed et al. [103]) and BALANCED SUBGRAPH easily reduces to it (Hüffner et al. [69]). Razgon and O'Sullivan [102] gave a more general result, by showing that Almost 2SAT, i.e., Min UnSAT($\{(x \vee y), (x \vee \neg y), (\neg x \vee \neg y)\}$), is fixed-parameter tractable. As to kernelizability of Min UnSAT(Γ) problems, let us recall that even the existence of a polynomial kernelization for EDGE BIPARTIZATION (Min UnSAT($\{(x \neq y)\}$)) is an open problem. It is likely that Min UnSAT(Γ) is W[1]-hard already for fairly simple choices of Γ and that only few cases admit polynomial kernelizations (modulo standard complexity assumptions). To add to this intuition let us state Khanna et al.'s [75] approximability characterization for Min UnSAT(Γ) problems.

Theorem 10.2 ([75])**.** *Let Γ be a finite set of Boolean relations.*

- *If Γ is one-valid, zero-valid, or 2-monotone then* Min UnSAT(Γ) *is in* P.
- *Else, if Γ is IHS-B then* Min UnSAT(Γ) *is* APX*-complete.*
- *Else, if Γ is width-2 affine then* Min UnSAT(Γ) *is* MIN UNCUT*-complete.*
- *Else, if Γ is bijunctive then* Min UnSAT(Γ) *is* MIN 2CNF DELETION*-complete.*
- *Else, if Γ is affine then* Min UnSAT(Γ) *is* NEAREST CODEWORD*-complete.*
- *Else, if Γ is Horn then* Min UnSAT(Γ) *is* MIN HORN DELETION*-complete.*
- *Else,* Min UnSAT(Γ) *is not approximable.*

Remark. MIN UNCUT is the optimization variant of EDGE BIPARTIZATION and MIN 2CNF DELETION corresponds to Almost 2SAT.

Already without a full discussion of the different degrees of inapproximability it is quite apparent that Min UnSAT(Γ) problems are hard to tackle even for simple constraint languages Γ. On a positive note at least, we may restrict our attention to constraint languages Γ which are zero-valid, one-valid, Horn, anti-Horn, affine, or bijunctive: For other choices of Γ already satisfiability of formulas over Γ, i.e., SAT(Γ), is NP-hard; asking whether $k = 0$ deletions suffice is equivalent to SAT(Γ), thus Min UnSAT(Γ) is not in XP (unless P = NP).

Outlook

In the past few years some deeply influencing results about kernelization were uncovered. There are meta-theorems, essentially tight lower bounds, and connections to classical complexity. Nonetheless there are still lots of fascinating open problems, and we should also expect more general results. Also, in reaction to the recent lower bound developments, there will likely be more research on relaxations of kernelization, e.g., disjunctive kernelizations. Similarly, all the new results put wear and tear onto the definitions of this fairly new area, and it may be time to carefully assess the influence of some details that are not yet set in stone.

Bibliography

[1] Faisal N. Abu-Khzam. Kernelization algorithms for d-hitting set problems. In Dehne et al. [43], pages 434–445. 6, 24, 32, 39, 129

[2] Jochen Alber, Michael R. Fellows, and Rolf Niedermeier. Polynomial-time data reduction for dominating set. *Journal of the ACM*, 51(3):363–384, 2004. 6

[3] Susanne Albers and Jean-Yves Marion, editors. *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, volume 09001 of *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, Germany, 2009. 136, 139, 141

[4] Gunnar Andersson and Lars Engebretsen. Better approximation algorithms for SET SPLITTING and NOT-ALL-EQUAL SAT. *Information Processing Letters*, 65(6):305–311, 1998. 8

[5] Sanjeev Arora, László Babai, Jacques Stern, and Elizabeth Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54(2):317–331, 1997. 127

[6] Sanjeev Arora, David R. Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. *Journal of Computer and System Sciences*, 58(1):193–210, 1999. 34

[7] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. 33

[8] Takao Asano, Kuniaki Hori, Takao Ono, and Tomio Hirata. A theoretical framework of hybrid approaches to MAX SAT. In Hon Wai Leong, Hiroshi Imai, and Sanjay Jain, editors, *ISAAC*, volume 1350 of *Lecture Notes in Computer Science*, pages 153–162. Springer, 1997. 8

[9] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999. 8

[10] Cristina Bazgan. Schémas d'approximation et complexité paramétrée, Rapport du stage (DEA). Technical report, Universitée Paris Sud, 1995. 8

