INSTITUT FÜR INFORMATIONSSYSTEME
ABTEILUNG WISSENSBASIERTE SYSTEME

FIXED-PARAMETER TRACTABLE REDUCTIONS TO SAT

Ronald de Haan   Stefan Szeider

INFSYS RESEARCH REPORT 1843-14-04
DECEMBER 2014

Institut für Informationssysteme
Abtg. Wissensbasierte Systeme
Technische Universität Wien
Favoritenstraße 9-11
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at

INFSYS Research Report
INFSYS Research Report 1843-14-04, December 2014

Fixed-Parameter Tractable Reductions to SAT

Ronald de Haan[1*] Stefan Szeider[1*]

Abstract. Today's SAT solvers have an enormous importance and impact in many practical settings. They are used as efficient back-end to solve many NP-complete problems. However, many computational problems are located at the second level of the Polynomial Hierarchy or even higher, and hence polynomial-time transformations to SAT are not possible, unless the hierarchy collapses. In certain cases one can break through these complexity barriers by fixed-parameter tractable (fpt) reductions which exploit structural aspects of problem instances in terms of problem parameters. Recent research established a general theoretical framework that supports the classification of parameterized problems on whether they admit such an fpt-reduction to SAT or not. We use this framework to analyze some problems that are related to Boolean satisfiability. We consider several natural parameterizations of these problems, and we identify for which of these an fpt-reduction to SAT is possible. The problems that we look at are related to minimizing an implicant of a DNF formula, minimizing a DNF formula, and satisfiability of quantified Boolean formulas.

[1] Institute of Information Systems, Vienna University of Technology, Vienna, Austria
[*] Supported by the European Research Council (ERC), project 239962 (COMPLEX REASON), and the Austrian Science Fund (FWF), project P26200 (Parameterized Compilation).

Publication Information: This is the authors' self-archived copy of a paper that appeared in the *Proceedings of the 17th International Symposium on the Theory and Applications of Satisfiability Testing (SAT 2014)*, dx.doi.org/10.1007/978-3-319-09284-3_8. The final publication is available at www.springer.com.

1 Introduction

Modern SAT solvers have an enormous importance and impact in many practical settings that require solutions to NP-complete problems. In fact, due to the success of SAT, NP-complete problems have lost their scariness, as in many cases one can efficiently encode NP-complete problems to SAT and solve them by means of a SAT solver [8, 21, 32, 39]. However, many important computational problems are located above the first level of the Polynomial Hierarchy (PH) and thus considered significantly "harder" than SAT. Hence we cannot hope for polynomial-time reductions from these problems to SAT, as such transformations would cause the (unexpected) collapse of the PH.

Realistic problem instances are not random and often contain some kind of structure. Recent research succeeded to exploit such structure to break the complexity barriers between levels of the PH [16, 37]. The idea is to exploit problem structure in terms of a problem *parameter*, and to develop reductions to SAT that can be computed efficiently as long as the problem parameter is reasonably small. The theory of *parameterized complexity* [13, 18, 34] provides exactly the right type of reduction suitable for this purpose, called *fixed-parameter tractable reductions*, or *fpt-reductions* for short. Now, for a suitable choice of the parameter, one can aim at developing fpt-reductions from the hard problem under consideration to SAT.

Such positive results go significantly beyond the state-of-the-art of current research in parameterized complexity. By shifting the scope from fixed-parameter tractability to fpt-reducibility (to SAT), parameters can be less restrictive and hence larger classes of inputs can be processed efficiently. Therefore, the potential for positive tractability results is greatly enlarged. In fact, there are some known reductions that, in retrospect, can be seen as fpt-reductions to SAT. A prominent example is Bounded Model Checking [6, 7], which can be seen as an fpt-reduction from the model checking problem for linear temporal logic (LTL), which is PSPACE-complete, to SAT, where the parameter is an upper bound on the size of a counterexample.

Recently, extending the work of Flum and Grohe [17], we initiated the development of a general theoretical framework to support the classification of hard problems on whether they admit an fpt-reduction to SAT or not [25]. This framework provides a hardness theory that can be used to provide evidence that certain problems do not admit an fpt-reduction to SAT, similar to NP-hardness which provides evidence against polynomial-time tractability [19] and W[1]-hardness which provides evidence against fixed-parameter tractability [13]. For an overview of the parameterized complexity classes in this framework and the relation between them, see Figure 1.

New Contributions We use this new framework to analyze problems related to Boolean satisfiability. We focus on problems that are located at the second level of the PH, i.e., problems complete for Σ^{p}_2. This initiates a structured investigation of fpt-reducibility to SAT of problems related to Boolean satisfiability that are "beyond NP." Concretely, we look at the following problems, consider several parameterizations of these problems, and identify for which of these parameterized problems an fpt-reduction to SAT is possible and for which this is not possible:

- minimizing an implicant of a formula in disjunctive normal form (DNF) (parameterizations: the size of the minimized implicant, and the difference in size between the original and the minimized implicant);
- minimizing a DNF formula (parameterizations: the size of the minimized formula, and the difference in size between the original and the minimized formula); and
- the satisfiability problem of quantified Boolean formulas (QBFs) (parameterizations: the treewidth of the incidence graph of the formula restricted to several subsets of variables).

In particular, we show that minimizing an implicant of a DNF formula does not become significantly easier when the minimized implicant is small (Proposition 2), nor when the difference in size between the original and the minimized implicant is small (Proposition 3). The problem of reducing a DNF formula in size (while preserving logical equivalence) also does not become significantly easier when the difference in size is small (Proposition 4). However, the problem of reducing a DNF formula to an equivalent DNF formula that is small can be done with a small number of SAT calls (Theorem 9). Moreover, we show that deciding satisfiability of a quantified Boolean formula with one quantifier alternation can be reduced to a single SAT instance if the variables in the second quantifier block interact with each other in a tree-like fashion (Theorem 11), whereas a similar restriction on the variables in the first quantifier block does not make the problem any easier (Proposition 10).

Related work Many of the decision problems analyzed in this paper have been studied before in a classical complexity setting [20, 42, 44]. The logic minimization problems that we consider in this paper have been studied since the 1950s (cf. [44]). The problem of minimizing an implicant of a DNF formula plays a central role in the analysis of logic minimization problems [44]. Variants of the minimization problems that we consider, where a subset-minimal solution is sought, are often solved by calling SAT solvers as subroutines. One example of such work is related to identifying minimal unsatisfiable subsets (MUSes) of a CNF formula [3, 24, 33]. Recent work on MUS extraction indicates that reducing the number of SAT calls made in these algorithms is beneficial for the practical performance of these algorithms [33]. Decision procedures using SAT solvers as a subroutine have also been used to solve problems that lie at the second level of the PH, e.g., problems related to abstract argumentation [15].

