The Parameterized Complexity of Reasoning Problems Beyond NP

Ronald de Haan* and **Stefan Szeider***
Institute of Information Systems
Vienna University of Technology
dehaan@kr.tuwien.ac.at stefan@szeider.net

Abstract

Today's propositional satisfiability (SAT) solvers are extremely powerful and can be used as an efficient back-end for solving NP-complete problems. However, many fundamental problems in knowledge representation and reasoning 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. Recent research shows that 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.

In this paper we develop a general theoretical framework that supports the classification of parameterized problems on whether they admit such an fpt-reduction to SAT or not. We instantiate our theory by classifying the complexities of several case study problems, with respect to various natural parameters. These case studies include the consistency problem for disjunctive answer set programming and a robust version of constraint satisfaction.

1 Introduction

Over the last two decades, propositional satisfiability (SAT) has become one of the most successful and widely applied techniques for the solution of NP-complete problems. Today's SAT-solvers are extremely efficient and robust, instances with hundreds of thousands of variables and clauses can be solved routinely. 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 (Gomes et al., 2008; Biere et al., 2009; Sakallah and Marques-Silva, 2011; Malik and Zhang, 2009). However, many important computational problems, most prominently in knowledge representation and reasoning, are located above the first level of the Polynomial Hierarchy (PH) and thus considered "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.

*Supported by the European Research Council (ERC), project 239962 (COMPLEX REASON), and the Austrian Science Fund (FWF), project P26200 (Parameterized Compilation).

Realistic problem instances are not random and often contain some kind of "hidden structure." Recent research succeeded to exploit such hidden structure to break the complexity barriers between levels of the PH, for problems that arise in disjunctive answer set programming (Fichte and Szeider, 2013) and abductive reasoning (Pfandler, Rümmele, and Szeider, 2013). 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* (Downey and Fellows, 1999; Flum and Grohe, 2006; Niedermeier, 2006) 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 (Biere et al., 1999), 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. Bounded Model Checking is widely used for hardware and software verification at industrial scale (Biere, 2009).

New Contributions The aim of this paper is to establish a general theoretical framework that supports the classification of hard problems on whether they admit an fpt-reduction to SAT or not. The main contribution is the development of a new 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 (Garey and Johnson, 1979) and W[1]-hardness which provides evidence against fixed-parameter tractability (Downey and Fellows, 1999).

At the center of our theory are two hierarchies of parameterized complexity classes: the $*$-k hierarchy and the k-$*$ hierarchy. We define the complexity classes in terms of weighted variants of the quantified Boolean satisfiability problem with

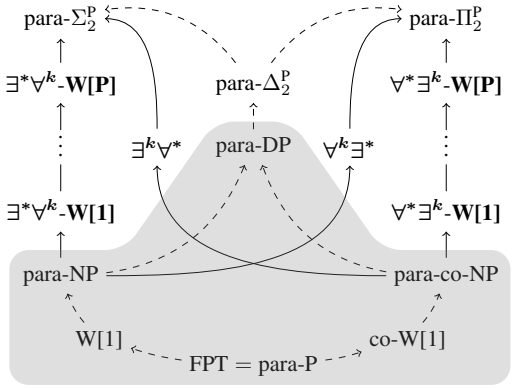

Figure 1: The parameterized complexity classes of the $*$-k and k-$*$ hierarchies (bold) in relation to existing classes. Arrows indicate inclusion relations. Dashed arrows indicate previously known relations. The classes highlighted in gray allow fpt-reductions to SAT; the other classes are unlikely to allow this.

one quantifier alternation, which is canonical for the second level of the PH. For the classes in the k-$*$ hierarchy, the (Hamming) weight of the assignment to the variables in the first quantifier block is bounded by the parameter k, the weight of the second quantifier block is unrestricted ("$*$"). For the classes in the $*$-k hierarchy it is the other way around, the weight in the second block restricted by k and the first block is unrestricted. Both hierarchies span various degrees of hardness between the classes para-NP and para-co-NP at the bottom and para-Σ_2^P at the top (para-C contains all parameterized problems that, after fpt-time preprocessing, ultimately belong to complexity class C (Flum and Grohe, 2003)). Figure 1 illustrates the relationship between the various parameterized complexity classes under consideration.

To illustrate the usefulness of our theory, we consider as a running example the fundamental problem of *answer set programming* which asks whether a disjuncive logic program has a stable model. This problem is Σ_2^P-complete (Eiter and Gottlob, 1995), and exhibits completeness or hardness for various of our complexity classes; see Table 1 for an overview. As a second case study, we will classify the complexity of a robust version of constraint satisfaction (Abramsky, Gottlob, and Kolaitis, 2013). In addition we were able to identify many other natural problems that populate our new complexity classes. We refer to a technical report corresponding to this paper which contains full proofs of all results, contains a compendium of problems, and is available on arXiv (http://arxiv.org/abs/1312.1672).

2 Preliminaries

2.1 Parameterized Complexity Theory

We introduce some core notions from parameterized complexity theory. For an in-depth treatment we refer to other sources (Downey and Fellows, 1999; Flum and Grohe, 2006; Niedermeier, 2006). 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, $k > 3$.

Parameterized complexity also offers a hardness theory, similar to the theory of NP-hardness, that allows researchers to give strong theoretical evidence that some parameterized problems are not fixed-parameter tractable. This theory is based on the *Weft hierarchy* of complexity classes FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \cdots \subseteq$ W[SAT] $\subseteq$ W[P], where all inclusions are believed to be strict. For a hardness theory, a notion of reduction is needed. Let $L \subseteq \Sigma^* \times \mathbb{N}$ and $L' \subseteq (\Sigma')^* \times \mathbb{N}$ be two parameterized problems. An *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. We write $L \leq_{\text{fpt}} L'$ if there is an fpt-reduction from L to L'.

The parameterized complexity classes W[t], $t \geq 1$, W[SAT] and W[P] are based on the satisfiability problems of Boolean circuits and formulas. We consider *Boolean circuits* with a single output gate. We call input nodes *variables*. We distinguish between *small* gates, with fan-in ≤ 2, and *large gates*, with fan-in > 2. The *depth* of a circuit is the length of a longest path from any variable to the output gate. The *weft* of a circuit is the largest number of large gates on any path from a variable to the output gate. We let Nodes(C) denote the set of all nodes of a circuit C. A *Boolean formula* can be considered as a Boolean circuit where all gates have fan-out ≤ 1. We adopt the usual notions of truth assignments and satisfiability of a Boolean circuit. We say that a truth assignment for a Boolean circuit has *weight* k if it sets exactly k of the variables of the circuit to true. We denote the class of Boolean circuits with depth u and weft t by $\Gamma_{t,u}$. We denote the class of all Boolean circuits by Γ, and the class of all Boolean formulas by Φ. For any class $\mathcal{C}$ of Boolean circuits, we define the following parameterized problem.

p-WSAT[$\mathcal{C}$] *Instance:* A Boolean circuit $C \in \mathcal{C}$, and an integer k. *Parameter:* k. *Question:* Does there exist an assignment of weight k that satisfies C?

We denote closure under fpt-reductions by $[\ \cdot\ ]^{\text{fpt}}$. The classes W[t] are defined by letting W[t] $= [\ \{\, p\text{-WSAT}[\Gamma_{t,u}] : u \geq 1 \,\}\ ]^{\text{fpt}}$ for all $t \geq 1$. The classes W[SAT] and W[P] are defined by letting W[SAT] $= [\ p\text{-WSAT}[\Phi]\ ]^{\text{fpt}}$ and W[P] $= [\ p\text{-WSAT}[\Gamma]\ ]^{\text{fpt}}$.

Parameterized complexity theory also offers complexity classes for problems that lie higher in the polynomial hierarchy. Let K be a classical complexity class, e.g., NP. The parameterized complexity class para-K 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 $P \in K$ 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 (Flum and Grohe, 2003). The class para-NP can also be defined via nondeterministic fpt-algorithms.

2.2 The Polynomial Hierarchy

There are many natural decision problems that are not contained in the classical complexity classes P and NP. The *Polynomial Hierarchy* (Meyer and Stockmeyer, 1972; Stockmeyer, 1976; Wrathall, 1976; Papadimitriou, 1994) contains a hierarchy of increasing complexity classes Σ_i^{P}, for all $i \geq 0$. We give a characterization of these classes based on the satisfiability problem of various classes of quantified Boolean formulas. A *quantified Boolean formula* 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 $\bigcup_{i=1}^{m} X_i$. The quantifier-free part of such formulas is called the *matrix* of the formula. Truth of such formulas is defined in the usual way. 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 $\varphi[\gamma]$. For each $i \geq 1$ we define the following decision problem.