[11] Stéphane Bessy, Fedor V. Fomin, Serge Gaspers, Christophe Paul, Anthony Perez, Saket Saurabh, and Stéphan Thomassé. Kernels for feedback arc set in tournaments. In Ravi Kannan and K. Narayan Kumar, editors, *FSTTCS*, volume 4 of *LIPIcs*, pages 37–47. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2009. 6

[12] Daniel Bienstock and Michael A. Langston. Chapter 8 Algorithmic implications of the graph minor theorem. In C.L. Monma M.O. Ball, T.L. Magnanti and G.L. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 481 – 502. Elsevier, 1995. 16

[13] Andreas Björklund and Thore Husfeldt. Exact algorithms for exact satisfiability and number of perfect matchings. *Algorithmica*, 52(2):226–249, 2008. 93

[14] Hans L. Bodlaender. Kernelization: New upper and lower bound techniques. In Chen and Fomin [31], pages 17–37. 6, 18

[15] Hans L. Bodlaender, Leizhen Cai, Jianer Chen, Michael R. Fellows, Jan Arne Telle, and Dániel Marx. Open problems in parameterized and exact computation – IWPEC 2006, 2006. 51, 123

[16] Hans L. Bodlaender, Rodney G. Downey, Michael R. Fellows, and Danny Hermelin. On problems without polynomial kernels. *Journal of Computing and System Sciences*, 75(8):423–434, 2009. 7, 8, 21, 22, 24, 51, 53, 129

[17] Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos. (Meta) kernelization. In *FOCS*, pages 629–638. IEEE Computer Society, 2009. 6, 7, 21, 130

[18] Hans L. Bodlaender, Stéphan Thomassé, and Anders Yeo. Kernel bounds for disjoint cycles and disjoint paths. In Amos Fiat and Peter Sanders, editors, *ESA*, volume 5757 of *Lecture Notes in Computer Science*, pages 635–646. Springer, 2009. 7, 23, 94

[19] Elmar Böhler, Nadia Creignou, Steffen Reith, and Heribert Vollmer. Playing with Boolean blocks, Part I: Post's lattice with applications to complexity theory. *ACM-SIGACT News*, 34(4):3–52, 2003. Complexity Theory Column 42. 74

[20] Elmar Böhler, Nadia Creignou, Steffen Reith, and Heribert Vollmer. Playing with Boolean blocks, Part II: Constraint satisfaction problems. *ACM-SIGACT News*, 35(1):22–35, 2004. Complexity Theory Column 43. 74

[21] Elmar Böhler, Steffen Reith, Henning Schnoor, and Heribert Vollmer. Bases for Boolean co-clones. *Information Processing Letters*, 96(2):59–66, 2005. 71, 72

[22] Nicolas Bousquet, Jean Daligault, Stéphan Thomassé, and Anders Yeo. A polynomial kernel for multicut in trees. In Albers and Marion [3], pages 183–194. 6

[23] Daniel Brügmann, Christian Komusiewicz, and Hannes Moser. On generating triangle-free graphs. *Electronic Notes in Discrete Mathematics*, 32:51–58, 2009. 51, 52

[24] Andrei A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *FOCS*, pages 649–658. IEEE, 2002. 66

[25] Andrei A. Bulatov. Tractable conservative constraint satisfaction problems. In *LICS*, page 321. IEEE, 2003. 66

[26] Pablo Burzyn, Flavia Bonomo, and Guillermo Durán. NP-completeness results for edge modification problems. *Discrete Applied Mathematics*, 154(13):1824–1844, 2006. 51, 52

[27] Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996. 52, 131

[28] Liming Cai and Jianer Chen. On fixed-parameter tractability and approximability of NP optimization problems. *Journal of Computer and System Sciences*, 54(3):465–474, 1997. 34

[29] Marco Cesati and Luca Trevisan. On the efficiency of polynomial time approximation schemes. *Electronic Colloquium on Computational Complexity (ECCC)*, 4(1), 1997. 8

[30] Hubie Chen. A rendezvous of logic, complexity, and algebra. *ACM Computing Surveys*, 42(1), 2009. 74

[31] Jianer Chen and Fedor V. Fomin, editors. *Parameterized and Exact Computation, 4th International Workshop, IWPEC 2009, Copenhagen, Denmark, September 10-11, 2009, Revised Selected Papers*, volume 5917 of *Lecture Notes in Computer Science*. Springer, 2009. 136, 141

[32] Jianer Chen, Xiuzhen Huang, Iyad A. Kanj, and Ge Xia. Linear FPT reductions and computational lower bounds. In László Babai, editor, *STOC*, pages 212–221. ACM, 2004. 5

[33] Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex cover: Further observations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001. 6, 8, 24