2 Preliminaries

Propositional Logic and Quantified Boolean Formulas A *literal* is a propositional variable x or a negated variable $\neg x$. The *complement* $\overline{x}$ of a positive literal x is $\neg x$, and the complement $\overline{\neg x}$ of a negative literal $\neg x$ is x. For literals $l \in \{x, \neg x\}$, we let $\text{Var}(l) = x$ denote the variable occurring in l. A *clause* is a finite set of literals, not containing a complementary pair x, $\neg x$, and is interpreted as the disjunction of these literals. A *term* is a finite set of literals, not containing a complementary pair x, $\neg x$, and is interpreted as the conjunction of these literals. We let $\top$ denote the empty clause. A formula in *conjunctive normal form (CNF)* is a finite set of clauses, interpreted as the conjunction of these clauses. A formula in *disjunctive normal form (DNF)* is a finite set of terms, interpreted as the disjunction of these terms. We say that a DNF formula φ is a *term-wise subformula* of another DNF formula φ' if for all terms $t \in \varphi$ there exists a term $t' \in \varphi'$ such that $t \subseteq t'$. We define the *size* $\|\varphi\|$ of a DNF formula φ to be $\sum_{t \in \varphi} |t|$; the number of terms of φ is denoted by $|\varphi|$. For a DNF formula φ, the set $\text{Var}(\varphi)$ denotes the set of all variables x such that some term of φ contains x or $\neg x$. We use the standard notion of *(truth) assignments* $\alpha : \text{Var}(\varphi) \to \{0, 1\}$ for Boolean formulas and *truth* of a formula under such an assignment. We denote the problem of deciding whether a propositional formula φ is satisfiable by SAT, and the problem of deciding whether φ is not satisfiable by UNSAT. Let φ be a DNF formula. We say that a set C of literals is an *implicant of* φ if all assignments that satisfy $\bigwedge_{l \in C} l$ also satisfy φ. Let $\gamma = \{x_1 \mapsto d_1, \dots, x_n \mapsto d_n\}$ be a function that maps some variables of a formula φ to other variables or to truth values. We let $\varphi[\gamma]$ denote the application of such a substitution γ to the formula φ. We also write $\varphi[x_1 \mapsto d_1, \dots, x_n \mapsto d_n]$ to denote $\varphi[\gamma]$.

A *(prenex) quantified Boolean formula (QBF)* is a formula of the form $Q_1 X_1 Q_2 X_2 \dots Q_m X_m \psi$, where each Q_i is either $\forall$ or $\exists$, the X_i are disjoint sets of propositional variables, and ψ is a Boolean formula over the variables in $\bigcup_{i=1}^{m} X_i$. We call ψ the *matrix* of the formula. Truth of such formulas is defined in the usual way. We say that a QBF is *in QDNF* if the matrix is in DNF. For the remainder of this paper, we will restrict our attention to QDNF formulas. Consider the following decision problem.

$\exists\forall$-SAT
Instance: A QDNF $\varphi = \exists X.\forall Y.\psi$, where ψ is quantifier-free.
Question: Is φ satisfiable?

The complexity class consisting of all problems that are polynomial-time reducible to $\exists\forall$-SAT is denoted by Σ_2^{p}, and its co-class is denoted by Π_2^{p}. These classes form the second level of the PH [36].

Parameterized Complexity We introduce some core notions from parameterized complexity theory. For an in-depth treatment we refer to other sources [13, 18, 34]. A *parameterized problem* L is a subset of $\Sigma^* \times \mathbb{N}$ for some finite alphabet Σ. For an instance $(I, k) \in \Sigma^* \times \mathbb{N}$, we call I the *main part* and k the *parameter*. The following generalization of polynomial time computability is commonly regarded as the tractability notion of parameterized complexity theory. A parameterized problem L is *fixed-parameter tractable* if there exists a computable function f and a constant c such that there exists an algorithm that decides whether $(I, k) \in L$ in time $O(f(k)\|I\|^c)$, where $\|I\|$ denotes the size of I. Such an algorithm is called an *fpt-algorithm*, and this amount of time is called *fpt-time*. FPT is the class of all fixed-parameter tractable decision problems. If the parameter is constant, then fpt-algorithms run in polynomial time where the order of the polynomial is independent of the parameter. This provides a good scalability in the parameter in contrast to running times of the form $\|I\|^k$, which are also polynomial for fixed k, but are already impractical for, say, $k > 3$. By XP we denote the class of all problems L for which it can be decided whether $(I, k) \in L$ in time $O(\|I\|^{f(k)})$, for some fixed computable function f.

Parameterized complexity also generalizes the notion of polynomial-time reductions. Let $L \subseteq \Sigma^* \times \mathbb{N}$ and $L' \subseteq (\Sigma')^* \times \mathbb{N}$ be two parameterized problems. A *(many-one) fpt-reduction* from L to L' is a mapping $R : \Sigma^* \times \mathbb{N} \to (\Sigma')^* \times \mathbb{N}$ from instances of L to instances of L' such that there exist some computable function $g : \mathbb{N} \to \mathbb{N}$ such that for all $(I, k) \in \Sigma^* \times \mathbb{N}$: (i) (I, k) is a yes-instance of L if and only if $(I', k') = R(I, k)$ is a yes-instance of L', (ii) $k' \leq g(k)$, and (iii) R is computable in fpt-time. Similarly, we call reductions that satisfy properties (i) and (ii) but that are computable in time $O(\|I\|^{f(k)})$, for some fixed computable function f, *xp-reductions.*

Let C be a classical complexity class, e.g., NP. The parameterized complexity class para-C is then defined as the class of all parameterized problems $L \subseteq \Sigma^* \times \mathbb{N}$, for some finite alphabet Σ, for which there exist an alphabet Π, a computable function $f : \mathbb{N} \to \Pi^*$, and a problem $P \subseteq \Sigma^* \times \Pi^*$ such that $P \in C$ and for all instances $(x, k) \in \Sigma^* \times \mathbb{N}$ of L we have that $(x, k) \in L$ if and only if $(x, f(k)) \in P$. Intuitively, the class para-C consists of all problems that are in C after a precomputation that only involves the parameter [17].

3 Fpt-Reductions to SAT

Problems in NP and co-NP can be encoded into SAT in such a way that the time required to produce the encoding and consequently also the size of the resulting SAT instance are polynomial in the input (the encoding is a polynomial-time many-one reduction). Typically, the SAT encodings of problems proposed for practical use are of this kind (cf. [38]). For problems that are "beyond NP," say for problems on the second level of the PH, such polynomial SAT encodings do not exist, unless the PH collapses. However, for such problems, there still could exist SAT encodings which can be produced in fpt-time in terms of some parameter associated with the problem. In fact, such fpt-time SAT encodings have been obtained for various problems on the second level of the PH [16, 25, 37]. The classes para-NP and para-co-NP contain exactly those parameterized problems that admit such a many-one fpt-reduction to SAT and UNSAT, respectively. Thus, with fpt-time encodings, one can go significantly beyond what is possible by conventional polynomial-time SAT encodings.

Consider the following example. The problem of deciding satisfiability of a QBF does not allow a polynomial-time SAT encoding. However, if the number of universal variables is small, one can use known methods in QBF solving to get an fpt-time encoding into SAT.

QBF-SAT(# $\forall$-vars)
Instance: A QBF φ.
Parameter: The number of universally quantified variables of φ.
Question: Is φ true?