QSAT_i
Instance: A quantified Boolean formula $\varphi = \exists X_1 \forall X_2 \exists X_3 \dots Q_i X_i \psi$, where Q_i is a universal quantifier if i is even and an existential quantifier if i is odd.
Question: Is φ true?

Input formulas to the problem QSAT_i are called Σ_i^{P}-formulas. For each nonnegative integer $i \leq 0$, the complexity class Σ_i^{P} can be characterized as the closure of the problem QSAT_i under polynomial-time reductions (Stockmeyer, 1976; Wrathall, 1976). The Σ_i^{P}-hardness of QSAT_i holds already when the matrix of the input formula is restricted to 3CNF for odd i, and restricted to 3DNF for even i. Note that the class Σ_0^{P} coincides with P, and the class Σ_1^{P} coincides with NP. For each $i \geq 1$, the class Π_i^{P} is defined as co-Σ_i^{P}.

2.3 Fpt-reductions to SAT

Every problem in NP $\cup$ co-NP can be solved with one call to a SAT solver, and every problem in DP $= \{ L_1 \cap L_2 : L_1 \in$ NP, $L_2 \in$ co-NP $\}$ can be solved with two calls to a SAT solver. The Boolean Hierarchy (Cai and Hemachandra, 1986) contains all problems that can be solved with a constant number of calls to a SAT solver. On the other hand, (under complexity theoretic assumptions) there are problems in Δ_2^{P} that cannot be solved efficiently with a constant number of calls to a SAT solver. Hence, in particular, by showing that a parameterized problem is in para-NP or para-co-NP (see Section 2.1) we establish that the problem admits an fpt-reduction to SAT (see Figure 1 and Table 1).

In addition, we could consider the class of parameterized problems that can be solved by an fpt-algorithm that makes $f(k)$ many calls to a SAT solver, for some function f. This notion opens another possibility to obtain (parameterized) tractability results for problems beyond NP (cf. De Haan and Szeider, 2014).

2.4 Answer Set Programming

We will use the logic programming setting of answer set programming (ASP) (cf. Marek and Truszczynski, 1999; Brewka, Eiter, and Truszczynski, 2011) as a running example in the remainder of the paper. A *disjunctive logic program* (or simply: a *program*) P is a finite set of rules of the form $r = (a_1 \vee \dots \vee a_k \leftarrow b_1, \dots, b_m, \mathit{not}\ c_1, \dots, \mathit{not}\ c_n)$, for $k, m, n \geq 0$, where all a_i, b_j and c_l are atoms. A rule is called *disjunctive* if $k > 1$, and it is called *normal* if $k \leq 1$ (note that we only call rules with strictly more than one disjunct in the head disjunctive). A rule is called *negation-free* if $n = 0$. A program is called normal if all its rules are normal, and called negation-free if all its rules are negation-free. We let $\mathrm{At}(P)$ denote the set of all atoms occurring in P. By *literals* we mean atoms a or their negations $\mathit{not}\ a$. With $\mathrm{NF}(r)$ we denote the rule $(a_1 \vee \dots \vee a_k \leftarrow b_1, \dots, b_m)$. The *(GL) reduct* of a program P with respect to a set M of atoms, denoted P^M, is the program obtained from P by: (i) removing rules with $\mathit{not}\ a$ in the body, for each $a \in M$, and (ii) removing literals $\mathit{not}\ a$ from all other rules (Gelfond and Lifschitz, 1991). An *answer set* A of a program P is a subset-minimal model of the reduct P^A. The following decision problem is concerned with the question of whether a given program has an answer set.

ASP-CONSISTENCY
Instance: A disjunctive logic program P.
Question: Does P have an answer set?

Many implementations of answer set programming already employ SAT solving techniques, e.g., Cmodels (Giunchiglia, Lierler, and Maratea, 2006), ASSAT (Lin and Zhao, 2004), and Clasp (Gebser et al., 2007). Work has also been done on translations from ASP to SAT, both for classes of programs that allow reasoning within NP or co-NP (Ben-Eliyahu and Dechter, 1994; Fages, 1994; Lin and Zhao, 2004; Janhunen et al., 2006) and for classes of programs for which reasoning is beyond NP and co-NP (Janhunen et al., 2006; Lee and Lifschitz, 2003; Lifschitz and Razborov, 2006). We hope that our work provides new means for a theoretical study of these and related approaches to ASP.

3 Parameterizations for ASP

ASP-CONSISTENCY is Σ_2^{P}-complete in general, and can therefore (under complexity theoretic assumptions) not be reduced to SAT in polynomial time. With the aim of identifying fpt-reductions from ASP-CONSISTENCY to SAT, we consider several parameterizations.

Parameter	Complexity
normality-bd size	para-NP-complete (Fichte and Szeider, 2013)
# contingent atoms	para-co-NP-complete (Prop 1)
# contingent rules	$\exists^k\forall^*$-complete (Thms 7 and 8)
# disjunctive rules	$\exists^*\forall^k$-W[1]-hard (Thm 14)
max atom occurrence	para-Σ_2^{P}-complete (Cor 16)

Table 1: Complexity results for different parameterizations of ASP-CONSISTENCY.

Fichte and Szeider (2013) identified one parameterization of ASP-CONSISTENCY under which the problem is contained in para-NP. This parameterization is based on the notion of backdoors to normality for disjunctive logic programs. A set X of atoms is a *normality-backdoor* for a program P if deleting the atoms $x \in X$ and their negations $not\ x$ from the rules of P results in a normal program. ASP-CONSISTENCY is contained in para-NP, when parameterized by the size of a smallest normality-backdoor of the input program.

Two other parameterizations that we consider are related to atoms that must be part of any answer set of a program P. We identify a subset $\mathrm{Comp}(P)$ of *compulsory atoms*, that any answer set must include. Given a program P, we let $\mathrm{Comp}(P)$ be the smallest set such that: (i) if $(w \leftarrow not\ w)$ is a rule of P, then $w \in \mathrm{Comp}(P)$; and (ii) if $(b \leftarrow a_1, \ldots, a_n)$ is a rule of P, and $a_1, \ldots, a_n \in \mathrm{Comp}(P)$, then $b \in \mathrm{Comp}(P)$. We then let the set $\mathrm{Cont}(P)$ of *contingent atoms* be those atoms that occur in P but are not in $\mathrm{Comp}(P)$. We call a rule *contingent* if it contains contingent atoms in the head. (In fact, we could use any polynomial time computable algorithm A that computes for every program P a set $\mathrm{Comp}_A(P)$ of atoms that must be included in any answer set of P.)

The following are candidates for additional parameters that could result in fpt-reductions to SAT: (i) the number of disjunctive rules in the program (i.e., the number of rules with strictly more than one disjunct in the head); (ii) the number of contingent atoms in the program; and (iii) the number of contingent rules in the program. We will often denote the parameterized problems based on ASP-CONSISTENCY and these parameters (i) ASP-CONSISTENCY(#disj.rules), (ii) ASP-CONSISTENCY(#cont.atoms) and (iii) ASP-CONSISTENCY(#cont.rules), respectively.

The question that we would like to answer is which (if any) of these parameterizations allows an fpt-reduction to SAT. Tools from classical complexity theory seem unfit to distinguish these parameters from each other and from the parameterization by Fichte and Szeider: if the parameter values are given as part of the input, the problem remains Σ_2^{P}-complete in all cases; if we bound the parameter values by a constant, then in all cases the complexity of the problem decreases to the first-level of the PH (a proof of this can be found in the technical report). However, some of the parameterizations allow an fpt-reduction to SAT, whereas others seemingly do not.

Also the existing tools from parameterized complexity theory are unfit to distinguish between these different parameterizations of ASP-CONSISTENCY. Practically all existing parameterized complexity classes that can be used to show that an fpt-reduction is unlikely to exist (such as the classes of the W-hierarchy) are located below para-NP. Therefore, these classes do not allow us to differentiate between problems that are in para-NP and problems that are not.

