Electronic Colloquium on Computational Complexity, Report No. 143 (2014)

Compendium of Parameterized Problems at Higher Levels of the Polynomial Hierarchy

Ronald de Haan*, Stefan Szeider*

Institute of Information Systems
Vienna University of Technology

dehaan@kr.tuwien.ac.at stefan@szeider.net

Abstract

We present a list of parameterized problems together with a complexity classification of whether they allow a fixed-parameter tractable reduction to SAT or not. These parameterized problems are based on problems whose complexity lies at the second level of the Polynomial Hierarchy or higher. The list will be updated as necessary.

Contents

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

ISSN 1433-8092

1 Preliminaries

The remarkable performance of today's SAT solvers (see, e.g., [49]) offers a practically successful strategy for solving NP-complete combinatorial problems by reducing them in polynomial time to SAT. In order to apply this strategy to problems that are harder than NP, one needs to employ reductions that are more powerful than polynomial-time reductions. A compelling option for such reductions are fixed-parameter tractable reductions (i.e., reductions that are computable in time $f(k)n^{O(1)}$ for some computable function f) as they can exploit some structural aspects of the problem instances in terms of a problem parameter k. In this compendium, we give a list of parameterized problems that are based on problems at higher levels of the Polynomial Hierarchy, together with a complexity classification of whether they allow a (many-to-one or Turing) fpt-reduction to SAT or not.

The compendium that we provide is similar in concept to the compendia by Schaefer and Umans [46] and Cesati [13], that also list problems along with their computational complexity. We group the list by the type of problems. A list of problems grouped by their complexity can be found at the end of this paper. First, we give an overview of the parameterized complexity classes involved in the classification of whether problems allow an fpt-reduction to SAT.

Computational Complexity We assume that the reader is familiar with basic notions from the theory of computational complexity, such as the complexity classes P and NP. For more details, we refer to textbooks on the topic (cf. [3, 43]).

There are many natural decision problems that are not contained in the classical complexity classes P and NP (under some common complexity-theoretic assumptions). The *Polynomial Hierarchy* [40, 43, 47, 50] contains a hierarchy of increasing complexity classes Σ^{P}_i, 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^{P}_i-formulas. For each nonnegative integer $i \leq 0$, the complexity class Σ^{P}_i can be characterized as the closure of the problem QSAT$_i$ under polynomial-time reductions [47, 50]. The Σ^{P}_i-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^{P}_0 coincides with P, and the class Σ^{P}_1 coincides with NP. For each $i \geq 1$, the class Π^{P}_i is defined as co-Σ^{P}_i.

The classes Σ^{P}_i and Π^{P}_i can also be defined by means of nondeterministic Turing machines with an oracle. For any complexity class C, we let NP^C be the set of decision problems that is decided in polynomial time by a nondeterministic Turing machine with an oracle for a problem that is complete for the class C. Then, the classes Σ^{P}_i and Π^{P}_i, for $i \geq 0$, can be equivalently defined by letting $\Sigma^{\mathrm{P}}_0 = \Pi^{\mathrm{P}}_0 = \mathrm{P}$, and for each $i \geq 1$ letting $\Sigma^{\mathrm{P}}_i = \mathrm{NP}^{\Sigma^{\mathrm{P}}_{i-1}}$ and $\Pi^{\mathrm{P}}_i = \text{co-NP}^{\Sigma^{\mathrm{P}}_{i-1}}$.

The Polynomial Hierarchy also includes complexity classes between Σ^{P}_i and Π^{P}_i, on the one hand, and $\Sigma^{\mathrm{P}}_{i+1}$ and Π^{P}_{i+1}, on the other hand. The class $\Delta^{\mathrm{P}}_{i+1}$ consists of all decision problems that are decided in polynomial time by a deterministic Turing machine with an oracle for a problem that is complete for the class Σ^{P}_i. Similarly, the class $\Theta^{\mathrm{P}}_{i+1}$ consists of all decision problems that are decided in polynomial time by a deterministic Turing machine with an oracle for a problem that is complete for the class Σ^{P}_i, with the restriction that the Turing machine is only allowed to make $O(\log n)$ oracle queries, where n is the input size.

Many natural decision problems are located between NP and co-NP on the one hand, and Θ^{P}_2 on the other hand. The *Boolean Hierarchy* (BH) [12, 14, 36] consists of a hierarchy of complexity classes BH$_i$, for each $i \geq 1$, that can be used to classify the complexity of decision problems between NP and Θ_2^{P}. Each class BH_i can be characterized as the class of problems that can be reduced to the problem BH_i-SAT, which is defined inductively as follows. The problem BH_1-SAT consists of all sequences (φ) of length 1, where φ is a satisfiable propositional formula. For even $i \geq 2$, the problem BH_i-SAT consists of all sequences $(\varphi_1, \dots, \varphi_i)$ of propositional formulas such that both $(\varphi_1, \dots, \varphi_{i-1}) \in \text{BH}_{(i-1)}$-SAT and φ_i is unsatisfiable. For odd $i \geq 2$, the problem BH_i-SAT consists of all sequences $(\varphi_1, \dots, \varphi_i)$ of propositional formulas such that $(\varphi_1, \dots, \varphi_{i-1}) \in \text{BH}_{(i-1)}$-SAT or φ_i is satisfiable. The class BH_2 is also denoted by DP, and the problem BH_2-SAT is also denoted by SAT-UNSAT. The class BH is defined as the union of all BH_i, for $i \geq 1$. It holds that $\text{NP} \cup \text{co-NP} \subseteq \text{BH}_2 \subseteq \text{BH}_3 \subseteq \cdots \subseteq \text{BH} \subseteq \Theta_2^{\text{P}}$.