The idea behind this encoding is to repeatedly use universal quantifier expansion [2, 5]. Eliminating k many universally quantified variables in this manner leads to an existentially quantified formula that is at most a factor of 2^k larger than the original formula.

Fpt-time encodings to SAT also have their limits. Clearly, para-Σ_2^p-hard and para-Π_2^p-hard parameterized problems do not admit fpt-time encodings to SAT, even when the parameter is a constant, unless the PH collapses. There are problems that apparently do not admit fpt-time encodings to SAT, but are neither para-Σ_2^p-hard nor para-Π_2^p-hard. In recent work [25] we have introduced several complexity classes for such intermediate problems, including the following. The parameterized complexity class $\exists^k\forall^*$ consists of all parameterized problems that can be many-one fpt-reduced to the following variant of quantified Boolean satisfiability that is based on truth assignments of restricted weight.

$\exists^k\forall^*$-WSat
Instance: A quantified Boolean formula $\varphi = \exists X.\forall Y.\psi$, and an integer k.
Parameter: k.
Question: Does there exist a truth assignment α to X with weight k such that for all truth assignments β to Y the assignment $\alpha \cup \beta$ satisfies ψ?

For each problem in $\exists^k\forall^*$ there exists an xp-reduction to Unsat. However, there is evidence that problems that are hard for $\exists^k\forall^*$ do not allow a many-one fpt-reduction to Sat [25]. Many natural parameterized problems from various domains are complete for the class $\exists^k\forall^*$, and for none of them an fpt-reduction to Sat or Unsat has been found. If there exists an fpt-reduction to Sat for any $\exists^k\forall^*$-complete problem then this is the case for all $\exists^k\forall^*$-complete problems. The dual complexity class of $\exists^k\forall^*$ is denoted by $\forall^k\exists^*$, and has similar (yet dual) properties. Note that the notion of reducibility underlying hardness for all parameterized complexity classes mentioned above is that of many-one fpt-reductions. For a more detailed discussion on the complexity classes $\exists^k\forall^*$ and $\forall^k\exists^*$, we refer to previous work [25].

One can also enhance the power of polynomial-time SAT encodings by considering polynomial-time algorithms that can query a SAT solver multiple times. Such an approach has been shown to be quite effective in practice (see, e.g., [3, 15, 33]) and extends the scope of SAT solvers to problems in the class Δ_2^p, but not to problems that are Σ_2^p-hard or Π_2^p-hard. Also here, switching from polynomial-time to fpt-time provides a significant increase in power. The class para-Δ_2^p contains all parameterized problems that can be solved by an fpt-algorithm that can query a SAT solver multiple times (i.e., by an fpt-time Turing reduction to SAT).

$$\begin{tikzcd}
 & \text{para-}\Sigma^{\text{p}}_2 & & \text{para-}\Pi^{\text{p}}_2 & \\
 & & \text{para-}\Delta^{\text{p}}_2 \arrow[ul, bend right] \arrow[ur, bend left] & & \\
\exists^k\forall^* \arrow[uur, bend left] & & \text{FPT}^{\text{NP}[f(k)]} \arrow[u] & & \forall^k\exists^* \arrow[uul, bend right] \\
 & & \text{para-DP} \arrow[u] & & \\
 & \text{para-NP} \arrow[uuuu, bend left] \arrow[ur, bend left] \arrow[uurrr, bend right] & & \text{para-co-NP} \arrow[uuuu, bend right] \arrow[ul, bend right] \arrow[uulll, bend left] & \\
 & \text{W}[1] \arrow[u, dashed] & & \text{co-W}[1] \arrow[u, dashed] & \\
 & & \text{FPT} = \text{para-P} \arrow[ul] \arrow[ur] & &
\end{tikzcd}$$

Figure 1: Parameterized complexity classes up to the second level of the polynomial hierarchy. Arrows indicate inclusion relations. (We omit the full definition of some of the parameterized complexity classes depicted in the figure. For a detailed definition of these, we refer to other sources [13, 18, 36].)

An overview of all relevant parameterized complexity classes can be found in Figure 1. Locating problems in the complexity landscape as laid out in this figure can provide a guideline for practitioners, to indicate what algorithmic approaches are possible and where complexity theoretic obstacles lie.

There are two fundamental aspects that are relevant for an algorithm that makes queries to a SAT solver: (i) the running time of the algorithm (which does not take into account the time needed by the SAT solver to answer the queries) and (ii) the number of SAT calls. Results from classical *bounded query complexity* [26, 45] suggests that if the running time is polynomial, then increasing the numbers of SAT calls increases the computing power. Several such separation results are known [12, 31]. From a practical point of view, the number of SAT calls may seem to be relatively insignificant, assuming that the queries are easy for the solver, and the solver can reuse information from previous calls [4, 27, 46]. For a theoretical worst-case model, however, one must assume that all queries involve hard SAT instances, and that no information from previous calls can be reused. Therefore, in a theoretical analysis, it makes sense to study the number of SAT calls made by fpt-time algorithms. In the parameterized setting, it is natural to bound the number of SAT calls by a function of the parameter. This yields the class $\text{FPT}^{\text{NP}[f(k)]}$, which lies between para-DP (two calls) and para-Δ^{p}_2 (unrestricted number of calls).

4 Minimization Problems for DNF Formulas

We consider several problems related to minimizing implicants of DNF formulas and minimizing DNF formulas. We consider several parameterizations, and we show that some of these allow an fpt-reduction to SAT, whereas others apparently do not.

The following decision problem, that is concerned with reducing a given implicant of a DNF formula in size, is Σ_2^p-complete [44].

SHORTEST-IMPLICANT-CORE
Instance: A DNF formula φ, an implicant C of φ of size n, and an integer m.
Question: Does there exist an implicant $C' \subseteq C$ of φ of size m?

We consider two parameterizations of this problem: (1) SHORTEST-IMPLICANT-CORE(core size), where the parameter $k = m$ is the size of the minimized implicant, and (2) SHORTEST-IMPLICANT-CORE(reduction size), where the parameter $k = n - m$ is the difference in size between the original implicant and the minimized implicant. We show that neither of these restrictions is enough to admit an fpt-reduction to SAT. All results can straightforwardly be extended to the variant of the problem where implicants of size at most m are accepted.

Next, consider the following decision problem, that is concerned with deciding whether a given DNF formula φ is logically equivalent to a DNF formula φ' of size m, and that is Σ_2^p-complete [44].

DNF-MINIMIZATION
Instance: A DNF formula φ of size n, and an integer m.
Question: Does there exist a term-wise subformula φ' of φ of size m such that $\varphi \equiv \varphi'$?

We consider the following two parameterizations of this problem: (1) DNF-MINIMIZATION(reduction size), where the parameter $k = n - m$ is the difference in size between the original formula φ and the minimized formula φ', and (2) DNF-MINIMIZATION(core size), where the parameter $k = m$ is the size of the minimized formula φ'. We show that the former parameterization is not enough to allow an fpt-reduction to SAT, but that for the latter parameterization, the problem can be solved with an fpt-algorithm that uses at most $\lceil \log_2 k \rceil + 1$ many SAT calls. Moreover, this algorithm works even for the case where equivalent DNF formulas that are not term-wise subformulas of φ are also accepted.