However, using the parameterized complexity classes developed in this paper we will be able to make the distinction between parameterizations that allow an fpt-reduction to SAT and parameterizations that seem not to allow this. Furthermore, our theory relates the latter ones in such a way that an fpt-reduction to SAT for any of them gives us an fpt-reduction to SAT for all of them. As can be seen in Table 1, ASP-CONSISTENCY(#cont.atoms) can be fpt-reduced to SAT, whereas we have evidence that this is not possible for ASP-CONSISTENCY(#disj.rules) and ASP-CONSISTENCY(#cont.rules).

We will use ASP-CONSISTENCY together with the various parameterizations discussed above as a running example, which allows us to demonstrate the developed theoretical tools. We begin with showing a positive result for ASP-CONSISTENCY(#cont.atoms).

Proposition 1. ASP-CONSISTENCY(#cont.atoms) *is* para-co-NP-*complete.*

Proof. Hardness for para-co-NP follows from the reduction of Eiter and Gottlob (1995, Theorem 3). We show membership in para-co-NP. Let P be a program that contains k many contingent atoms. We sketch an fpt-reduction to SAT for the problem whether P has no answer set. There are 2^k candidate sets that could be an answer set, namely $N \cup \mathrm{Comp}(P)$ for each $N \subseteq \mathrm{Cont}(P)$. For each such set $M_N = N \cup \mathrm{Comp}(P)$ it can be checked in deterministic polynomial time whether M_N is a model of P^{M_N}, and it can be checked by an NP-algorithm whether M_N is not a minimal model of P^{M_N}. Therefore, by the NP-completeness of SAT, for each $N \subseteq \mathrm{Cont}(P)$, there exists a propositional formula φ_N that is satisfiable if and only if M_N is not a minimal model of P^{M_N}. All together, the statement that for no $N \subseteq \mathrm{Cont}(P)$ the set $N \cup \mathrm{Comp}(P)$ is an answer set holds true if and only if the disjunction $\bigvee_{N \subseteq \mathrm{Cont}(P)} \varphi_N$ is satisfiable. □

4 The Hierarchies $*$-k and k-$*$

We are going to define two hierarchies of parameterized complexity classes that will act as intractability classes in our hardness theory. All classes will be based on weighted variants of the satisfiability problem QSAT_2. An instance of the problem QSAT_2 has both an existential quantifier and a universal quantifier block. Therefore, there are several ways of restricting the weight of assignments. Restricting the weight of assignments to the existential quantifier block will result in the k-$*$ hierarchy, and restricting the weight of assignments to the universal quantifier block will result in the $*$-k hierarchy. The two hierarchies are based on the following two parameterized decision problems. Let $\mathcal{C}$ be a class of Boolean circuits. The problem $\exists^k\forall^*$-$\mathrm{WSAT}(\mathcal{C})$ provides the foundation for the k-$*$ hierarchy.

$\exists^k\forall^*$-WSAT($\mathcal{C}$)
Instance: A Boolean circuit $C \in \mathcal{C}$ over two disjoint sets X and Y of variables, 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 C?

Similarly, the problem $\exists^*\forall^k$-WSAT($\mathcal{C}$) provides the foundation for the $*$-k hierarchy.

$\exists^*\forall^k$-WSAT($\mathcal{C}$)
Instance: A Boolean circuit $C \in \mathcal{C}$ over two disjoint sets X and Y of variables, and an integer k.
Parameter: k.
Question: Does there exist a truth assignment α to X such that for all truth assignments β to Y with weight k the assignment $\alpha \cup \beta$ satisfies C?

For convenience, instances to these two problems consisting of a circuit C over sets X and Y of variables and an integer k, we will denote by $(\exists X.\forall Y.C, k)$. We now define the following parameterized complexity classes, that together form the k-$*$ hierarchy. We let $\exists^k\forall^*$-W[t] $= [\, \{\, \exists^k\forall^*\text{-WSAT}(\Gamma_{t,u}) : u \geq 1 \,\} \,]^{\text{fpt}}$, we let $\exists^k\forall^*$-W[SAT] $= [\, \exists^k\forall^*\text{-WSAT}(\Phi) \,]^{\text{fpt}}$, and we let $\exists^k\forall^*$-W[P] $= [\, \exists^k\forall^*\text{-WSAT}(\Gamma) \,]^{\text{fpt}}$.

We define the classes of the $*$-k hierarchy similarly. We let $\exists^*\forall^k$-W[t] $= [\, \{\, \exists^*\forall^k\text{-WSAT}(\Gamma_{t,u}) : u \geq 1 \,\} \,]^{\text{fpt}}$, we let $\exists^*\forall^k$-W[SAT] $= [\, \exists^*\forall^k\text{-WSAT}(\Phi) \,]^{\text{fpt}}$, and we let $\exists^*\forall^k$-W[P] $= [\, \exists^*\forall^k\text{-WSAT}(\Gamma) \,]^{\text{fpt}}$. Note that these definitions are analogous to those of the parameterized complexity classes of the W-hierarchy (Downey and Fellows, 1999).

We can define dual classes for each of the parameterized complexity classes in the k-$*$ and $*$-k hierarchies. These co-classes are based on problems complementary to the problems $\exists^k\forall^*$-WSAT and $\exists^*\forall^k$-WSAT, i.e., these problems have as yes-instances exactly the no-instances of $\exists^k\forall^*$-WSAT and $\exists^*\forall^k$-WSAT, respectively. Equivalently, these complementary problems can be considered as variants of $\exists^k\forall^*$-WSAT and $\exists^*\forall^k$-WSAT where the existential and universal quantifiers are swapped, and are therefore denoted with $\forall^k\exists^*$-WSAT and $\forall^*\exists^k$-WSAT. We use a similar notation for the dual complexity classes, e.g., we denote co-$\exists^*\forall^k$-W[t] by $\forall^*\exists^k$-W[t].

5 The Class $\exists^k\forall^*$

In this section, we consider the k-$*$ hierarchy. It turns out that this hierarchy collapses entirely into a single parameterized complexity class. This class we will denote by $\exists^k\forall^*$. As we will see, the class $\exists^k\forall^*$ turns out to be quite robust. We start this section with showing that that the k-$*$ hierarchy collapses. We discuss how this class is related to existing parameterized complexity classes, and we show how it can be used to show the intractability of a variant of the answer set existence problem whose complexity the existing theory cannot classify properly.

5.1 Collapse of the k-$*$ hierarchy

Theorem 2 (Collapse of the k-$*$ hierarchy). $\exists^k\forall^*$-W[1] $= \exists^k\forall^*$-W[2] $= \ldots = \exists^k\forall^*$-W[SAT] $= \exists^k\forall^*$-W[P].

Proof. Since by definition $\exists^k\forall^*$-W[1] $\subseteq \exists^k\forall^*$-W[2] $\subseteq \ldots \subseteq \exists^k\forall^*$-W[P], it suffices to show that $\exists^k\forall^*$-W[P] $\subseteq \exists^k\forall^*$-W[1]. We show this by giving an fpt-reduction from $\exists^k\forall^*$-WSAT(Γ) to $\exists^k\forall^*$-WSAT(3DNF). Since 3DNF $\subseteq \Gamma_{1,3}$, this suffices. We remark that this reduction is based on the standard Tseitin transformation that transforms arbitrary Boolean formulas into 3CNF by means of additional variables.

Let (φ, k) be an instance of $\exists^k\forall^*$-WSAT(Γ) with $\varphi = \exists X.\forall Y.C$. Assume without loss of generality that C contains only binary conjunctions and negations. Let o denote the output gate of C. We construct an instance (φ', k) of $\exists^k\forall^*$-WSAT(3DNF) as follows. The formula φ' will be over the set of variables $X \cup Y \cup Z$, where $Z = \{\, z_r : r \in \text{Nodes}(C) \,\}$. For each $r \in \text{Nodes}(C)$, we define a subformula χ_r. We distinguish three cases. If $r = r_1 \wedge r_2$, then we let $\chi_r = (z_r \wedge \neg z_{r_1}) \vee (z_r \wedge \neg z_{r_2}) \vee (z_{r_1} \wedge z_{r_2} \wedge \neg z_r)$. If $r = \neg r_1$, then we let $\chi_r = (z_r \wedge z_{r_1}) \vee (\neg z_r \wedge \neg z_{r_1})$. If $r = w$, for some $w \in X \cup Y$, then we let $\chi_r = (z_r \wedge \neg w) \vee (\neg z_r \wedge w)$. Now we define $\varphi' = \exists X.\forall Y \cup Z.\psi$, where $\psi = \bigvee_{r \in \text{Nodes}(C)} \chi_r \vee z_o$. It is straightforward to verify that this reduction is correct. □

As mentioned above, in order to simplify notation, we will use $\exists^k\forall^*$ to denote the class $\exists^k\forall^*$-W[1] $= \ldots = \exists^k\forall^*$-W[P]. Also, we will denote $\exists^k\forall^*$-WSAT(Γ) by $\exists^k\forall^*$-WSAT. We make some observations about the relation of $\exists^k\forall^*$ to existing parameterized complexity classes. It is straightforward to see that $\exists^k\forall^* \subseteq$ para-Σ_2^{P}. In polynomial time, any formula $\exists X.\forall Y.\psi$ can be transformed into a Σ_2^{P}-formula that is true if and only if for some assignment α of weight k to the variables X the formula $\forall Y.\psi[\alpha]$ is true. Trivially, para-co-NP $\subseteq \exists^k\forall^*$. To summarize, we obtain the following inclusions: para-co-NP $\subseteq \exists^k\forall^* \subseteq \Sigma_2^{\text{P}}$, and para-NP $\subseteq \forall^k\exists^* \subseteq \Pi_2^{\text{P}}$. This immediately leads to the following result.