Parameterized Complexity We introduce some core notions from parameterized complexity theory. For an in-depth treatment we refer to other sources [17, 18, 27, 42]. 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 main 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*.

The parameterized complexity classes W[t], $t \geq 1$, W[SAT] and W[P] can be used to give evidence that a given parameterized problem is not fixed-parameter tractable. These classes 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 $\text{Nodes}(C)$ denote the set of all nodes of a circuit C. We say that a circuit C is in *negation normal form* if all negation nodes in C have variables as inputs. 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 $\text{CIRC}_{t,u}$. We denote the class of all Boolean circuits by CIRC, and the class of all Boolean formulas by FORM. 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 $\text{W[t]} = [\ \{\, p\text{-WSAT}[\text{CIRC}_{t,u}] : u \geq 1 \,\}\ ]^{\text{fpt}}$ for all $t \geq 1$. The classes W[SAT] and W[P] are defined by letting $\text{W[SAT]} = [\ p\text{-WSAT}[\text{FORM}]\ ]^{\text{fpt}}$ and $\text{W[P]} = [\ p\text{-WSAT}[\text{CIRC}]\ ]^{\text{fpt}}$.

Let K be a classical complexity class, e.g., NP. The parameterized complexity class para-K is 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-K consists of all problems that are in K after a precomputation that only involves the parameter. The class para-NP can also be defined via nondeterministic fpt-algorithms [26]. The class para-K can be seen as a direct analogue of the class K in parameterized complexity.

We define the following (trivial) parameterization of SAT, the satisfiability problem for propositional logic. We let $\text{SAT} = \{ (\varphi, 1) : \varphi \in \text{SAT} \}$. In other words, SAT is the parameterized variant of SAT where the parameter is the constant value 1. Similarly, we let $\text{UNSAT} = \{ (\varphi, 1) : \varphi \in \text{UNSAT} \}$. The problem SAT is para-NP-complete, and the problem UNSAT is para-co-NP-complete. In other words, the class para-NP consists of all parameterized problems that can be fpt-reduced to SAT, and para-co-NP consists of all parameterized problems that can be fpt-reduced to UNSAT.

Another analogue to the classical complexity class K is the parameterized complexity class XK^{nu}, that is defined as the class of those parameterized problems P whose slices P_k are in K, i.e., for each positive integer k the classical problem $P_k = \{ x : (x, k) \in P \}$ is in K [17]. For instance, the class XP^{nu} consists of those parameterized problems whose slices are decidable in polynomial time. Note that this definition is non-uniform, that is, for each positive integer k there might be a completely different polynomial-time algorithm that witnesses that P_k is polynomial-time solvable. There are also uniform variants XK of these classes XK^{nu}. We define XP to be the class of parameterized problems Q for which there exists a computable function f and an algorithm A that decides whether $(x, k) \in Q$ in time $|x|^{f(k)}$ [17, 26, 27]. Similarly, we define XNP to be the class of parameterized problems that are decidable in nondeterministic time $|x|^{f(k)}$. Its dual class we denote by Xco-NP. Alternatively, we can view XNP as the class of parameterized problems for which there exists an xp-reduction to SAT and Xco-NP as the class of parameterized problems for which there exists an xp-reduction to UNSAT.

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. [45]). 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 [22, 25, 33, 44]. 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.

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 seem not to be para-Σ_2^{P}-hard nor para-Π_2^{P}-hard either. Recently, several complexity classes have been introduced to classify such intermediate problems [33, 34]. These parameterized complexity classes are dubbed the k-$*$ class and the $*$-k hierarchy, inspired by their definition, which is based on the following weighted variants of the quantified Boolean satisfiability problem that is canonical for the second level of the PH. Let $\mathcal{C}$ be a class of Boolean circuits. The problem $\exists^k\forall^*$-WSAT($\mathcal{C}$) provides the foundation for the k-$*$ class.

$\exists^k\forall^*$-WSAT
Instance: A quantified Boolean formula $\exists X.\forall Y.\psi$, 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?

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?

The parameterized complexity class $\exists^k\forall^*$ (also called the k-$*$ class) is then defined as follows:

$$\exists^k\forall^* = \quad [\ \exists^k\forall^*\text{-WS}\textsc{at}\ ]^{\text{fpt}}.$$

Similarly, the classes of the $*$-k hierarchy are defined as follows:

$$\begin{aligned}
\exists^*\forall^k\text{-W[t]} &= \quad [\ \{\ \exists^*\forall^k\text{-WS}\textsc{at}(\textsc{circ}_{t,u}) : u \geq 1\ \}\ ]^{\text{fpt}},\\
\exists^*\forall^k\text{-W[SAT]} &= \quad [\ \exists^*\forall^k\text{-WS}\textsc{at}(\textsc{form})\ ]^{\text{fpt}},\ \text{and}\\
\exists^*\forall^k\text{-W[P]} &= \quad [\ \exists^*\forall^k\text{-WS}\textsc{at}(\textsc{circ})\ ]^{\text{fpt}}.
\end{aligned}$$