We will now set out to prove the complexity results mentioned in the discussion above. In order to prove $\exists^k\forall^*$-hardness of SHORTEST-IMPLICANT-CORE(core size), we need the following technical lemma (we omit its straightforward proof).

Lemma 1. *Let (φ, k) be an instance of $\exists^k\forall^*$-WSAT. In polynomial time, we can construct an equivalent instance (φ', k) of $\exists^k\forall^*$-WSAT with $\varphi' = \exists X.\forall Y.\psi$, such that for every assignment $\alpha : X \to \{0, 1\}$ that has weight $m \neq k$, it holds that $\forall Y.\psi[\alpha]$ is true.*

Proposition 2. SHORTEST-IMPLICANT-CORE(core size) *is $\exists^k\forall^*$-complete.*

Proof (sketch). To show hardness, we give a many-one fpt-reduction from $\exists^k\forall^*$-WSAT(DNF) to SHORTEST-IMPLICANT-CORE(core size). Intuitively, the choice for some $C' \subseteq C$ with $|C'| = k$ corresponds directly to the choice of some assignment $\alpha : X \to \{0,1\}$ of weight k. Both involve a choice between $\binom{n}{k}$ many candidates, and in both cases verifying whether the chosen candidate witnesses that the instance is a yes-instance involves solving a co-NP-complete problem. Any implicant C' forces those variables x that are included in C' to be set to true (and the other variables are not forced to take any truth value). However, by Lemma 1, any assignment that sets more than k variables x to true will trivially satisfy ψ. Therefore, the only relevant assignment is the assignment that sets only those x to true that are forced to be true by some C' of length k, and hence the choice for such a C' corresponds exactly to the choice for some assignment α of weight k. To verify whether some C' of length k is an implicant of the formula φ is equivalent to checking whether the formula $\bigwedge_{c \in C'} c \wedge \varphi$ is valid, which in turn is equivalent to checking whether a formula $\forall Y.\psi[\alpha]$ is true, for some assignment α.

Let (φ, k) be an instance of $\exists^k\forall^*$-WSAT(DNF), with $\varphi = \exists X.\forall Y.\psi$. By Lemma 1, we may assume without loss of generality that for any assignment $\alpha : X \to \{0,1\}$ of weight $m \neq k$, $\forall Y.\psi[\alpha]$ is true. We may also assume without loss of generality that $|X| > k$; if this were not the case, (φ, k) would trivially be a no-instance. We construct an instance (φ', C, k) of SHORTEST-IMPLICANT-CORE(core size) by letting $\text{Var}(\varphi') = X \cup Y$, $C = \bigwedge_{x \in X} x$, and $\varphi' = \psi$. Clearly, φ' is a Boolean formula in DNF. Also, consider the assignment $\alpha : X \to \{0,1\}$ where $\alpha(x) = 1$ for all $x \in X$. We know that $\forall Y.\psi[\alpha]$ is true, since α has weight more than k. Therefore C is an implicant of φ'. We omit a detailed proof of correctness for this reduction.

To show membership in $\exists^k\forall^*$, we give a many-one fpt-reduction from SHORTEST-IMPLICANT-CORE(core size) to $\exists^k\forall^*$-WSAT. This reduction uses exactly the same similarity between the two problems, i.e., the fact that assignments of weight k correspond exactly to implicants of length k, and that verifying whether this choice witnesses that the instance is a yes-instance in both cases involves checking validity of a propositional formula. We describe the reduction, and omit a detailed proof of correctness. Let (φ, C, k) be an instance of SHORTEST-IMPLICANT-CORE(core size), where $C = \{c_1, \dots, c_n\}$. We construct an instance (φ', k) of $\exists^k\forall^*$-WSAT, where $\varphi' = \exists X.\forall Y.\psi$, by defining $X = \{x_1, \dots, x_n\}$, $Y = \text{Var}(\varphi)$, $\psi = \psi^{X,Y}_{\text{corr}} \to \varphi$, and $\psi^{X,Y}_{\text{corr}} = \bigwedge_{1 \leq i \leq n}(x_i \to c_i)$. $\square$

Proposition 3. SHORTEST-IMPLICANT-CORE(reduction size) *is* $\exists^k\forall^*$*-complete.*

Proof (sketch). As an auxiliary problem, we consider the parameterized problem $\exists^{n-k}\forall^*$-WSAT, which is a variant of $\exists^k\forall^*$-WSAT. Given an input consisting of a QDNF $\varphi = \exists X.\forall Y.\psi$ with $|X| = n$ and an integer k, the problem is to decide whether there exists an assignment α to X with weight $n - k$ such that $\forall Y.\psi[\alpha]$ is true. The parameter for this problem is k. We claim that this problem has the following properties. We omit a proof of these claims.

Claim 1. $\exists^{n-k}\forall^*$-WSAT is $\exists^k\forall^*$-complete.

Claim 2. Let (φ, k) be an instance of $\exists^{n-k}\forall^*$-WSAT. In polynomial time, we can construct an equivalent instance (φ', k) of $\exists^{n-k}\forall^*$-WSAT with $\varphi' = \exists X.\forall Y.\psi$, such that for any assignment

$\alpha : X \rightarrow \{0, 1\}$ that has weight $m \neq (|X| - k)$, it holds that $\forall Y.\psi[\alpha]$ is true.

Using these claims, both membership and hardness for $\exists^k\forall^*$ follow straightforwardly using arguments similar to the $\exists^k\forall^*$-completeness proof of Shortest-Implicant-Core(core size). The fpt-reductions in the proof of Proposition 2 show that Shortest-Implicant-Core(reduction size) fpt-reduces to and from $\exists^{n-k}\forall^*$-WSat. □

We can now turn to proving complexity results for the problems of minimizing DNF formulas.

Proposition 4. DNF-Minimization(reduction size) *is $\exists^k\forall^*$-complete.*

Proof (sketch). To show $\exists^k\forall^*$-hardness, we use the reduction from the literature that is used to show Σ_2^p-hardness of the unparameterized version of DNF-Minimization(reduction size). The polynomial-time reduction from the unparameterized version of Shortest-Implicant-Core(reduction size) to the unparameterized version of DNF-Minimization(reduction size) given by Umans [44, Theorem 2.2] is an fpt-reduction from Shortest-Implicant-Core(reduction size) to DNF-Minimization(reduction size).

To show membership in $\exists^k\forall^*$, one can give an fpt-reduction to $\exists^k\forall^*$-WSat. We describe the main idea behind this reduction, and we omit a detailed proof. Given an instance (φ, k) of DNF-Minimization(reduction size) we construct an instance (φ', k) of $\exists^k\forall^*$-WSat where the assignment to the existentially quantified variables of φ' represents the k many literal occurrences that are to be removed, and where universally quantified part of φ' is used to verify the equivalence of φ and the formula obtained from φ by removing the k literals chosen by the assignment to the existential variables. □

The following result, which we give without proof, gives some first upper and lower bounds on the complexity of DNF-Minimization(core size).