[34] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved parameterized upper bounds for vertex cover. In Rastislav Kralovic and Pawel Urzyczyn, editors, *MFCS*, volume 4162 of *Lecture Notes in Computer Science*, pages 238–249. Springer, 2006. 5

[35] Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. In Ladner and Dwork [83], pages 177–186. 131

[36] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. *Complexity Classifications of Boolean Constraint Satisfaction Problems*, volume 7 of *SIAM Monographs on Discrete Mathematics and Applications.* 2001. 66, 68, 74

[37] Nadia Creignou, Phokion G. Kolaitis, and Heribert Vollmer, editors. *Complexity of Constraints - An Overview of Current Research Themes [Result of a Dagstuhl Seminar]*, volume 5250 of *Lecture Notes in Computer Science.* Springer, 2008. 74, 138

[38] Nadia Creignou, Phokion G. Kolaitis, and Bruno Zanuttini. Structure identification of Boolean relations and plain bases for co-clones. *Journal of Computer and System Sciences*, 74(7):1103–1115, 2008. 71, 72, 78, 111, 112, 114

[39] Nadia Creignou, Henning Schnoor, and Ilka Schnoor. Non-uniform Boolean constraint satisfaction problems with cardinality constraint. In *CSL*, volume 5213 of *Lecture Notes in Computer Science*, pages 109–123. Springer, 2008. 66, 107, 110

[40] Nadia Creignou and Heribert Vollmer. Boolean constraint satisfaction problems: When does Post's lattice help? In Creignou et al. [37], pages 3–37. 72

[41] Pierluigi Crescenzi and Viggo Kann. Approximation on the web: A compendium of NP optimization problems. In José D. P. Rolim, editor, *RANDOM*, volume 1269 of *Lecture Notes in Computer Science*, pages 111–118. Springer, 1997. 13

[42] Pierluigi Crescenzi and Gianluca Rossi. On the Hamming distance of constraint satisfaction problems. *Theoretical Computer Science*, 288(1):85–100, 2002. 66

[43] Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Norbert Zeh, editors. *Algorithms and Data Structures, 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007, Proceedings*, volume 4619 of *Lecture Notes in Computer Science.* Springer, 2007. 135, 140

[44] Holger Dell and Dieter van Melkebeek. Satisfiability allows no non-trivial sparsification unless the polynomial hierarchy collapses. In *STOC*, 2010. 7, 21, 24, 129, 131

[45] Josep Díaz and Dimitrios M. Thilikos. Fast FPT-algorithms for cleaning grids. In Bruno Durand and Wolfgang Thomas, editors, *STACS*, volume 3884 of *Lecture Notes in Computer Science*, pages 361–371. Springer, 2006. 51

[46] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Incompressibility through colors and IDs. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *ICALP (1)*, volume 5555 of *Lecture Notes in Computer Science*, pages 378–389. Springer, 2009. 7, 8, 23, 49, 86

[47] Rod G. Downey and M. R. Fellows. *Parameterized Complexity (Monographs in Computer Science).* Springer, November 1998. 15, 18, 107

[48] Paul Erdős and Richard Rado. Intersection theorems for systems of sets. *Journal of the London Mathematical Society*, 35:85–90, 1960. 27

[49] Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity of Computation*, volume 7 of *SIAM-AMS Proceedings*, pages 43–73, SIAM, Philadelphia, PA, 1974. 33

[50] Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1999. 66

[51] Uriel Feige, MohammadTaghi Hajiaghayi, and James R. Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM Journal on Computing*, 38(2):629–657, 2008. 8

[52] Michael R. Fellows, Daniel Lokshtanov, Neeldhara Misra, Matthias Mnich, Frances A. Rosamond, and Saket Saurabh. The complexity ecology of parameters: An illustration using bounded max leaf number. *Theory of Computing Systems*, 45(4):822–848, 2009. 18

[53] Henning Fernau, Fedor V. Fomin, Daniel Lokshtanov, Daniel Raible, Saket Saurabh, and Yngve Villanger. Kernel(s) for problems with no kernel: On out-trees with many leaves. In Albers and Marion [3], pages 421–432. 7, 23, 126

[54] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory (Texts in Theoretical Computer Science. An EATCS Series).* Springer, March 2006. 15, 18, 31, 39

[55] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Dimitrios Thilikos. Bidimensionality and kernels. In *SODA*, 2010. To appear. 7, 130

[56] Lance Fortnow and Rahul Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In Ladner and Dwork [83], pages 133–142. 7, 22, 129