Note that these definitions are entirely analogous to those of the parameterized complexity classes of the W-hierarchy [17]. The following inclusion relations hold between the classes of the $*$-k hierarchy:

$$\exists^*\forall^k\text{-W[1]} \subseteq \exists^*\forall^k\text{-W[2]} \subseteq \cdots \subseteq \exists^*\forall^k\text{-W[SAT]} \subseteq \exists^*\forall^k\text{-W[P]}.$$

Dual to the classical complexity class Σ^{P}_2 is its co-class Π^{P}_2, whose canonical complete problem is complementary to the problem QS$\textsc{at}_2$. Similarly, we can define dual classes for the k-$*$ class and for each of the parameterized complexity classes in the $*$-k hierarchy. These co-classes are based on problems complementary to the problems $\exists^k\forall^*$-WS$\textsc{at}$ and $\exists^*\forall^k$-WS$\textsc{at}$, i.e., these problems have as yes-instances exactly the no-instances of $\exists^k\forall^*$-WS$\textsc{at}$ and $\exists^*\forall^k$-WS$\textsc{at}$, respectively. Equivalently, these complementary problems can be considered as variants of $\exists^k\forall^*$-WS$\textsc{at}$ and $\exists^*\forall^k$-WS$\textsc{at}$ where the existential and universal quantifiers are swapped, and are therefore denoted with $\forall^k\exists^*$-WS$\textsc{at}$ and $\forall^*\exists^k$-WS$\textsc{at}$. 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].

The class $\exists^k\forall^*$ includes the class para-co-NP as a subset, and is contained in the class Xco-NP as a subset. Similarly, each of the classes $\exists^*\forall^k$-W[t] include the the class para-NP as a subset, and is contained in the class XNP. Under some common complexity-theoretic assumptions, the class $\exists^k\forall^*$ can be separated from para-NP on the one hand, and para-Σ^{P}_2 on the other hand. In particular, assuming that NP $\neq$ co-NP, it holds that $\exists^k\forall^* \not\subseteq$ para-NP, that para-NP $\not\subseteq \exists^k\forall^*$ and that $\exists^k\forall^* \subsetneq$ para-Σ^{P}_2 [33, 34]. Similarly, the classes $\exists^*\forall^k$-W[t] can be separated from para-co-NP and para-Σ^{P}_2. Assuming that NP $\neq$ co-NP, it holds that $\exists^*\forall^k$-W[1] $\not\subseteq$ para-co-NP, that para-co-NP $\not\subseteq \exists^*\forall^k$-W[P] and thus in particular that para-co-NP $\not\subseteq \exists^*\forall^k$-W[1], and that $\exists^*\forall^k$-W[P] $\subsetneq$ para-Σ^{P}_2 [33, 34].

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., [6, 19, 39]) and extends the scope of SAT solvers to problems in the class Δ^{P}_2, but not to problems that are Σ^{P}_2-hard or Π^{P}_2-hard. Also here, switching from polynomial-time to fpt-time provides a significant increase in power. The class para-Δ^{P}_2 contains all parameterized problems that can be decided by an fpt-algorithm that can query a SAT solver multiple times (i.e., by an fpt-time Turing reduction to SAT). In addition, one could restrict the number of queries that the algorithm is allowed to make. The class para-Θ^{P}_2 consists of all parameterized problems that can de decided by an fpt-algorithm that can query a SAT solver at most $f(k)\log n$ many times, where k is the parameter value, n is the input size, and f is some computable function. Restricting the number of queries even further, we define the parameterized complexity class $\text{FPT}^{\text{NP}}[f(k)]$ as the class of all parameterized problems that can be decided by an fpt-algorithm that can query a SAT solver at most $f(k)$ times, where k is the parameter value and f is some computable function [32, 34].

2 Propositional Logic Problems

We start with the quantified circuit satisfiability problems on which the k-$*$ and $*$-k hierarchies are based. We present only a two canonical forms of the problems in the k-$*$ hierarchy. For problems in the $*$-k hierarchy, we let $\mathcal{C}$ range over classes of Boolean circuits.

$\exists^k\forall^*$-WS$\textsc{at}(\mathcal{C})$ *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 of weight k, such that for all truth assignments β to Y the assignment $\alpha \cup \beta$ satisfies C?
Complexity: $\exists^k\forall^*$-complete [33, 34].