Proposition 5. DNF-Minimization(core size) *is* para-co-NP*-hard and is in $\exists^k\forall^*$.*

Next, we turn our attention to an fpt-algorithm that solves DNF-Minimization(core size) by using $f(k)$ many SAT calls, for some computable function f. In order to so, we will define the notion of relevant variables, and establish several lemmas that help us to describe and analyze the algorithm (the first of which we state without proof).

Let φ be a DNF formula and let $x \in \text{Var}(\varphi)$ be a variable occurring in φ. We call x *relevant in φ* if there exists some assignment $\alpha : \text{Var}(\varphi)\backslash\{x\} \rightarrow \{0, 1\}$ such that $\varphi[\alpha \cup \{x \mapsto 0\}] \neq \varphi[\alpha \cup \{x \mapsto 1\}]$.

Lemma 6. *Let φ be a DNF formula and let φ' be a DNF formula of minimal size that it is equivalent to φ. Then for every variable $x \in \text{Var}(\varphi)$ it holds that $x \in \text{Var}(\varphi')$ if and only if x is relevant in φ.*

Lemma 7. *Given a DNF formula φ and a positive integer m (given in unary), deciding whether there are at least m variables that are relevant in φ is in NP.*

Proof. We describe a guess-and-check algorithm that decides the problem. The algorithm first guesses m distinct variables occurring in φ, and for each guessed variable x the algorithm guesses an assignment α_x to the remaining variables $\text{Var}(\varphi)\backslash\{x\}$. Then, the algorithm verifies whether the guessed variables are really relevant by checking that, under α_x, assigning different values to x changes the outcome of the Boolean function represented by φ, i.e., $\varphi[\alpha_x \cup \{x \mapsto 0\}] \neq \varphi[\alpha_x \cup \{x \mapsto 1\}]$. It is straightforward to construct a SAT instance ψ that implements this guess-and-check procedure. Moreover, from any assignment that satisfies ψ it is easy to extract the relevant variables. □

Lemma 8. *Let $x_1, \dots, x_k$ be propositional variables. There are $2^{O(k \log k)}$ many different DNF formulas ψ over the variables $x_1, \dots, x_k$ that are of size k.*

Proof. Each suitable DNF formula $\psi = t_1 \vee \dots \vee t_\ell$ can be formed by writing down a sequence $\sigma = (l_1, \dots, l_k)$ of literals l_i over $x_1, \dots, x_k$, and splitting this sequence into terms, i.e., choosing integers $1 = d_1 < \dots < d_{\ell+1} = k + 1$ such that $t_i = \{l_{d_i}, \dots, l_{d_{i+1}-1}\}$ for each $1 \leq i \leq \ell$. To see that there are $2^{O(k \log k)}$ many formulas ψ, it suffices to see that there are $O(k^k)$ many sequences σ, and $O(2^k)$ many choices for the integers d_i. □

Theorem 9. DNF-MINIMIZATION(core size) *can be solved by an fpt-algorithm that uses $(\lceil \log_2 k \rceil + 1)$ many SAT calls, where the SAT solver returns a model for satisfiable formulas. Moreover, the first $\lceil \log_2 k \rceil$ many calls to the solver use SAT instances of size $O(k^2 n^2)$, whereas the last call uses a SAT instance of size $2^{O(k \log k)} \cdot n$, where n is the input size.*

Proof. The algorithm given in pseudo-code in Algorithm 1 solves the problem DNF-MINIMIZATION(core size) in the required time bounds. To obtain the required running time, we assume that each call to a SAT solver takes only a single time step. By Lemma 6, we know that any minimal equivalent formula of φ must contain all and only the variables that are relevant in φ. The algorithm firstly determines how many variables are relevant in φ. By Lemma 7, we know that this can be done with a binary search using $\lceil \log_2 k \rceil$ SAT calls. If there are more than k relevant variables, the algorithm rejects. Otherwise, the algorithm will have computed the set rvars of relevant variables. Next, with a single SAT call, it checks whether there exists some DNF formula ψ of size k over the variables in rvars. By Lemma 8, we know that there are $2^{O(k \log k)}$ many different DNF formulas ψ of size k over the variables in rvars. Verifying whether a particular DNF formula ψ is equivalent to the original formula φ can be done by checking whether the formula $\varphi_\psi = (\psi \wedge \neg\varphi) \vee (\neg\psi \wedge \varphi)$ is unsatisfiable. Verifying whether there exists some suitable DNF formula ψ that is equivalent to φ can be done by making variable-disjoint copies of all φ_ψ and checking whether the conjunction of these copies is unsatisfiable. □

Note that the algorithm requires that the SAT solver returns a model if the query is satisfiable. Also, the algorithm can be modified straightforwardly to return a DNF formula ψ of size at most k that is equivalent to an input φ if such a formula ψ exists. It would need to search for this ψ that is equivalent to φ, for which it would need an additional $O(k \log k)$ many SAT calls (with instances of size $2^{O(k \log k)} \cdot \|\varphi\|$).

Algorithm 1: Solving DNF-Minimization(core size) in fpt-time using ($\lceil \log_2 k \rceil + 1$) many SAT calls.

```
input  : an instance $(\varphi, k)$ of DNF-Minimization(core size)
output : YES iff $(\varphi, k) \in$ DNF-Minimization(core size)

rvars $\leftarrow \emptyset$ ;                                   // variables relevant in $\varphi$
$i \leftarrow 0$; $j \leftarrow k + 2$ ;                              // bounds on # of rvars
while $i + 1 < j$ do                         // logarithmic search for the # of rvars
    $\ell \leftarrow \lceil (i + j)/2 \rceil$ ;
    query the SAT solver whether there exist at least $\ell$ variables
         that are relevant in $\varphi$ ;        // for idea behind encoding, see Lemma 7
    if the SAT solver returns a model $M$ then
        rvars $\leftarrow$ the $\ell$ many relevant variables encoded by the model $M$ ;
    else break;
if |rvars| $> k$ then
    return NO ;                          // too many rvars for any DNF of size $\leq k$
else
    foreach DNF formula $\psi$ of size $k$ over var's in rvars do        // $2^{O(k \log k)}$ many
        construct a formula $\varphi_\psi$ that is unsatisfiable iff $\psi \equiv \varphi$;
                                       // the formulas $\varphi_\psi$ must be variable disjoint
    query the SAT solver whether $\bigwedge_\psi \varphi_\psi$ is satisfiable ;
    if the SAT solver returns YES then
        return NO ;                           // no candidate $\psi$ is equivalent to $\varphi$
    else
        return YES ;                        // some candidate $\psi$ is equivalent to $\varphi$
```

From a practical point of view, the algorithm given in Algorithm 1 might not be the best approach to solve the problem. The (single) instance produced for the last SAT call in the algorithm is rather large (exponential in k). However, this instance is equivalent to the conjunction of $2^{O(k \log k)}$ many instances of linear size, and these instances can be solved in parallel. Such a parallel approach involves more (yet easier) SAT calls, but might be more efficient in practice.