Proposition 3. *If* $\exists^k\forall^* \subseteq$ para-NP*, then* NP $=$ co-NP.

A natural question to ask is whether para-NP $\subseteq \exists^k\forall^*$. The following result indicates that this is unlikely.

Proposition 4. *If* para-NP $\subseteq \exists^k\forall^*$*, then* NP $=$ co-NP.

Proof (sketch). Let SAT be the language of satisfiable propositional formulas, and UNSAT the language of unsatisfiable propositional formulas. The parameterized problem $P = \{\, (\varphi, 1) : \varphi \in \text{SAT} \,\}$ is in para-NP. Since the parameter value is constant for all instances of P, an fpt-reduction from P to $\exists^k\forall^*$-WSAT can be transformed into an polynomial time reduction from SAT to UNSAT. □

This implies that $\exists^k\forall^*$ is very likely to be a strict subset of para-Σ_2^{P}.

Corollary 5. *If* $\exists^k\forall^* =$ para-Σ_2^{P}*, then* NP $=$ co-NP.

The following result shows another way in which the class $\exists^k\forall^*$ relates to the existing complexity class co-NP. Let P be a parameterized decision problem, and let $c \geq 1$ be an integer. We say that the *c-th slice of P*, denoted P_c, is the (unparameterized) decision problem $\{\, x : (x, c) \in P \,\}$.

Proposition 6. *Let P be a parameterized problem complete for $\exists^k\forall^*$, and let $c \geq 1$ be an integer. Then P_c is in* co-NP*. Moreover, there exists some integer $d \geq 1$ such that $P_1 \cup \cdots \cup P_d$ is* co-NP*-complete.*

A proof of this statement can be found in the technical report.

5.2 Answer set programming and completeness for the k-$*$ hierarchy

Now that we defined this new intractability class $\exists^k\forall^*$ and that we have some basic results about it in place, we are able to prove the intractability of a variant of our running example problem. In fact, we show that one variant of our running example is complete for the class $\exists^k\forall^*$.

Theorem 7. ASP-CONSISTENCY(#cont.rules) *is $\exists^k\forall^*$-hard.*

Proof. We give an fpt-reduction from $\exists^k\forall^*$-WSAT(3DNF). This reduction is a parameterized version of a reduction of Eiter and Gottlob (1995, Theorem 3). Let (φ, k) be an instance of $\exists^k\forall^*$-WSAT(3DNF), where $\varphi = \exists X.\forall Y.\psi$, $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$, $\psi = \delta_1 \vee \cdots \vee \delta_u$, and $\delta_\ell = l^\ell_1 \wedge l^\ell_2 \wedge l^\ell_3$ for each $1 \leq \ell \leq u$. We construct a disjunctive program P. We consider the sets X and Y of variables as atoms. In addition, we introduce fresh atoms $v_1, \ldots, v_n$, $z_1, \ldots, z_m$, w, and x^j_i for all $1 \leq j \leq k$, $1 \leq i \leq n$. We let P consist of the following rules:

$$
\begin{array}{lll}
x^j_1 \vee \cdots \vee x^j_n \leftarrow & \text{for } 1 \leq j \leq k; & (1)\\
\leftarrow x^j_i, x^{j'}_i & \text{for } 1 \leq i \leq n, 1 \leq j < j' \leq k; & (2)\\
y_i \vee z_i \leftarrow, \quad w \leftarrow y_i, z_i & \text{for } 1 \leq i \leq m; & (3)\\
y_i \leftarrow w, \quad z_i \leftarrow w & \text{for } 1 \leq i \leq m; & (4)\\
x_i \leftarrow w, \quad v_i \leftarrow w & \text{for } 1 \leq i \leq n; & (5)\\
x_i \leftarrow x^j_i & \text{for } 1 \leq i \leq n, 1 \leq j \leq k; & (6)\\
v_i \leftarrow not\ x^1_i, \ldots, not\ x^k_i & \text{for } 1 \leq i \leq n; & (7)\\
w \leftarrow \sigma(l^\ell_1), \sigma(l^\ell_2), \sigma(l^\ell_3) & \text{for } 1 \leq \ell \leq u; & (8)\\
w \leftarrow not\ w. & & (9)
\end{array}
$$

Here we let $\sigma(x_i) = x_i$ and $\sigma(\neg x_i) = v_i$ for each $1 \leq i \leq n$; and we let $\sigma(y_j) = y_j$ and $\sigma(\neg y_j) = z_j$ for each $1 \leq j \leq m$. Intuitively, v_i corresponds to $\neg x_i$, and z_j corresponds to $\neg y_j$. The main difference with the reduction of Eiter and Gottlob is that we use the rules in (1), (2), (6) and (7) to let the variables x_i and v_i represent an assignment of weight k to the variables in X. The rules in (5) ensure that the atoms v_i and x_i are compulsory. It is straightforward to verify that $\text{Comp}(P) = \{w\} \cup \{\, x_i, v_i : 1 \leq i \leq n \,\} \cup \{\, y_i, z_i : 1 \leq i \leq m \,\}$. Notice that P has exactly k contingent rules, namely the rules in (1). A full proof that $(\varphi, k) \in \exists^k\forall^*$-WSAT if and only if P has an answer set can be found in the technical report. □

Theorem 8. ASP-CONSISTENCY(#cont.rules) *is in $\exists^k\forall^*$.*

Proof. We show membership in $\exists^k\forall^*$ by reducing ASP-CONSISTENCY(#cont.rules) to $\exists^k\forall^*$-WSAT. Let P be a program, where $r_1, \ldots, r_k$ are the contingent rules of P and where $\text{At}(P) = \{d_1, \ldots, d_n\}$. We construct a quantified Boolean formula $\varphi = \exists X.\forall Y \cup Z \cup W.\psi$ such that $(\varphi, k) \in \exists^k\forall^*$-WSAT if and only if P has an answer set.

In order to do so, we firstly construct a Boolean formula $\psi_P(z_1, \ldots, z_n, z'_1, \ldots, z'_n)$ (or, for short: ψ_P) over variables $z_1, \ldots, z_n, z'_1, \ldots, z'_n$ such that for any $M \subseteq \text{At}(P)$ and any $M' \subseteq \text{At}(P)$ holds that M is a model of $P^{M'}$ if and only if $\psi_P[\alpha_M \cup \alpha_{M'}]$ evaluates to true, where $\alpha_M : \{z_1, \ldots, z_n\} \to \{0, 1\}$ is defined by letting $\alpha_M(z_i) = 1$ if and only if $d_i \in M$, and $\alpha_{M'} : \{z'_1, \ldots, z'_n\} \to \{0, 1\}$ is defined by letting $\alpha_{M'}(z'_i) = 1$ if and only if $d_i \in M'$, for all $1 \leq i \leq n$. We define $\psi_P = \bigwedge_{r \in P}(\psi^1_r \vee \psi^2_r)$, where $\psi^1_r = (z'_{i^3_1} \vee \cdots \vee z'_{i^3_c})$ and $\psi^2_r = ((z_{i^1_1} \vee \cdots \vee z_{i^1_a}) \leftarrow (z_{i^2_1} \wedge \cdots \wedge z_{i^2_b}))$ for $r = (d_{i^1_1} \vee \cdots \vee d_{i^1_a} \leftarrow d_{i^2_1}, \ldots, d_{i^2_b}, not\ d_{i^3_1}, \ldots, not\ d_{i^3_c})$. It is easy to verify that ψ_P satisfies the required property.