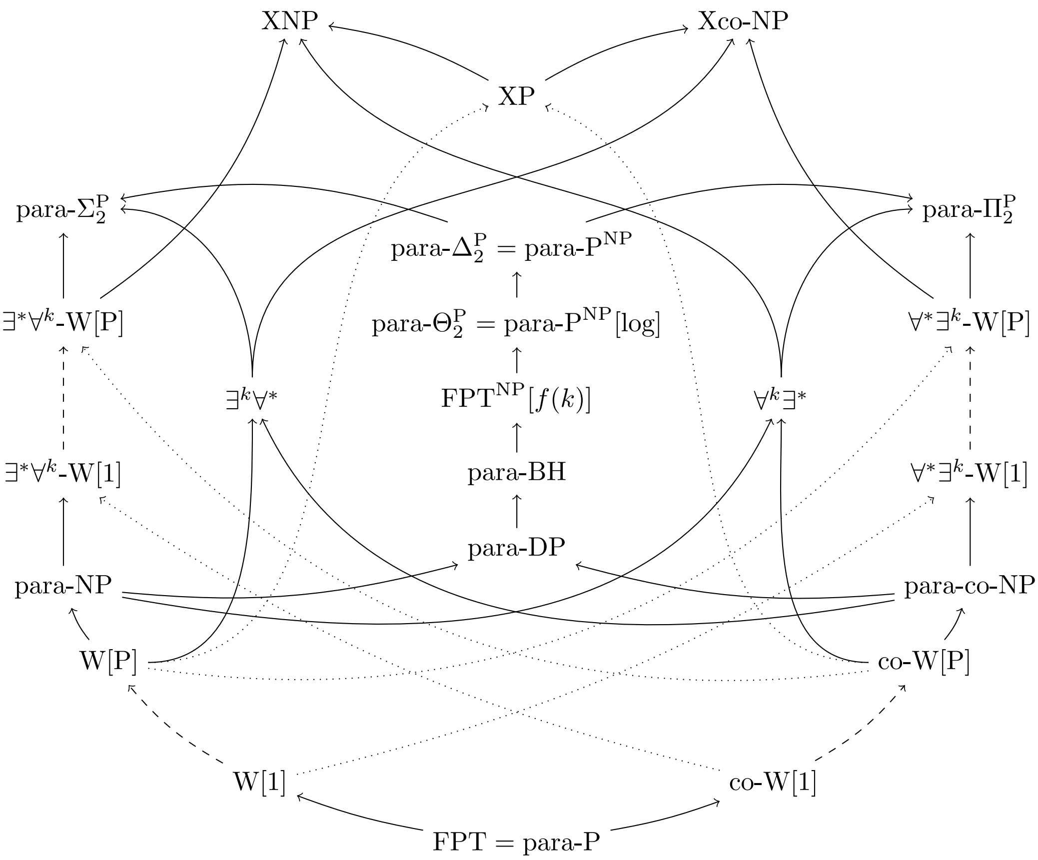

Figure 1: An overview of parameterized complexity classes up to the second level of the Polynomial Hierarchy

$\exists^k\forall^*$-WSat
Instance: A quantified Boolean formula $\phi = \exists X.\forall Y.\psi$, and an integer k.
Parameter: k.
Question: Does there exist a truth assignment α to X with weight k, such that $\forall Y.\psi[\alpha]$ evaluates to true?

Complexity: $\exists^k\forall^*$-complete [33, 34].

$\exists^k\forall^*$-WSat(3DNF)
Instance: A quantified Boolean formula $\phi = \exists X.\forall Y.\psi$ with $\psi \in$ 3DNF, and an integer k.
Parameter: k.
Question: Does there exist a truth assignment α to X with weight k, such that $\forall Y.\psi[\alpha]$ evaluates to true?

Complexity: $\exists^k\forall^*$-complete [33, 34].

$\exists^*\forall^k$-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 of weight k the assignment $\alpha \cup \beta$ satisfies C?
Complexity: $\exists^*\forall^k$-W[t]-complete when restricted to circuits of weft t, for any $t \geq 1$ (by definition); $\exists^*\forall^k$-W[SAT]-complete if $\mathcal{C} = \text{FORM}$ (by definition); $\exists^*\forall^k$-W[P]-complete if $\mathcal{C} = \text{CIRC}$ (by definition).

2.1 Weighted Quantified Boolean Satisfiability in the $*$-k Hierarchy

Consider the following variants of $\exists^k\forall^*$-WSAT, most of which are $\exists^k\forall^*$-complete.

$\exists^{\leq k}\forall^*$-WSAT *Instance:* A quantified Boolean formula $\phi = \exists X.\forall Y.\psi$, and an integer k. *Parameter:* k. *Question:* Does there exist an assignment α to X with weight at most k, such that $\forall Y.\psi[\alpha]$ evaluates to true?
Complexity: $\exists^k\forall^*$-complete [34].

$\exists^{\geq k}\forall^*$-WSAT *Instance:* A quantified Boolean formula $\phi = \exists X.\forall Y.\psi$, and an integer k. *Parameter:* k. *Question:* Does there exist an assignment α to X with weight at least k, such that $\forall Y.\psi[\alpha]$ evaluates to true?
Complexity: para-Σ_2^{P}-complete [34].

$\exists^{n-k}\forall^*$-WSAT *Instance:* A quantified Boolean formula $\phi = \exists X.\forall Y.\psi$, and an integer k. *Parameter:* k. *Question:* Does there exist an assignment α to X with weight $\|X\| - k$, such that $\forall Y.\psi[\alpha]$ evaluates to true?
Complexity: $\exists^k\forall^*$-complete [34].

2.2 Weighted Quantified Boolean Satisfiability for the k-$*$ Classes

The following variant of $\exists^*\forall^k$-WSAT is $\exists^*\forall^k$-W[1]-complete.

$\exists^*\forall^k$-WSAT(2DNF) *Instance:* A quantified Boolean formula $\varphi = \exists X.\forall Y.\psi$ with $\psi \in$ 2DNF, and an integer k *Parameter:* k. *Question:* Does there exist an assignment α to X, such that for all assignments β to Y of weight k the assignment $\alpha \cup \beta$ satisfies ψ?
Complexity: $\exists^*\forall^k$-W[1]-complete [33, 34].

Let $d \geq 2$ be an arbitrary constant. Then the following problem is also $\exists^*\forall^k$-W[1]-complete.