An interesting topic for further research is to investigate how many SAT calls are needed for an fpt-algorithm to solve DNF-Minimization(core size) when the SAT solver only returns whether or not the input is satisfiable, and does not return a satisfying assignment in case the input is satisfiable, i.e., whether DNF-Minimization(core size) is contained in $\mathrm{FPT}^{\mathrm{NP}[f(k)]}$. The difference between SAT solvers that return a yes-or-no answer and a satisfying assignment is (theoretically) not relevant when the number of calls is unrestricted, but it seems to make a difference in cases where the number of calls is bounded to logarithmically many in the input size (cf. [22, Theorem 5.4]) or bounded by a function of the parameter.

From a practical point of view, the algorithm given in Algorithm 1 might not be the best approach to solve the problem. The (single) instance produced for the last SAT call in

the algorithm is rather large (exponential in k). However, this instance is equivalent to the conjunction of $2^{O(k \log k)}$ many instances of linear size, and these instances can be solved in parallel. Such a parallel approach involves more (yet easier) SAT calls, but might be more efficient in practice.

5 QBF Satisfiability and Treewidth

The graph parameter *treewidth* measures in a certain sense the tree-likeness of a graph (for a definition of treewidth, see, e.g., [10, 11]). Many hard problems are fixed-parameter tractable when parameterized by the treewidth of a graph associated with the input [11, 23]. By associating the following graph with a QDNF formulas one can apply the parameter treewidth also to QDNF formulas (for QCNF formulas the graph can be defined analogously, taking clauses instead of terms).

The *incidence graph* of a QDNF formula φ is the bipartite graph where one side of the partition consists of the variables and the other side consists of the terms; a variable and a term are adjacent if the variable appears positively or negatively in the term. The *incidence treewidth* of φ, in symbols $\text{incid.tw}(\varphi)$, is the treewidth of the incidence graph of φ. It is well known that checking the truth of a QDNF formula whose number of quantifier alternations is bounded by a constant is fixed-parameter tractable when parameterized by incid.tw (this can be easily shown using Courcelle's Theorem [23]).

Bounding the treewidth of the entire incidence graph is very restrictive. In this section we investigate whether bounding the treewidth of certain subgraphs of the incidence graph is sufficient to reduce the complexity. To this aim we define the *existential incidence treewidth* of a QDNF formula φ, in symbols $\exists\text{-incid.tw}(\varphi)$, as the treewidth of the incidence graph of φ after deletion of all universal variables. The *universal incidence treewidth*, in symbols $\forall\text{-incid.tw}(\varphi)$, is the treewidth of the incidence graph of φ after deletion of all existential variables.

The existential and universal treewidth can be small for formulas whose incidence treewidth is arbitrarily large. Take for instance a QDNF formula φ whose incidence graph is an $n \times n$ square grid, as in Figure 2. In this example, $\exists\text{-incid.tw}(\varphi) = \forall\text{-incid.tw}(\varphi) = 2$ (since after the deletion of the universal or the existential variables the incidence graph becomes a collection of trivial path-like graphs), but $\text{incid.tw}(\varphi) = n$ [10]. Hence, a tractability result in terms of existential or universal incidence treewidth would apply to a significantly larger class of instances than a tractability result in terms of incidence treewidth. In the following we pinpoint the exact complexity of checking the satisfiability of $\exists\forall$-QDNF formulas parameterized by $\exists$-incid.tw and $\forall$-incid.tw. We let $\text{QS}\textsc{at}_2(\forall\text{-incid.tw})$ denote the problem $\exists\forall$-S<span>AT</span> parameterized by $\forall$-incid.tw, and similarly we let $\text{QS}\textsc{at}_2(\exists\text{-incid.tw})$ denote the problem $\exists\forall$-SAT parameterized by $\exists$-incid.tw. We show that parameterizing by $\exists$-incid.tw does not decrease the complexity and leaves the problem on the second level of the PH, but for $\text{QS}\textsc{at}_2(\forall\text{-incid.tw})$ we get an fpt-reduction to SAT.

Proposition 10. $\text{QS}\textsc{at}_2(\exists\text{-incid.tw})$ *is* para-Σ^{p}_2*-complete.*

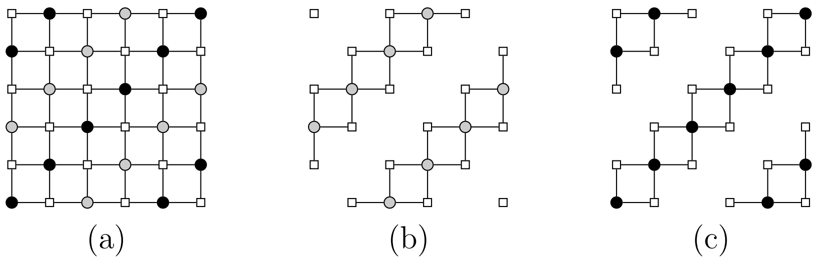

Figure 2: Incidence graph of a QDNF formula (a). Universal variables are drawn with black round shapes, existential variables with grey round shapes, and terms are drawn with square shapes. Both deleting the universal variables (b) and the existential variables (c) significantly decreases the treewidth of the incidence graph.

Proof. Membership in para-Σ_2^{p} is obvious. To show para-Σ_2^{p}-hardness, it suffices to show that the problem is already Σ_2^{p}-hard when the parameter value is restricted to 1 [17]. We show this by means of a reduction from $\exists\forall$-Sat. The idea of this reduction is to introduce for each existentially quantified variable x a corresponding universally quantified variable z_x that is used to represent the truth value assigned to x. Each of the existentially quantified variables then only directly interacts with universally quantified variables.

Take an arbitrary instance of $\exists\forall$-Sat, specified by $\varphi = \exists X.\forall Y.\psi(X,Y)$, where $\psi(X,Y)$ is in DNF. We introduce a new set $Z = \{\, z_x : x \in X \,\}$ of variables. It is straightforward to verify that $\varphi = \exists X.\forall Y.\psi(X,Y)$ is equivalent to the formula $\exists Z.\forall X.\forall Y.\chi$, where $\chi = \bigvee_{x \in X}[(x \wedge \neg z_x) \vee (\neg x \wedge z_x)] \vee \psi(X,Y)$. Clearly, if we now delete all universally quantified variables, the incidence graph of χ consists only of isolated paths of length 2, and therefore the treewidth is 1. This proves that QSat$_2$($\exists$-incid.tw) is para-Σ_2^{p}-hard. □

Theorem 11. QSat$_2$($\forall$-incid.tw) *is* para-NP-*complete.*

Proof. Hardness for para-NP can be proven by showing that the problem is already NP-hard when restricted to instances where the parameter value is 1 [17]. In order to do this, one can reduce an instance of Sat to an instance of $\exists\forall$-Sat whose matrix is in DNF by using the standard Tseitin transformation, resulting in tree-like interactions between the universally quantified variables. Therefore, the resulting formula has universal incidence treewidth 1.

We now show para-NP-membership of QSat$_2$($\forall$-incid.tw). We construct a CNF formula φ' that is satisfiable if and only if φ is true. The idea is to construct a formula that encodes the following guess-and-check algorithm. Firstly, the algorithm guesses an assignment γ to the existential variables. Note that the incidence graph of the formula instantiated with γ has a small treewidth, because instantiating with γ removes all existentially quantified variables. Then, the algorithm employs dynamic programming to exploit the fact that the incidence graph of the remaining formula has small treewidth to decide validity of the remaining formula. This dynamic programming approach is widely used to solve problems for instances where some graph representing the structure of the instance has small treewidth (cf. [9]).