We now introduce the set X of existentially quantified variables of φ. For each contingent rule r_i of P we let $a^i_1, \ldots, a^i_{\ell_i}$ denote the atoms that occur in the head of r_i. For each r_i, we introduce variables $x^i_0, x^i_1, \ldots, x^i_{\ell_i}$, i.e., $X = \{\, x^i_j : 1 \leq i \leq k, 0 \leq j \leq \ell_i \,\}$. Furthermore, for each atom d_i, we add universally quantified variables y_i, z_i and w_i, i.e., $Y = \{\, y_i : 1 \leq i \leq n \,\}$, $Z = \{\, z_i : 1 \leq i \leq n \,\}$, and $W = \{\, w_i : 1 \leq i \leq n \,\}$.

We then construct ψ as follows:

$$
\begin{array}{rl}
\psi = & \psi_X \wedge \left(\psi^1_Y \vee \psi_W \vee \psi_{\text{min}}\right) \wedge \left(\psi^1_Y \vee \psi^2_Y\right);\\
\psi_X = & \displaystyle\bigwedge_{1 \leq i \leq k} \left(\bigvee_{0 \leq j \leq \ell_i} x^i_j \wedge \bigwedge_{0 \leq j < j' \leq \ell_i} (\neg x^i_j \vee \neg x^i_{j'}) \right);\\
\psi^1_Y = & \displaystyle\bigvee_{\substack{1 \leq i \leq k\\ 1 \leq j \leq \ell_i}} \psi^{i,j}_y \vee \bigvee_{d_i \in \text{Cont}(P)} \psi^{d_i}_y \vee \bigvee_{d_i \in \text{Comp}(P)} \neg y_i;\\
\psi^{d_m}_y = & (y_m \wedge \neg x^{i_1}_{j_1} \wedge \cdots \wedge \neg x^{i_u}_{j_u})\\
& \text{if } \{\, x^i_j : 1 \leq i \leq k, 1 \leq j \leq \ell_i, a^i_j = d_m \,\} =\\
& \{a^{i_1}_{j_1}, \ldots, a^{i_u}_{j_u}\}, \text{ and}\\
\psi^{d_m}_y = & \bot\\
& \text{if } \{\, x^i_j : 1 \leq i \leq k, 1 \leq j \leq \ell_i, a^i_j = d_m \,\} = \emptyset;\\
\psi^{i,j}_y = & (x^i_j \wedge \neg y_m) \qquad \text{where } a^i_j = d_m;\\
\psi^2_Y = & \psi_P(y_1, \ldots, y_n, y_1, \ldots, y_n);\\
\psi_W = & \displaystyle\bigvee_{1 \leq i \leq n} (w_i \leftrightarrow (y_i \leftrightarrow z_i));\\
\psi_{\text{min}} = & \psi^1_{\text{min}} \vee \psi^2_{\text{min}} \vee \psi^3_{\text{min}};\\
\psi^1_{\text{min}} = & \displaystyle\bigvee_{1 \leq i \leq n} (z_i \wedge \neg y_i);\\
\psi^2_{\text{min}} = & (\neg w_1 \wedge \cdots \wedge \neg w_m); \text{ and}\\
\psi^3_{\text{min}} = & \neg\psi_P(z_1, \ldots, z_n, y_1, \ldots, y_n).
\end{array}
$$

The idea behind this construction is the following. The variables in X represent guessing at most one atom in the head of each contingent rule to be true. Such a guess represents a possible answer set $M \subseteq \text{At}(P)$ (a proof of this can be found in the technical report). The formula ψ_X ensures that for each $1 \leq i \leq k$, exactly one x^i_j is set to true. The formula ψ^1_Y filters out every assignment in which the variables Y are not set corresponding to M. The formula ψ^2_Y filters out every assignment corresponding to a candidate $M \subseteq \text{At}(P)$ such that $M \not\models P$. The formula ψ_W filters out every assignment such that w_i is not set to the value $(y_i \text{ XOR } z_i)$. The formula ψ^1_{min} filters out every assignment where the variables Z correspond to a set M' such that $M' \not\subseteq M$. The formula ψ^2_{min} filters out every assignment where the variables Z correspond to the set M, by referring to the variables w_i. The formula ψ^3_{min}, finally, ensures that in every remaining assignment, the

variables Z do not correspond to a set $M' \subseteq M$ such that $M' \models P$. A full proof that P has an answer set if and only if $(\varphi, k) \in \exists^k\forall^*$-WSAT can be found in the technical report. □

Corollary 9. ASP-CONSISTENCY(#cont.rules) *is* $\exists^k\forall^*$-*complete.*

6 The $*$-k Hierarchy

We now turn our attention to the $*$-k hierarchy. Unlike in the k-$*$ hierarchy, in the canonical quantified Boolean satisfiability problems of the $*$-k hierarchy, we cannot add auxiliary variables to the second quantifier block whose truth assignment is not restricted. Therefore, because of similarity to the W-hierarchy, we believe that the classes of the $*$-k hierarchy are distinct. We will mainly focus on the first level of the $*$-k hierarchy. We begin with proving some basic properties. The main result in this section is a normalization result for $\forall^*\exists^k$-W[1].

Proposition 10. *If* para-co-NP $\subseteq \exists^*\forall^k$-W[P]*, then* NP $=$ co-NP.

Proof (sketch). With an argument similar to the one in the proof of Proposition 4, a polynomial-time reduction from UNSAT to SAT can be constructed. An additional technical observation needed for this case is that SAT is in NP also when the input is a Boolean circuit. □

Next, we show that the problem $\exists^*\forall^k$-WSAT is already $\exists^*\forall^k$-W[1]-hard when the input circuits are restricted to formulas in c-DNF, for any constant $c \geq 2$. In order to make our life easier, we switch our perspective to the co-problem $\forall^*\exists^k$-WSAT when stating and proving the following results. Note that the proofs of the following results make heavy use of the original normalization proof for the class W[1] by Downey and Fellows (1995; 1999).

Lemma 11. *For any* $u \geq 1$, $\forall^*\exists^k$-WSAT$(\Gamma_{1,u}) \leq_{\text{fpt}} \forall^*\exists^k$-WSAT$(s$-CNF$)$*, where* $s = 2^u + 1$.

Proof (sketch). The reduction is completely analogous to the reduction used in the proof of Downey and Fellows (1995, Lemma 2.1), where the presence of universally quantified variables is handled in four steps. In Steps 1 and 2, in which only the form of the circuit is modified, no changes are needed. In Step 3, universally quantified variables can be handled exactly as existentially quantified variables. Step 4 can be performed with only a slight modification, the only difference being that universally quantified variables appearing in the input circuit will also appear in the resulting clauses that verify whether a given product-of-sums or sum-of-products is satisfied. It is straightforward to verify that this reduction with the mentioned modifications works for our purposes. □

Theorem 12. $\forall^*\exists^k$-WSAT(2CNF) *is* $\forall^*\exists^k$-W[1]*-complete.*

Proof (sketch). Clearly $\forall^*\exists^k$-WSAT(2CNF) is in $\forall^*\exists^k$-W[1], since a 2CNF formula can be considered as a constant-depth circuit of weft 1. To show that $\forall^*\exists^k$-WSAT(2CNF) is $\forall^*\exists^k$-W[1]-hard, we give an fpt-reduction from $\forall^*\exists^k$-WSAT$(\Gamma_{1,u})$ to $\forall^*\exists^k$-WSAT(2CNF), for arbitrary $u \geq 1$. By Lemma 11, we know that we can reduce $\forall^*\exists^k$-WSAT$(\Gamma_{1,u})$ to $\forall^*\exists^k$-WSAT$(s$-CNF$)$, for $s = 2^u + 1$. We continue the reduction in multiple steps. In each step, we let C denote the circuit resulting from the previous step, and we let Y denote the universally quantified and X the existentially quantified input variables of C, and we let k denote the parameter value. We only briefly describe the last two steps, since these are completely analogous to constructions in the work of Downey and Fellows (1999).