$\exists^*\forall^k$-WSAT(d-DNF) *Instance:* A quantified Boolean formula $\varphi = \exists X.\forall Y.\psi$ with $\psi \in d$-DNF, and an integer k *Parameter:* k. *Question:* Does there exist an assignment α to X, such that for all assignments β to Y of weight k the assignment $\alpha \cup \beta$ satisfies ψ?
Complexity: $\exists^*\forall^k$-W[1]-complete [33, 34].

The problem $\exists^*\forall^k$-WSAT(2-DNF) is $\exists^*\forall^k$-W[1]-hard, even when we restrict the input formula to be anti-monotone in the universal variables, i.e., the universal variables occur only in negative literals [33, 34].

Let C be a Boolean circuit with input nodes Z that is in negation normal form, and let $Y \subseteq Z$ be a subset of the input nodes. We say that C is *monotone in* Y if the only negation nodes that occur in the circuit C act on input nodes in $Z \backslash Y$, i.e., input nodes in Y can appear only positively in the circuit. Similarly, we say that C is *anti-monotone in* Y if the only nodes that have input nodes in Y as input are negatio nodes, i.e., all input nodes in Y appear only negatively in the circuit. The following problems are $\exists^*\forall^k$-W[P]-complete.

$\exists^*\forall^k$-WSAT($\forall$-monotone) *Instance:* A Boolean circuit $C \in$ CIRC over two disjoint sets X and Y of variables, that is in negation normal form and that is monotone in Y, 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 of weight k the assignment $\alpha \cup \beta$ satisfies C?
Complexity: $\exists^*\forall^k$-W[P]-complete [34].

$\exists^*\forall^k$-WSAT($\forall$-anti-monotone) *Instance:* A Boolean circuit $C \in$ CIRC over two disjoint sets X and Y of variables, that is in negation normal form and that is anti-monotone in Y, 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 of weight k the assignment $\alpha \cup \beta$ satisfies C?
Complexity: $\exists^*\forall^k$-W[P]-complete [34].

2.3 Quantified Boolean Satisfiability with Bounded Treewidth

Let $\psi = \delta_1 \vee \cdots \vee \delta_u$ be a DNF formula. For any subset $Z \subseteq \text{Var}(\psi)$ of variables, we define *the incidence graph* $\text{IG}(Z, \psi)$ *of* ψ *with respect to* Z to be the graph $\text{IG}(Z, \psi) = (V, E)$, where $V = Z \cup \{\delta_1, \dots, \delta_u\}$ and $E = \{ \{\delta_j, z\} : 1 \leq j \leq u, z \in Z, z \text{ occurs in the clause } \delta_j \}$. If ψ is a DNF formula, $Z \subseteq \text{Var}(\psi)$ is a subset of variables, and $(\mathcal{T}, (B_t)_{t \in T})$ is a tree decomposition of $\text{IG}(Z, \psi)$, we let $\text{Var}(t)$ denote $B_t \cap Z$, for any $t \in T$.

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:

- for every $v \in V$, the set $B^{-1}(v) = \{ t \in T : v \in B_t \}$ is nonempty and connected in $\mathcal{T}$; and
- for every edge $\{v, w\} \in E$, there is a $t \in T$ such that $v, w \in B_t$.

The *width* of the decomposition $(\mathcal{T}, (B_t)_{t \in T})$ is the number $\max\{ |B_t| : t \in T \} - 1$. The *treewidth* of G is the minimum of the widths of all tree decompositions of G. Let G be a graph and k a nonnegative integer. There is an fpt-algorithm that computes a tree decomposition of G of width k if it exists, and fails otherwise [10]. 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 [37].

The following parameterized decision problems are variants of QSAT$_2$, where the treewidth of the incidence graph graph for certain subsets of variables is bounded.

∃∀-SAT(incid.tw) *Instance:* A quantified Boolean formula $\varphi = \exists X.\forall Y.\psi$, with ψ in DNF. *Parameter:* The treewidth of the incidence graph $\mathrm{IG}(X \cup Y, \psi)$ of ψ with respect to $X \cup Y$. *Question:* Is φ satisfiable?
Complexity: fixed-parameter tractable [15, 24].

∃∀-SAT(∃-incid.tw) *Instance:* A quantified Boolean formula $\varphi = \exists X.\forall Y.\psi$, with ψ in DNF. *Parameter:* The treewidth of the incidence graph $\mathrm{IG}(X, \psi)$ of ψ with respect to X. *Question:* Is φ satisfiable?
Complexity: para-Σ_2^{P}-complete [32, 34].

∃∀-SAT(∀-incid.tw) *Instance:* A quantified Boolean formula $\varphi = \exists X.\forall Y.\psi$, with ψ in DNF. *Parameter:* The treewidth of the incidence graph $\mathrm{IG}(Y, \psi)$ of ψ with respect to Y. *Question:* Is φ satisfiable?
Complexity: para-NP-complete [32, 34].

The above problems are parameterized by the treewidth of the incidence graph of the formula ψ (with respect to different subsets of variables). Since computing the treewidth of a given graph is NP-complete, it is unlikely that the parameter value can be computed in polynomial time for these problems. However, computing the treewidth (and a tree decomposition) of a graph is fixed-parameter tractable in the treewidth [10, 27]. Alternatively, one could consider a variant of the problem where a tree decomposition of width k is given as part of the input.