Next, we show how to encode this guess-and-check algorithm into a formula φ' that is satisfiable if and only if the algorithm accepts. In order to do so, we formally define treewidth and tree decompositions of graphs. A tree decomposition of a graph $G = (V, E)$ is a pair $(\mathcal{T}, (B_t)_{t \in T})$ where $\mathcal{T} = (T, F)$ is a rooted tree and $(B_t)_{t \in T}$ is a family of subsets of V such that: (i) for every $v \in V$, the set $B^{-1}(v) = \{\, t \in T : v \in B_t \,\}$ induces a nonempty subtree of $\mathcal{T}$; and (ii) for every edge $\{v, w\} \in E$, there is a $t \in T$ such that $v, w \in B_t$. In order to simplify the proof, we will consider the following normal form of tree decompositions. We call a tree decomposition $(\mathcal{T}, (B_t)_{t \in T})$ *nice* if every node $t \in T$ is of one of the following four types: (*leaf node*) t has no children and $|B_t| = 1$; (*introduce node*) t has one child t' and $B_t = B_{t'} \cup \{v\}$ for some vertex $v \notin B_{t'}$; (*forget node*) t has one child t' and $B_t = B_{t'} \backslash \{v\}$ for some vertex $v \in B_{t'}$; or (*join node*) t has two children t_1, t_2 and $B_t = B_{t_1} = B_{t_2}$. Given any graph G and a tree decomposition of G of width k, a nice tree decomposition of G of width k can be computed in polynomial time [11, 28].

Let $\varphi = \exists X. \forall Y. \psi$ be a quantified Boolean formula where $\psi = \delta_1 \vee \dots \vee \delta_u$, and let $(\mathcal{T}, (B_t)_{t \in T})$ be a tree decomposition of width k of the incidence graph of φ after deletion of the existentially quantified variables. We may assume without loss of generality that $(\mathcal{T}, (B_t)_{t \in T})$ is a nice tree decomposition. We may also assume without loss of generality that for each $t \in T$, B_t contains some $y \in Y$.

We let $\text{Var}(\varphi') = X \cup Z$ where $Z = \{\, z_{t,\alpha,i} : t \in T, \alpha : \text{Var}(t) \rightarrow \{0,1\}, 1 \leq i \leq u \,\}$. Intuitively, the variables $z_{t,\alpha,i}$ represent whether at least one assignment extending α (to the variables occurring in nodes t' below t) violates the term δ_i of ψ. We then construct φ' as follows by using the structure of the tree decomposition. For all $t \in T$, all $\alpha : \text{Var}(t) \rightarrow \{0,1\}$, all $1 \leq i \leq u$, and each literal $l \in \delta_i$ such that $\text{Var}(l) \in X$, we introduce the clause (I): $(\bar{l} \rightarrow z_{t,\alpha,i})$. Then, for all $t \in T$, all $\alpha : \text{Var}(t) \rightarrow \{0,1\}$, and all $1 \leq i \leq u$ such that for some $l \in \delta_i$ it holds that $\text{Var}(l) \in Y$ and $\alpha(l) = 0$, we introduce the clause (II): $(z_{t,\alpha,i})$. Next, let $t \in T$ be any introduction node with child t', and let $\alpha : \text{Var}(t') \rightarrow \{0,1\}$ be an arbitrary assignment. For any assignment $\alpha' : \text{Var}(t) \rightarrow \{0,1\}$ that extends α, and for each $1 \leq i \leq u$, we introduce the clause (III): $(z_{t',\alpha,i} \rightarrow z_{t,\alpha',i})$. Then, let $t \in T$ be any forget node with child t', and let $\alpha : \text{Var}(t) \rightarrow \{0,1\}$ be an arbitrary assignment. For any assignment $\alpha' : \text{Var}(t') \rightarrow \{0,1\}$ that extends α, and for each $1 \leq i \leq u$, we introduce the clause (IV): $(z_{t',\alpha',i} \rightarrow z_{t,\alpha,i})$. Next, let $t \in T$ be any join node with children t_1, t_2, and let $\alpha : \text{Var}(t) \rightarrow \{0,1\}$ be an arbitrary assignment. For each $1 \leq i \leq u$, we introduce the clauses (V): $(z_{t_1,\alpha,i} \rightarrow z_{t,\alpha,i})$ and $(z_{t_2,\alpha,i} \rightarrow z_{t,\alpha,i})$. Finally, for the root node $t_{\text{root}} \in T$ and for each $\alpha : \text{Var}(t_{\text{root}}) \rightarrow \{0,1\}$ we introduce the clause (VI): $\bigvee_{1 \leq i \leq u} \neg z_{t_{\text{root}},\alpha,i}$. It is straightforward to verify that φ' contains $O(2^k |T|)$ many clauses. We claim that this reduction is correct, i.e., that φ is true if and only if φ' is satisfiable. We omit a detailed proof of this. □

Instead of incidence graphs one can also use *primal graphs* to model the structure of QBF formulas (see, e.g., [1, 35]). One can define corresponding parameters primal treewidth, universal primal treewidth, and existential primal treewidth. The proof of Proposition 10 shows that $\exists\forall$-SAT is para-Σ_2^{p}-hard when parameterized by existential primal treewidth. The parameter incidence treewidth is more general than primal treewidth in the sense that small primal treewidth implies small incidence treewidth [29, 40], but the converse does not hold in general. Hence, Theorem 11 also holds for the parameter universal primal treewidth.

6 Conclusion

We studied the fpt-reducibility to SAT for several problems beyond NP under natural parameters. Our positive results show that in some cases it is possible to utilize structure in terms of parameters to break through the barriers between classical complexity classes. Parameters that admit an fpt-reduction to SAT can be less restrictive than parameters that provide fixed-parameter tractability of the problem itself, hence our approach extends the scope of fixed-parameter tractability. Additionally, we show that fpt-time algorithms that can query a SAT solver (i.e., fpt-time Turing reductions to SAT) exist for some problems that cannot be solved in polynomial-time with the help of queries to a SAT solver (unless the PH collapses), hence our approach also extends the scope of algorithms using SAT queries. Our negative results point out the limits of the approach, showing that some problems do, most likely, not admit fpt-reductions to SAT under certain natural parameters.

References

[1] Albert Atserias and Sergi Oliva. Bounded-width QBF is PSPACE-complete. In Natacha Portier and Thomas Wilke, editors, *30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, February 27 - March 2, 2013, Kiel, Germany*, volume 20 of *LIPIcs*, pages 44–54. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[2] Abdelwaheb Ayari and David A. Basin. Qubos: Deciding quantified Boolean logic using propositional satisfiability solvers. In Mark Aagaard and John W. O'Leary, editors, *Proceedings of FMCAD 2002 (Formal Methods in Computer-Aided Design, 4th International Conference, November 6-8, 2002, Portland, OR, USA)*, volume 2517 of *Lecture Notes in Computer Science*, pages 187–201. Springer Verlag, 2002.