Step 1: contracting the universally quantified variables. This step transforms C into a CNF formula C' such that each clause contains at most one variable in Y such that (C, k) is a yes-instance if and only if (C', k) is a yes-instance. We introduce new universally quantified variables Y' containing a variable y'_A for each set A of literals over Y of size at least 1 and at most s. Now, it is straightforward to construct a set D of polynomially many ternary clauses over Y and Y' such that the following property holds. An assignment α to $Y \cup Y'$ satisfies D if and only if for each subset $A = \{l_1, \ldots, l_b\}$ of literals over Y it holds that $\alpha(l_1) = \alpha(l_2) = \ldots = \alpha(l_b) = 1$ if and only if $\alpha(y'_A) = 1$. Note that we do not directly add the set D of clauses to the formula C'.

We introduce $k - 1$ many new existentially quantified variables $x^\star_1, \ldots, x^\star_{k-1}$. We add binary clauses to C' that enforce that the variables $x^\star_1, \ldots, x^\star_{k-1}$ all get the same truth assignment. Also, we add binary clauses to C' that enforce that each $x \in X$ is set to false if $x^\star_1$ is set to true.

We introduce $|D|$ many existentially quantified variables, including a variable x''_d for each clause $d \in D$. For each $d \in D$, we add the following clauses to C'. Let $d = (l_1, l_2, l_3)$, where each l_i is a literal over $Y \cup Y'$. We add the clauses $(\neg x''_d \vee \neg l_1)$, $(\neg x''_d \vee \neg l_2)$ and $(\neg x''_d \vee \neg l_3)$, enforcing that the clause d cannot be satisfied if x''_d is set to true.

We then modify the clauses of C as follows. Let $c = (l^x_1, \ldots, l^x_{s_1}, l^y_1, \ldots, l^y_{s_2})$ be a clause of C, where $l^x_1, \ldots, l^x_{s_1}$ are literals over X, and $l^y_1, \ldots, l^y_{s_2}$ are literals over Y. We replace c by the clause $(l^x_1, \ldots, l^x_{s_1}, x^\star_1, y'_B)$, where $B = \{l^y_1, \ldots, l^y_{s_2}\}$. Clauses c of C that contain no literals over the variables Y remain unchanged.

The idea of this reduction is the following. If $x^\star_1$ is set to true, then exactly one of the variables x''_d must be set to true, which can only result in an satisfying assignment if the clause $d \in D$ is not satisfied. Therefore, if an assignment α to the variables $Y \cup Y'$ does not satisfy D, there is a satisfying assignment of weight k that sets both $x^\star_1$ and x''_d to true, for some $d \in D$ that is not satisfied by α. Otherwise, we know that the value α assigns to variables y'_A corresponds to the value α assigns to $\bigwedge_{a \in A} a$, for $A \subseteq \text{Lit}(Y)$. Then any satisfying assignments of weight k for C is also a satisfying assignments of weight k for C'.

Step 2: making C antimonotone in X. This step transforms C into a circuit C' that has only negative occurrences of existentially quantified variables, and transforms k into k' depending only on k, such that (C, k) is a yes-instance if and only if (C', k') is a yes-instance. The reduction is completely analogous to the reduction in the proof of Downey and Fellows (1999, Theorem 10.6).

Step 3: contracting the existentially quantified variables. This step transforms C into a circuit C' in CNF

that contains only clauses with two variables in X and no variables in Y and clauses with one variable in X and one variable in Y, and transforms k into k' depending only on k, such that (C, k) is a yes-instance if and only if (C', k') is a yes-instance. The reduction is completely analogous to the reduction in the proof of Downey and Fellows (1999, Theorem 10.7). □

Corollary 13. *For any fixed integer $r \geq 2$, the problem $\exists^*\forall^k$-WSAT(r-DNF) is $\exists^*\forall^k$-W[1]-complete.*

6.1 Answer set programming and hardness for the $*$-k hierarchy

We now turn to another variant of our running example problem.

Theorem 14. ASP-CONSISTENCY(#disj.rules) *is $\exists^*\forall^k$-W[1]-hard.*

Proof. We give an fpt-reduction from $\exists^*\forall^k$-WSAT(3DNF), which we know to be $\exists^*\forall^k$-W[1]-hard from Corollary 13. This is, like the reduction in the proof of Theorem 7 above, a parameterized version of a reduction of Eiter and Gottlob (1995, Theorem 3). Let (φ, k) be an instance of $\exists^*\forall^k$-WSAT(3DNF), where $\varphi = \exists X.\forall Y.\psi$, $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$, $\psi = \delta_1 \vee \cdots \vee \delta_u$, and $\delta_\ell = l^\ell_1 \wedge l^\ell_2 \wedge l^\ell_3$ for each $1 \leq \ell \leq u$. By step 2 in the proof of Theorem 12, we may assume without loss of generality that all universally quantified variables $y_1, \ldots, y_m$ occur only positively in ψ. We construct a disjunctive program P. We consider the variables X and Y as atoms. In addition, we introduce fresh atoms $v_1, \ldots, v_n, w$, and y^j_i for all $1 \leq j \leq k$, $1 \leq i \leq m$. We let P consist of the following rules:

$$x_i \leftarrow \mathit{not}\ v_i \qquad \text{for } 1 \leq i \leq n; \quad (10)$$
$$v_i \leftarrow \mathit{not}\ x_i \qquad \text{for } 1 \leq i \leq n; \quad (11)$$
$$y^j_1 \vee \cdots \vee y^j_m \leftarrow \qquad \text{for } 1 \leq j \leq k; \quad (12)$$
$$y_i \leftarrow y^j_i \qquad \text{for } 1 \leq i \leq m, 1 \leq j \leq k; \quad (13)$$
$$y^j_i \leftarrow w \qquad \text{for } 1 \leq i \leq m, 1 \leq j \leq k; \quad (14)$$
$$w \leftarrow y^j_i, y^{j'}_i \qquad \text{for } 1 \leq i \leq m, 1 \leq j < j' \leq k; \quad (15)$$
$$w \leftarrow \sigma(l^\ell_1), \sigma(l^\ell_2), \sigma(l^\ell_3) \qquad \text{for } 1 \leq \ell \leq u; \quad (16)$$
$$w \leftarrow \mathit{not}\ w. \quad (17)$$

Here we let $\sigma(\neg x_i) = v_i$ for each $1 \leq i \leq n$; we let $\sigma(x_i) = x_i$ for each $1 \leq i \leq n$; and we let $\sigma(y_i) = y_i$ for each $1 \leq i \leq m$. Intuitively, v_i corresponds to $\neg x_i$. The main difference with the reduction of Eiter and Gottlob is that we use the rules in (12)–(15) to let the variables y_i represent an assignment of weight k to the variables in Y. Note that P has k disjunctive rules, namely the rules (12). A full proof that $(\varphi, k) \in \exists^*\forall^k$-WSAT if and only if P has an answer set can be found in the technical report. □

This hardness result holds even for the case where each atom occurs only a constant number of times in the input program. In order to show this, we consider the following polynomial-time transformation on disjunctive logic programs. Let P be an arbitrary program, and let x be an atom of P that occurs $\ell \geq 3$ many times. We introduce new atoms x_j for all $1 \leq j \leq \ell$. We replace each occurrence of x in P by a unique atom x_j. Then, to P, we add the rules $(x_{j+1} \leftarrow x_j)$, for all $1 \leq j < \ell$, and the rule $(x_1 \leftarrow x_\ell)$. We call the resulting program P'. It is straightforward to verify that P has an answer set if and only if P' has an answer set.

Proposition 15. ASP-CONSISTENCY(#disj.rules) *is $\exists^*\forall^k$-W[1]-hard, even when each atom occurs at most 3 times.*

Proof. The transformation described above can be repeatedly applied to ensure that each atom occurs at most 3 times. Since this transformation does not introduce new disjunctive rules, the result follows by Theorem 14. □

Repeated application of the described transformation also gives us the following result.

Corollary 16. ASP-CONSISTENCY(max.atom.occ.) *is* para-Σ^{P}_2*-complete.*

7 Robust Constraint Satisfaction

Next, we consider another application of our hardness theory to a reasoning problem that originates in the domain of knowledge representation. We consider the class of robust constraint satisfaction problems, introduced recently by Gottlob (2012) and Abramsky, Gottlob, and Kolaitis (2013). These problems are concerned with the question of whether every partial assignment of a particular size can be extended to a full solution, in the setting of constraint satisfaction problems. As we will see, a natural parameterized variant of this class of problems is complete for the class $\forall^k\exists^*$.