2.4 Other Quantified Boolean Satisfiability

The following parameterized quantified Boolean satisfiability problem is para-NP-complete.

QBF-SAT(#∀-vars) *Instance:* A quantified Boolean formula φ. *Parameter:* The number of universally quantified variables of φ. *Question:* Is φ true?
Complexity: para-NP-complete [4, 7, 32].

2.5 Minimization for DNF Formulas

Let φ be a propositional formula in DNF. 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. Moreover, we say that a DNF formula φ' is a *term-wise subformula* of φ' if for all terms $t' \in \varphi'$ there exists a term $t \in \varphi$ such that $t' \subseteq t$. The following parameterized problems are natural parameterizations of problems shown to be Σ_2^{P}-complete by Umans [48].

SHORTEST-IMPLICANT-CORE(core size)
Instance: A DNF formula φ, an implicant C of φ, and an integer k. *Parameter:* k. *Question:* Does there exists an implicant $C' \subseteq C$ of φ of size k?
Complexity: $\exists^k\forall^*$-complete [32, 34].

SHORTEST-IMPLICANT-CORE(reduction size)
Instance: A DNF formula φ, an implicant C of φ of size n, and an integer k. *Parameter:* k. *Question:* Does there exists an implicant $C' \subseteq C$ of φ of size $n - k$?
Complexity: $\exists^k\forall^*$-complete [32, 34].

DNF-MINIMIZATION(reduction size)
Instance: A DNF formula φ of size n, and an integer k. *Parameter:* k. *Question:* Does there exist a term-wise subformula φ' of φ of size $n - k$ such that $\varphi \equiv \varphi'$?
Complexity: $\exists^k\forall^*$-complete [32, 34].

DNF-MINIMIZATION(core size)
Instance: A DNF formula φ of size n, and an integer k. *Parameter:* k. *Question:* Does there exist an DNF formula φ' of size k, such that $\varphi \equiv \varphi'$?
Complexity: para-co-NP-hard, in $\mathrm{FPT}^{\mathrm{NP}}[f(k)]$, and in $\exists^k\forall^*$ [32, 34].

2.6 Sequences of Propositional Formulas

The following problem is related to a Boolean combination of satisfiability checks on a sequence of propositional formulas. This is a parameterized version of the problem BH_i-SAT, which is canonical for the different levels of the Boolean Hierarchy (see Section 1). The problem is complete for the class $\mathrm{FPT}^{\mathrm{NP}}[f(k)]$.

BH-SAT(level)
Instance: a positive integer k and a sequence $(\varphi_1, \ldots, \varphi_k)$ of propositional formulas. *Parameter:* k. *Question:* is it the case that $(\varphi_1, \ldots, \varphi_k) \in \mathrm{BH}_k$-SAT?
Complexity: $\mathrm{FPT}^{\mathrm{NP}}[f(k)]$-complete [23].

The above problem is used to show the following lower bound result for $\mathrm{FPT}^{\mathrm{NP}}[f(k)]$-complete problems. No $\mathrm{FPT}^{\mathrm{NP}}[f(k)]$-hard problem can be decided by an fpt-algorithm that uses only $O(1)$ many queries to an NP oracle, unless the Polynomial Hierarchy collapses [23].

3 Knowledge Representation and Reasoning Problems

3.1 Disjunctive Answer Set Programming

The following problems from the setting of disjunctive answer set programming (ASP) are based on the notions of disjunctive logic programs and answer sets for such programs (cf. [11, 38]). A *disjunctive logic program* P is a finite set of rules of the form $r = (a_1 \vee \cdots \vee a_k \leftarrow b_1, \ldots, b_m, \mathit{not}\ c_1, \ldots, \mathit{not}\ c_n)$, 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 *dual-Horn* if $m \leq 1$. A program is called normal if all its rules are normal, it is called negation-free if all its rules are negation-free, and it is called dual-Horn if all its rules are dual-Horn. We let $\mathrm{At}(P)$ denote the set of all atoms occurring in P. By *literals* we mean atoms a or their negations *not* a. 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 *not* a in the body, for each $a \in M$, and (ii) removing literals *not* a from all other rules [28]. An *answer set* A of a program P is a subset-minimal model of the reduct P^A. One important decision problem is to decide, given a disjunctive logic program P, whether P has an answer set. We consider the following parameterizations of this problem.

ASP-CONSISTENCY(#cont.atoms) *Instance:* A disjunctive logic program P. *Parameter:* The number of contingent atoms of P. *Question:* Does P have an answer set?
Complexity: para-co-NP-complete [33, 34].

ASP-CONSISTENCY(#cont.rules) *Instance:* A disjunctive logic program P. *Parameter:* The number of contingent rules of P. *Question:* Does P have an answer set?
Complexity: $\exists^k\forall^*$-complete [33, 34].

ASP-CONSISTENCY(#disj.rules) *Instance:* A disjunctive logic program P. *Parameter:* The number of disjunctive rules of P. *Question:* Does P have an answer set?
Complexity: $\exists^*\forall^k$-W[P]-complete [34].

ASP-CONSISTENCY(#dual-Horn.rules) *Instance:* A disjunctive logic program P. *Parameter:* The number of rules of P that are dual-Horn. *Question:* Does P have an answer set?
Complexity: $\exists^*\forall^k$-W[P]-complete [30].