[3] Anton Belov, Inês Lynce, and João Marques-Silva. Towards efficient MUS extraction. *AI Commun.*, 25(2):97–116, 2012.

[4] Marco Benedetti and Sara Bernardini. Incremental compilation-to-SAT procedures. In *Proceedings of SAT 2004 (Seventh International Conference on Theory and Applications of Satisfiability Testing, 10–13 May, 2004, Vancouver, BC, Canada)*, pages 46–58, 2004.

[5] Armin Biere. Resolve and expand. In *Proceedings of SAT 2004 (Seventh International Conference on Theory and Applications of Satisfiability Testing, 10–13 May, 2004, Vancouver, BC, Canada)*, pages 59–70, 2004.

[6] Armin Biere. Bounded model checking. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 457–481. IOS Press, 2009.

[7] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In Rance Cleaveland, editor, *Proceedings of the 5th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 1999), Amsterdam, The Netherlands, March 22–28, 1999*, volume 1579 of *Lecture Notes in Computer Science*, pages 193–207. Springer Verlag, 1999.

[8] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2009.

[9] Hans L. Bodlaender. Dynamic programming on graphs with bounded treewidth. In *Automata, languages and programming (Tampere, 1988)*, volume 317 of *Lecture Notes in Computer Science*, pages 105–118. Springer Verlag, 1988.

[10] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1-2):1–45, 1998.

[11] Hans L. Bodlaender. Fixed-parameter tractability of treewidth and pathwidth. In Hans L. Bodlaender, Rod Downey, Fedor V. Fomin, and Dániel Marx, editors, *The Multivariate Algorithmic Revolution and Beyond - Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, volume 7370 of *Lecture Notes in Computer Science*, pages 196–227. Springer Verlag, 2012.

[12] Richard Chang and Jim Kadin. The Boolean Hierarchy and the Polynomial Hierarchy: A closer connection. *SIAM J. Comput.*, 25(2):340–354, 1996.

[13] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Monographs in Computer Science. Springer Verlag, New York, 1999.

[14] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity.* Texts in Computer Science. Springer Verlag, 2013.

[15] Wolfgang Dvořák, Matti Järvisalo, Johannes Peter Wallner, and Stefan Woltran. Complexity-sensitive decision procedures for abstract argumentation. *Artificial Intelligence*, 206(0):53–78, 2014.

[16] Johannes Klaus Fichte and Stefan Szeider. Backdoors to normality for disjunctive logic programs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI 2013)*, pages 320–327. AAAI Press, 2013.

[17] Jörg Flum and Martin Grohe. Describing parameterized complexity classes. *Information and Computation*, 187(2):291–319, 2003.

[18] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series.* Springer Verlag, Berlin, 2006.

[19] Michael R. Garey and David R. Johnson. *Computers and Intractability.* W. H. Freeman and Company, New York, San Francisco, 1979.

[20] Judy Goldsmith, Matthias Hagen, and Martin Mundhenk. Complexity of DNF minimization and isomorphism testing for monotone formulas. *Information and Computation*, 206(6):760–775, June 2008.

[21] Carla P. Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. Satisfiability solvers. In *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 89–134. Elsevier, 2008.

[22] Georg Gottlob and Christian G. Fermüller. Removing redundancy from a clause. *Artificial Intelligence*, 61(2):263–289, 1993.

[23] Georg Gottlob, Reinhard Pichler, and Fang Wei. Bounded treewidth as a key to tractability of knowledge representation and reasoning. *Artificial Intelligence*, 174(1):105–132, 2010.

[24] Éric Grégoire, Bertrand Mazure, and Cédric Piette. On approaches to explaining infeasibility of sets of Boolean clauses. In *20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2008), 3-5 November 2008, Daytion, Ohio, USA*, pages 74–83. IEEE Computer Society, 2008.

[25] Ronald de Haan and Stefan Szeider. The parameterized complexity of reasoning problems beyond NP. In Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter, editors, *Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2014), Vienna, Austria, July 20-24, 2014.* AAAI Press, 2014.

[26] Juris Hartmanis. New developments in structural complexity theory. *Theoretical Computer Science*, 71(1):79–93, 1990.

[27] John N. Hooker. Solving the incremental satisfiability problem. *J. Logic Programming*, 15(1&2):177–186, 1993.

[28] T. Kloks. *Treewidth: Computations and Approximations.* Springer Verlag, Berlin, 1994.

[29] Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. *J. of Computer and System Sciences*, 61(2):302–332, 2000. Special issue on the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (Seattle, WA, 1998).

[30] Jan Krajicek. *Bounded arithmetic, propositional logic and complexity theory.* Cambridge University Press, 1995.

[31] Mark W. Krentel. The complexity of optimization problems. *J. of Computer and System Sciences*, 36(3):490–509, 1988.

[32] Sharad Malik and Lintao Zhang. Boolean satisfiability from theoretical hardness to practical success. *Communications of the ACM*, 52(8):76–82, 2009.

[33] Joao Marques-Silva, Mikoláš Janota, and Anton Belov. Minimal sets over monotone predicates in Boolean formulae. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013, Proceedings*, volume 8044 of *LNCS*, pages 592–607. Springer Verlag, 2013.

[34] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms.* Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, Oxford, 2006.

[35] Guoqiang Pan and Moshe Y. Vardi. Fixed-parameter hierarchies inside PSPACE. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, pages 27–36. IEEE Computer Society, 2006.

[36] Christos H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[37] Andreas Pfandler, Stefan Rümmele, and Stefan Szeider. Backdoors to abduction. In Francesca Rossi, editor, *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013.* AAAI Press/IJCAI, 2013.

[38] Steven David Prestwich. CNF encodings. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, pages 75–97. IOS Press, 2009.

[39] Karem A. Sakallah and João Marques-Silva. Anatomy and empirical evaluation of modern SAT solvers. *Bulletin of the European Association for Theoretical Computer Science*, 103:96–121, 2011.

[40] Marko Samer and Stefan Szeider. Constraint satisfaction with bounded treewidth revisited. *J. of Computer and System Sciences*, 76(2):103–114, 2010.

[41] Jörg Siekmann and Graham Wrightson, editors. *Automation of reasoning. Classical Papers on Computer Science 1967–1970*, volume 2. Springer Verlag, 1983.

[42] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3(1):1–22, 1976.

[43] G. S. Tseitin. Complexity of a derivation in the propositional calculus. *Zap. Nauchn. Sem. Leningrad Otd. Mat. Inst. Akad. Nauk SSSR*, 8:23–41, 1968. English translation reprinted in [41].

[44] Christopher Umans. *Approximability and Completeness in the Polynomial Hierarchy.* PhD thesis, University of California, Berkeley, 2000.

[45] Klaus W. Wagner. Bounded query classes. *SIAM J. Comput.*, 19(5):833–846, 1990.

[46] Jesse Whittemore, Joonyoung Kim, and Karem A. Sakallah. SATIRE: A new incremental satisfiability engine. In *Proceedings of the 38th Design Automation Conference, DAC 2001, Las Vegas, NV, USA, June 18-22, 2001*, pages 542–545. ACM, 2001.