A *CSP instance* N is a triple (X, D, C), where X is a finite set of *variables*, the *domain* D is a finite set of *values*, and C is a finite set of *constraints*. Each constraint $c \in C$ is a pair (S, R), where $S = \mathrm{Var}(c)$, the *constraint scope*, is a finite sequence of distinct variables from X, and R, the *constraint relation*, is a relation over D whose arity matches the length of S, i.e., $R \subseteq D^r$ where r is the length of S.

Let $N = (X, D, C)$ be a CSP instance. A *partial instantiation* of N is a mapping $\alpha : X' \to D$ defined on some subset $X' \subseteq X$. We say that α *satisfies* a constraint $c = ((x_1, \ldots, x_r), R) \in C$ if $\mathrm{Var}(c) \subseteq X'$ and $(\alpha(x_1), \ldots, \alpha(x_r)) \in R$. If α satisfies all constraints of N then it is a *solution* of N. We say that α *violates* a constraint $c = ((x_1, \ldots, x_r), R) \in C$ if there is no extension β of α defined on $X' \cup \mathrm{Var}(c)$ such that $(\beta(x_1), \ldots, \beta(x_r)) \in R$.

Let k be a nonnegative integer. We say that a CSP instance $N = (X, D, C)$ is *k-robustly satisfiable* if for each instantiation $\alpha : X' \to D$ defined on some subset $X' \subseteq X$ of k many variables (i.e., $|X'| = k$) that does not violate any constraint in C, it holds that α can be extended to a solution for the CSP instance (X, D, C). Now consider the following parameterized problem.

ROBUST-CSP-SAT
Instance: A CSP instance (X, D, C), and a nonnegative integer k.
Parameter: k.
Question: Is (X, D, C) k-robustly satisfiable?

We show that this problem is complete for $\forall^k\exists^*$.

Theorem 17. ROBUST-CSP-SAT *is in $\forall^k\exists^*$.*

Proof. We give an fpt-reduction from ROBUST-CSP-SAT to $\forall^k\exists^*$-WSAT. Let (X, D, C, k) be an instance of ROBUST-CSP-SAT, where (X, D, C) is a CSP instance, $X = \{x_1, \ldots, x_n\}$, $D = \{d_1, \ldots, d_m\}$, and k is an integer. We

construct an instance (φ, k) of $\forall^k\exists^*$-WSAT. For the formula φ, we use propositional variables $Z = \{ z^i_j : 1 \leq i \leq n, 1 \leq j \leq m \}$ and $Y = \{ y^i_j : 1 \leq i \leq n, 1 \leq j \leq m \}$. Intuitively, the variables z^i_j will represent an arbitrary assignment α that assigns values to k variables in X. Any variable z^i_j represents that variable x_i gets assigned value d_j. The variables y^i_j will represent the solution β that extends the arbitrary assignment α. Similarly, any variable y^i_j represents that variable x_i gets assigned value d_j. We then let $\varphi = \forall Z.\exists Y.\psi$ with $\psi = (\psi^Z_{\text{proper}} \wedge \neg\psi^Z_{\text{violate}}) \rightarrow (\psi^{Y,Z}_{\text{corr}} \wedge \psi^Y_{\text{proper}} \wedge \bigwedge_{c \in C} \psi^Y_c)$. We will describe the subformulas of φ below, as well as the intuition behind them. We start with the formula ψ^Z_{proper}. This formula represents whether for each variable x_i at most one value is chosen for the assignment α. We let $\psi^Z_{\text{proper}} = \bigwedge_{1 \leq i \leq n} \bigwedge_{1 \leq j < j' \leq m} (\neg z^i_j \vee \neg z^i_{j'})$. Next, we consider the formula ψ^Z_{violate}. This subformula encodes whether the assignment α violates some constraint $c \in C$. We let $\psi^Z_{\text{violate}} = \bigvee_{c=(S,R) \in C} \bigwedge_{\overline{d} \in R} \bigvee_{z \in \Psi^{\overline{d},c}}$, where we define the set $\Psi^{\overline{d},c} \subseteq Z$ as follows. Let $c = ((x_{i_1}, \ldots, x_{i_r}), R) \in C$ and $\overline{d} = (d_{j_1}, \ldots, d_{j_r}) \in R$. Then we let $\Psi^{\overline{d},c} = \{ z^{i_\ell}_j : 1 \leq \ell \leq r, j \neq j_\ell \}$. Intuitively, the set $\Psi^{\overline{d},c}$ contains the variables z^i_j that represent those variable assignments in α that prevent that β satisfies c by assigning $\text{Var}(c)$ to $\overline{d}$. Then, the formula ψ^Y_{proper} ensures that for each variable x_i exactly one value d_j is chosen in β. We define: $\psi^Y_{\text{proper}} = \bigwedge_{1 \leq i \leq n} [\bigvee_{1 \leq j \leq m} y^i_j \wedge \bigwedge_{1 \leq j < j' \leq m} (\neg y^i_j \vee \neg y^i_{j'})]$. Next, the formula $\psi^{Y,Z}_{\text{corr}}$ ensures that β is indeed an extension of α. We define: $\psi^{Y,Z}_{\text{corr}} = \bigwedge_{1 \leq i \leq n} \bigwedge_{1 \leq j \leq m} (z^i_j \rightarrow y^i_j)$. Finally, for each $c \in C$, the formula ψ^Y_c represents whether β satisfies the constraint c. Let $c = ((x_{i_1}, \ldots, x_{i_r}), R) \in C$. We define $\psi^Y_c = \bigvee_{(d_{j_1}, \ldots, d_{j_r}) \in R} \bigwedge_{1 \leq \ell \leq r} y^{i_\ell}_{j_\ell}$. A full proof that $(X, D, C, k) \in$ ROBUST-CSP-SAT if and only if $(\varphi, k) \in \forall^k\exists^*$-WSAT can be found in the technical report. □

In order to prove $\forall^k\exists^*$-hardness, we need the following technical lemma.

Lemma 18. *Let (φ, k) be an instance of $\exists^k\forall^*$-WSAT with $\varphi = \exists X.\forall Y.\psi$. 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 any assignment $\alpha : X \rightarrow \{0,1\}$ that has weight $m \neq k$, it holds that $\forall Y'.\psi'[\alpha]$ is true.*

Proof (sketch). We introduce a set Z of additional universally quantified variables, and use these to verify whether k many existentially quantified variables are set to true. We do so by constructing a propositional formula χ containing variables in X and Z that is satisfiable if and only if exactly k many variables in X are set to true. We let $\chi' = (\exists Z.\chi) \rightarrow \psi$. We then know that $\forall Y.\chi'[\alpha]$ is true for all truth assignments $\alpha : X \rightarrow \{0,1\}$ of weight $m \neq k$. Moreover, the formula $\forall Y.\chi'$ is equivalent to the formula $\forall Y \cup Z.(\neg\chi) \vee \psi$. □

Theorem 19. ROBUST-CSP-SAT *is $\forall^k\exists^*$-hard, even when the domain size $|D|$ is restricted to* 2.

Proof. We give an fpt-reduction from $\forall^k\exists^*$-WSAT(3CNF) to ROBUST-CSP-SAT. Let (φ, k) be an instance of $\forall^k\exists^*$-WSAT(3CNF), with $\varphi = \forall X.\exists Y.\psi$, and $\psi = c_1 \wedge \cdots \wedge c_u$. By Lemma 18, we may assume without loss of generality that for any assignment $\alpha : X \rightarrow \{0,1\}$ of weight $m \neq k$, we have that $\exists Y.\psi[\alpha]$ is false. We construct an instance (Z, D, C, k) of ROBUST-CSP-SAT as follows. We define the set Z of variables by $Z = X \cup Y'$, where $Y' = \{ y^i : y \in Y, 1 \leq i \leq 2k+1 \}$, and we let $D = \{0,1\}$. We will define the set C of constraints below, by representing them as a set of clauses whose length is bounded by $f(k)$, for some fixed function f.

The intuition behind the construction of C is the following. We replace each variable $y \in Y$, by $2k+1$ copies y^i of it. Assigning a variable $y \in Y$ to a value $b \in \{0,1\}$ will then correspond to assigning a majority of variables y^i to b, i.e., assigning at least $k+1$ variables y^i to b. In order to encode this transformation in the constraints of C, intuitively, we will replace each occurrence of a variable y by the conjuction $\psi_y = \bigwedge_{1 \leq i_1 < \cdots < i_{k+1} \leq 2k+1} (y^{i_1} \vee \cdots \vee y^{i_{k+1}})$, and replace each occurrence of a literal $\neg y$ by a similar conjunction. We will then multiply the resulting formula out into CNF. Note that whenever a majority of variables y^i is set to $b \in \{0,1\}$, then the formula ψ_y will also evaluate to b.