ASP-CONSISTENCY(str.norm.bd-size) *Instance:* A disjunctive logic program P. *Parameter:* The size of the smallest normality-backdoor for P. *Question:* Does P have an answer set?
Complexity: para-NP-complete [25].

ASP-CONSISTENCY(max.atom.occ.) *Instance:* A disjunctive logic program P. *Parameter:* The maximum number of times that any atom occurs in P. *Question:* Does P have an answer set?
Complexity: para-Σ_2^{P}-complete [33, 34].

3.2 Robust Constraint Satisfaction

The following problem is based on the class of robust constraint satisfaction problems introduced by Gottlob [29] and Abramsky, Gottlob and Kolaitis [1]. 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.

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 $S = \text{Var}(c)$, 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 $c = ((x_1, \dots, x_r), R) \in C$ if $\text{Var}(c) \subseteq X'$ and $(\alpha(x_1), \dots, \alpha(x_r)) \in R$. If α satisfies all constraints of N then it is a *solution* of N. We say that α *violates* a constraint $c = ((x_1, \dots, x_r), R) \in C$ if there is no extension β of α defined on $X' \cup \text{Var}(c)$ such that $(\beta(x_1), \dots, \beta(x_r)) \in R$.

Let k be a positive 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).

Robust-CSP-Sat *Instance:* A CSP instance (X, D, C), and an integer k. *Parameter:* k. *Question:* Is (X, D, C) k-robustly satisfiable?
Complexity: $\forall^k\exists^*$-complete [33, 34].

3.3 Abductive Reasoning

The setting of (propositional) abductive reasoning can be formalized as follows. An *abduction instance* $\mathcal{P}$ consists of a tuple (V, H, M, T), where V is the set of *variables*, $H \subseteq V$ is the set of *hypotheses*, $M \subseteq V$ is the set of *manifestations*, and T is the theory, a formula in CNF over V. It is required that $M \cap H = \emptyset$. A set $S \subseteq H$ is a *solution* (or *explanation*) of $\mathcal{P}$ if (i) $T \cup S$ is consistent and (ii) $T \cup S \models M$. One central problem is to decide, given an abduction instance $\mathcal{P}$ and an integer m, whether there exists a solution S of $\mathcal{P}$ of size at most m. This problem is Σ_2^{P}-complete in general [20].

Abduction(Krom-bd-size): *Input:* an abduction instance $\mathcal{P} = (V, H, M, T)$, and a positive integer m. *Parameter:* The size of the smallest strong 2CNF-backdoor for T. *Question:* Does there exist a solution S of $\mathcal{P}$ of size at most m?
Complexity: para-NP-complete [44].

Abduction(#non-Krom-clauses): *Input:* an abduction instance $\mathcal{P} = (V, H, M, T)$, and a positive integer m. *Parameter:* The number of clauses in T that contains more than 2 literals. *Question:* Does there exist a solution S of $\mathcal{P}$ of size at most m?
Complexity: $\exists^*\forall^k$-W[1]-complete [31].

Abduction(Horn-bd-size): *Input:* an abduction instance $\mathcal{P} = (V, H, M, T)$, and a positive integer m. *Parameter:* The size of the smallest strong Horn-backdoor for T. *Question:* Does there exist a solution S of $\mathcal{P}$ of size at most m?
Complexity: para-NP-complete [44].

Abduction(#non-Horn-clauses):
Input: an abduction instance $\mathcal{P} = (V, H, M, T)$, and a positive integer m. *Parameter:* The number of clauses in T that are not Horn. *Question:* Does there exist a solution S of $\mathcal{P}$ of size at most m?
Complexity: $\exists^*\forall^k$-W[P]-complete [31].

4 Graph Problems

4.1 Clique Extensions

Let $G = (V, E)$ be a graph. A clique $C \subseteq V$ of G is a subset of vertices that induces a complete subgraph of G, i.e. $\{v, v'\} \in E$ for all $v, v' \in C$ such that $v \neq v'$. The W[1]-complete problem of determining whether a graph has a clique of size k is an important problem in the W-hierarchy, and is used in many W[1]-hardness proofs. We consider a related problem that is complete for $\forall^*\exists^k$-W[1].

Small-Clique-Extension
Instance: A graph $G = (V, E)$, a subset $V' \subseteq V$, and an integer k. *Parameter:* k. *Question:* Is it the case that for each clique $C \subseteq V'$, there is some k-clique D of G such that $C \cup D$ is a $(\lvert C \rvert + k)$-clique?
Complexity: $\forall^*\exists^k$-W[1]-complete [34].

4.2 Graph Coloring Extensions

The following problem related to extending colorings to the leaves of a graph to a coloring on the entire graph, is Π_2^{P}-complete in the most general setting [2].

Let $G = (V, E)$ be a graph. We will denote those vertices v that have degree 1 by *leaves*. We call a (partial) function $c : V \to \{1, 2, 3\}$ a 3-*coloring (of G)*. Moreover, we say that a 3-coloring c is *proper* if c assigns a color to every vertex $v \in V$, and if for each edge $e = \{v_1, v_2\} \in E$ holds that $c(v_1) \neq c(v_2)$. The problem of deciding, given a graph $G = (V, E)$ with n many leaves and an integer m, whether any 3-coloring that assigns a color to exactly m leaves of G (and to no other vertices) can be extended to a proper 3-coloring of G, is Π_2^{P}-complete [2]. We consider several parameterizations.