[57] Alan M. Frieze and Ravi Kannan. The regularity lemma and approximation schemes for dense problems. In *FOCS*, pages 12–20, 1996. 34

[58] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979. 8, 17

[59] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995. 8

[60] Jiong Guo. Problem kernels for NP-complete edge deletion problems: Split and related graphs. In Takeshi Tokuyama, editor, *ISAAC*, volume 4835 of *Lecture Notes in Computer Science*, pages 915–926. Springer, 2007. 51, 52

[61] Jiong Guo. A more effective linear kernelization for cluster editing. *Theoretical Computer Science*, 410(8-10):718–726, 2009. 51, 52, 132

[62] Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computer and System Sciences*, 72(8):1386–1396, 2006. 131

[63] Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1):31–45, 2007. 6, 18

[64] Jiong Guo and Rolf Niedermeier. Linear problem kernels for NP-hard problems on planar graphs. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP*, volume 4596 of *Lecture Notes in Computer Science*, pages 375–386. Springer, 2007. 7

[65] Jiong Guo and Johannes Uhlmann. Kernelization and complexity results for connectivity augmentation problems. In Dehne et al. [43], pages 483–494. 51

[66] Eran Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. *SIAM Journal on Computing*, 31(5):1608–1623, 2002. 8

[67] Danny Harnik and Moni Naor. On the compressibility of NP instances and cryptographic applications. *SIAM Journal on Computing*, 39(5):1667–1713, 2010. 23

[68] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *FOCS*, pages 627–636, 1996. 33

[69] Falk Hüffner, Nadja Betzler, and Rolf Niedermeier. Optimal edge deletions for signed graph balancing. In Camil Demetrescu, editor, *WEA*, volume 4525 of *Lecture Notes in Computer Science*, pages 297–310. Springer, 2007. 134

[70] Cor A. J. Hurkens and Alexander Schrijver. On the size of systems of sets every t of which have an SDR, with an application to the worst-case ratio of heuristics for packing problems. *SIAM Journal on Discrete Mathematics*, 2(1):68–72, 1989. 8

[71] Peter Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1-2):185–204, 1998. 72

[72] Stasys Jukna. *Extremal Combinatorics.* Springer, 2001. 28

[73] Marek Karpinski. Polynomial time approximation schemes for some dense instances of NP-hard optimization problems. *Algorithmica*, 30(3):386–397, 2001. 34

[74] Sanjeev Khanna, Rajeev Motwani, Madhu Sudan, and Umesh V. Vazirani. On syntactic versus computational views of approximability. *SIAM Journal on Computing*, 28(1):164–191, 1998. 34, 127

[75] Sanjeev Khanna, Madhu Sudan, Luca Trevisan, and David P. Williamson. The approximability of constraint satisfaction problems. *SIAM Journal on Computing.*, 30(6):1863–1920, 2000. 54, 55, 66, 68, 75, 85, 94, 97, 98, 110, 114, 127, 128, 133, 134

[76] Phokion G. Kolaitis and Madhukar N. Thakur. Logical definability of NP optimization problems. *Information and Computation*, 115(2):321–353, 1994. 34, 37, 124, 126

[77] Phokion G. Kolaitis and Madhukar N. Thakur. Approximation properties of NP minimization classes. *Journal of Computer and System Sciences*, 50(3):391–411, 1995. 34, 37, 124, 126, 127

[78] Stefan Kratsch. Polynomial kernelizations for MIN $F^+\Pi_1$ and MAX NP. In Albers and Marion [3], pages 601–612. 34

[79] Stefan Kratsch, Dániel Marx, and Magnus Wahlström. Parameterized complexity and kernelizability of max ones and exact ones problems. In *MFCS*, 2010. To appear. 98, 108

[80] Stefan Kratsch and Magnus Wahlström. Two edge modification problems without polynomial kernels. In Chen and Fomin [31], pages 264–275. 23, 53

[81] Stefan Kratsch and Magnus Wahlström. Preprocessing of min ones problems: A dichotomy. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *ICALP (1)*, volume 6198 of *Lecture Notes in Computer Science*, pages 653–665. Springer, 2010. 75

[82] Richard E. Ladner. On the structure of polynomial time reducibility. *Journal of the Association for Computing Machinery*, 22:155–171, 1975. 66

[83] Richard E. Ladner and Cynthia Dwork, editors. *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008.* ACM, 2008. 137, 139

[84] Dietlinde Lau. *Function Algebras on Finite Sets: Basic Course on Many-Valued Logic and Clone Theory (Springer Monographs in Mathematics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. 71, 72, 73