In the construction of C, we will directly encode the CNF formula that is a result of the transformation described above. For each literal $l = y \in Y$, let l^i denote y^i, and for each literal $l = \neg y$ with $y \in Y$, let l^i denote $\neg y^i$. For each literal l over the variables $X \cup Y$, we define a set $\sigma(l)$ of clauses: $\sigma(l) = (l^{i_1} \vee \cdots \vee l^{i_{k+1}}) : 1 \leq i_1 < \cdots < i_{k+1} \leq 2k+1 \}$ if l is a literal over Y, and $\sigma(l) = l$ if l is a literal over X.

Note that for each literal l, it holds that $|\sigma(l)| \leq g(k) = \binom{2k+1}{k+1}$. Next, for each clause $c_i = l^i_1 \vee l^i_2 \vee l^i_3$ of ψ, we introduce to C a set $\sigma(c_i)$ of clauses: $\sigma(c_i) = \{ d_1 \vee d_2 \vee d_3 : d_1 \in \sigma(l^i_1), d_2 \in \sigma(l^i_2), d_3 \in \sigma(l^i_3) \}$. Note that $|\sigma(c_i)| \leq g(k)^3$. Formally, we let C be the set of constraints corresponding to the set $\bigcup_{1 \leq i \leq u} \sigma(c_i)$ of clauses. Since each such clause is of length at most $3(k+1)$, representing a clause by means of a constraint can be done by specifying $\leq 2^{3(k+1)} - 1$ tuples, i.e., all tuples satisfying the clause. Therefore, the instance (Z, D, C, k) can be constructed in fpt-time. A full proof that $(\varphi, k) \in \forall^k\exists^*$-WSAT(3CNF) if and only if $(Z, D, C, k) \in$ ROBUST-CSP-SAT can be found in the technical report. □

8 Conclusion

We developed a general theoretical framework that supports the classification of parameterized problems on whether they admit an fpt-reduction to SAT or not. Our theory is based on two new hierarchies of complexity classes, the k-$*$ and $*$-k hierarchies. We illustrated the use of this theoretical toolbox by means of two case studies, in which we studied the complexity of the consistency problem for disjunctive answer set programming and a robust version of constraint satisfaction, with respect to various natural parameters. There are many more problems that occur in knowledge representation and reasoning where our theory can be used. In the technical report corresponding to this paper, we use our theory to analyze various additional problems, including the problem of minimizing DNF formulas and the problem of minimizing implicant cores. Additionally, we illustrate the robustness of

our theory by showing a number of complete problems for the newly introduced classes from various domains, as well as providing alternative characterizations of the complexity classes based on first-order model checking and alternating Turing machines. There are many more problems that can be analyzed within our framework.

We focused our attention on the range between the first and the second level of the PH, since many natural problems lie there (Schaefer and Umans, 2002). In general, any fpt-reduction from a problem whose complexity is higher in the PH to a lower level in the PH would be interesting. With this more general aim in mind, it would be helpful to have tools to gather evidence that an fpt-reduction across some complexity border in the PH is not possible. We hope that this paper provides a starting point for further developments.

References

Abramsky, S.; Gottlob, G.; and Kolaitis, P. G. 2013. Robust constraint satisfaction and local hidden variables in quantum mechanics. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013*. AAAI Press/IJCAI.

Ben-Eliyahu, R., and Dechter, R. 1994. Propositional semantics for disjunctive logic programs. *Ann. Math. Artif. Intell.* 12(1):53–87.

Biere, A.; Cimatti, A.; Clarke, E. M.; and Zhu, Y. 1999. Symbolic model checking without BDDs. In Cleaveland, R., ed., *Tools and Algorithms for Construction and Analysis of Systems, 5th International Conference, TACAS '99, Part of the European Joint Conferences on Software, ETAPS'99, Amsterdam, The Netherlands, March 22-28, 1999, Proceedings*, volume 1579 of *Lecture Notes in Computer Science*, 193–207. Springer Verlag.

Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds. 2009. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press.

Biere, A. 2009. Bounded model checking. In Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds., *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*. IOS Press. 457–481.

Brewka, G.; Eiter, T.; and Truszczynski, M. 2011. Answer set programming at a glance. *Communications of the ACM* 54(12):92–103.

Cai, J., and Hemachandra, L. 1986. The boolean hierachy: hardware over NP. In *Proceedings of the 1st Structure in Compexity Theory Conference*, number 223 in Lecture Notes in Computer Science, 105–124. Springer Verlag.

Downey, R. G., and Fellows, M. R. 1995. Fixed-parameter tractability and completeness. II. On completeness for $W[1]$. *Theoretical Computer Science* 141(1-2):109–131.

Downey, R. G., and Fellows, M. R. 1999. *Parameterized Complexity*. Monographs in Computer Science. New York: Springer Verlag.

Eiter, T., and Gottlob, G. 1995. On the computational cost of disjunctive logic programming: propositional case. *Ann. Math. Artif. Intell.* 15(3-4):289–323.

Fages, F. 1994. Consistency of Clark's completion and existence of stable models. *Methods of Logic in Computer Science* 1(1):51–60.

Fichte, J. K., and Szeider, S. 2013. Backdoors to normality for disjunctive logic programs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2013*, 320–327. AAAI Press.

Flum, J., and Grohe, M. 2003. Describing parameterized complexity classes. *Information and Computation* 187(2):291–319.

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Berlin: Springer Verlag.

Garey, M. R., and Johnson, D. R. 1979. *Computers and Intractability*. San Francisco: W. H. Freeman and Company, New York.

Gebser, M.; Kaufmann, B.; Neumann, A.; and Schaub, T. 2007. Conflict-driven answer set solving. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007*, 386–392. MIT Press.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Comput.* 9(3/4):365–386.

Giunchiglia, E.; Lierler, Y.; and Maratea, M. 2006. Answer set programming based on propositional satisfiability. *Journal of Automated Reasoning* 36:345–377.

Gomes, C. P.; Kautz, H.; Sabharwal, A.; and Selman, B. 2008. Satisfiability solvers. In *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*. Elsevier. 89–134.

Gottlob, G. 2012. On minimal constraint networks. *Artificial Intelligence* 191-192:42–60.

de Haan, R., and Szeider, S. 2014. Fixed-parameter tractable reductions to SAT. Manuscript, submitted, February 2014.

Janhunen, T.; Niemelä, I.; Seipel, D.; Simons, P.; and You, J.-H. 2006. Unfolding partiality and disjunctions in stable model semantics. *ACM Trans. Comput. Log.* 7(1):1–37.

Lee, J., and Lifschitz, V. 2003. Loop formulas for disjunctive logic programs. In Palamidessi, C., ed., *Logic Programming, 19th International Conference, ICLP 2003, Mumbai, India, December 9-13, 2003, Proceedings*, volume 2916 of *Lecture Notes in Computer Science*, 451–465. Springer Verlag.

Lifschitz, V., and Razborov, A. 2006. Why are there so many loop formulas? *ACM Trans. Comput. Log.* 7(2):261–268.

Lin, F., and Zhao, X. 2004. On odd and even cycles in normal logic programs. In Cohn, A. G., ed., *Proceedings of the 19th national conference on Artifical intelligence (AAAI 04)*, 80–85. AAAI Press.

Malik, S., and Zhang, L. 2009. Boolean satisfiability from theoretical hardness to practical success. *Communications of the ACM* 52(8):76–82.

Marek, V. W., and Truszczynski, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-Year Perspective*. Springer. 169–181.

Meyer, A. R., and Stockmeyer, L. J. 1972. The equivalence problem for regular expressions with squaring requires exponential space. In *SWAT*, 125–129. IEEE Computer Soc.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications. Oxford: Oxford University Press.

Papadimitriou, C. H. 1994. *Computational Complexity*. Addison-Wesley.

Pfandler, A.; Rümmele, S.; and Szeider, S. 2013. Backdoors to abduction. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013*. AAAI Press/IJCAI.

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

Schaefer, M., and Umans, C. 2002. Completeness in the polynomial-time hierarchy: A compendium. *SIGACT News* 33(3):32–49.

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

Wrathall, C. 1976. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science* 3(1):23–33.