[85] Daniel Lokshtanov and Christian Sloper. Fixed parameter set splitting, linear kernel and improved running time. In Hajo Broersma, Matthew Johnson 0002, and Stefan Szeider, editors, *ACiD*, volume 4 of *Texts in Algorithmics*, pages 105–113. King's College, London, 2005. 8

[86] Meena Mahajan and Venkatesh Raman. Parameterizing above guaranteed values: Maxsat and maxcut. *Journal of Algorithms*, 31(2):335–354, 1999. 8, 34

[87] Meena Mahajan, Venkatesh Raman, and Somnath Sikdar. Parameterizing MAX SNP problems above guaranteed values. In Hans L. Bodlaender and Michael A. Langston, editors, *IWPEC*, volume 4169 of *Lecture Notes in Computer Science*, pages 38–49. Springer, 2006. 34

[88] Dániel Marx. Parameterized complexity of constraint satisfaction problems. *Computational Complexity*, 14(2):153–183, 2005. 30, 66, 107, 108

[89] Dániel Marx. Parameterized complexity and approximation algorithms. *Computer Journal*, 51(1):60–78, 2008. 18

[90] Hannes Moser. A problem kernelization for graph packing. In Mogens Nielsen, Antonín Kucera, Peter Bro Miltersen, Catuscia Palamidessi, Petr Tuma, and Frank D. Valencia, editors, *SOFSEM*, volume 5404 of *Lecture Notes in Computer Science*, pages 401–412. Springer, 2009. 6, 8

[91] Assaf Natanzon, Ron Shamir, and Roded Sharan. A polynomial approximation algorithm for the minimum fill-in problem. *SIAM Journal on Computing*, 30(4):1067–1079, 2000. 8, 52

[92] Assaf Natanzon, Ron Shamir, and Roded Sharan. Complexity classification of some edge modification problems. *Discrete Applied Mathematics*, 113(1):109–128, 2001. 51, 52

[93] George L. Nemhauser and Leslie E. Trotter. Vertex packings: Structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975. 6

[94] Rolf Niedermeier. *Invitation to Fixed Parameter Algorithms (Oxford Lecture Series in Mathematics and Its Applications)*. Oxford University Press, USA, March 2006. 18

[95] Rolf Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In Jean-Yves Marion and Thomas Schwentick, editors, *STACS*, volume 5 of *LIPIcs*, pages 17–32. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010. 18

[96] Gustav Nordh and Bruno Zanuttini. Frozen Boolean partial co-clones. In *ISMVL*, pages 120–125. IEEE Computer Society, 2009. 74, 98, 103, 104, 105

[97] Alessandro Panconesi and Desh Ranjan. Quantifiers and approximation. *Theoretical Computer Science*, 107(1):145–163, 1993. 33

[98] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, November 1993. 37

[99] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991. 8, 33, 34, 37, 44

[100] Reinhard Pöschel and Lev Arkad'evich Kalužnin. *Funktionen- und Relationenalgebren. Ein Kapitel der diskreten Mathematik*. VEB Deutscher Verlag der Wissenschaften, Berlin, 1979. 71, 72

[101] Emil Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941. 70, 72

[102] Igor Razgon and Barry O'Sullivan. Almost 2-SAT is fixed-parameter tractable. *Journal of Computer and System Sciences*, 75(8):435–450, 2009. 134

[103] Bruce A. Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004. 131, 134

[104] Neil Robertson and Paul D. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004. 16

[105] Donald J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. *Graph Theory and Computing*, pages 183–217, 1972. 51

[106] Carla D. Savage. Depth-first search and the vertex cover problem. *Information Processing Letters*, 14(5):233–237, 1982. 8

[107] Thomas J. Schaefer. The complexity of satisfiability problems. In *STOC*, pages 216–226, New York, NY, USA, 1978. ACM. 66, 69, 90

[108] Roded Sharan, Adi Maron-Katz, and Ron Shamir. CLICK and EXPANDER: A system for clustering and visualizing gene expression data. *Bioinformatics*, 19(14):1787–1799, 2003. 51

[109] David Simchi-Levi. New worst-case results for the bin-packing problem. *Naval Research Logistics*, 41:579–585, 1994. 8

[110] Stéphan Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Transactions on Algorithms*, 6(2), 2010. 6, 8, 25, 131

[111] Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Information Processing Letters*, 109(6):315–318, 2009. 7, 22

[112] Gerhard J. Woeginger. Exact algorithms for NP-hard problems: A survey. In Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi, editors, *Combinatorial Optimization*, volume 2570 of *Lecture Notes in Computer Science*, pages 185–208. Springer, 2001. 5

[113] Chee-Keng Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26:287–300, 1983. 7, 22, 24