3-Coloring-Extension(degree)
Instance: a graph $G = (V, E)$ with n many leaves, and an integer m. *Parameter:* the degree of G. *Question:* can any 3-coloring that assigns a color to exactly m leaves of G (and to no other vertices) be extended to a proper 3-coloring of G?
Complexity: para-Π_2^{P}-complete [34, 35].

3-Coloring-Extension(#leaves)
Instance: a graph $G = (V, E)$ with n many leaves, and an integer m. *Parameter:* n. *Question:* can any 3-coloring that assigns a color to exactly m leaves of G (and to no other vertices) be extended to a proper 3-coloring of G?
Complexity: para-NP-complete [34, 35].

3-Coloring-Extension(#col.leaves)
Instance: a graph $G = (V, E)$ with n many leaves, and an integer m.
Parameter: m.
Question: can any 3-coloring that assigns a color to exactly m leaves of G (and to no other vertices) be extended to a proper 3-coloring of G?

Complexity: $\forall^k\exists^*$-complete [34, 35].

3-Coloring-Extension(#uncol.leaves)
Instance: a graph $G = (V, E)$ with n many leaves, and an integer m.
Parameter: $n - m$.
Question: can any 3-coloring that assigns a color to exactly m leaves of G (and to no other vertices) be extended to a proper 3-coloring of G?

Complexity: para-Π_2^{P}-complete [34, 35].

5 Other Problems

5.1 First-order Model Checking

First-order model checking is at the basis of a well-known hardness theory in parameterized complexity theory [27]. The following problem, also based on first-order model checking, offers another characterization of the parameterized complexity class $\exists^k\forall^*$. We introduce a few notions that we need for defining the model checking perspective on $\exists^k\forall^*$. A *(relational) vocabulary* τ is a finite set of relation symbols. Each relation symbol R has an *arity* $\mathrm{arity}(R) \geq 1$. A *structure* $\mathcal{A}$ of vocabulary τ, or *τ-structure* (or simply *structure*), consists of a set A called the *domain* and an interpretation $R^{\mathcal{A}} \subseteq A^{\mathrm{arity}(R)}$ for each relation symbol $R \in \tau$. We use the usual definition of truth of a first-order logic sentence φ over the vocubulary τ in a τ-structure $\mathcal{A}$. We let $\mathcal{A} \models \varphi$ denote that the sentence φ is true in structure $\mathcal{A}$. If φ is a first-order formula with free variables $\mathrm{Free}(\varphi)$, and $\mu : \mathrm{Free}(\varphi) \to A$ is an assignment, we use the notation $\mathcal{A}, \mu \models \varphi$ to denote that φ is true in structure $\mathcal{A}$ under the assignment μ.

$\exists^k\forall^*$-MC
Instance: A first-order logic sentence $\varphi = \exists x_1, \dots, x_k.\forall y_1, \dots, y_n.\psi$ over a vocabulary τ, where ψ is quantifier-free, and a finite τ-structure $\mathcal{A}$.
Parameter: k.
Question: Is it the case that $\mathcal{A} \models \varphi$?

Complexity: $\exists^k\forall^*$-complete [34, 35].

5.2 Bounded Model Checking

The following problem is concerned with the problem of verifying whether a linear temporal logic formula is satisfied on all paths in a Kripke structure. This problem is of importance in the area of software and hardware verification [8]. *Linear temporal logic* (LTL) is a modal temporal logic where one can encode properties related to the future of paths. *LTL formulas* are defined recursively as follows: propositional variables and their negations are in LTL; then, if $\varphi_1, \varphi_2 \in \mathrm{LTL}$, then so are $\varphi_1 \vee \varphi_2$, $\mathrm{F}\varphi_1$ (Future), $\mathrm{X}\varphi_1$ (neXt), $\varphi_1 \mathrm{U} \varphi_2$ (φ_1 Until φ_2). (Further temporal operators that are considered in the literature can be defined in terms of the operators X and U.)

The semantics of LTL is defined along paths of Kripke structures. A *Kripke structure* is a tuple $K = (S, I, T, L)$ such that (i) S is a set of states, where states are defined by valuations to a set V of propositional variables, (ii) $I \subseteq S$ is a nonempty set of initial states, (iii) $T \subseteq S \times S$ is the transition relation and (iv) $L : S \to 2^V$ is the labeling function. The initial states I and the transition relation T are given as functions in terms of S. A path π of K is an infinite sequence $(s_0, s_1, s_2, \dots)$ of states, where $s_i \in S$ and $T(s_i, s_{i+1})$ for all $i \in \mathbb{N}$. A path is initialized if $s_0 \in I$. We let $\pi(i) = s_i$ denote the i-th state of π. A suffix of a path is defined as $\pi^i = (s_i, s_{i+1}, \dots)$. We give the standard semantics of LTL formulas, defined recursively over the formula structure. We closely follow the definitions as given by Biere [8]. In what cases an LTL formula φ holds along a path π^i, written $\pi^i \models \varphi$, is specified by the following conditions:

$$
\begin{array}{lll@{\qquad}lll}
\pi^i \models v \in V & \text{iff} & v \in L(\pi(i)), & \pi^i \models \neg v & \text{iff} & v \notin L(\pi(i)), \\
\pi^i \models \varphi_1 \vee \varphi_2 & \text{iff} & \pi^i \models \varphi_1 \text{ or } \pi^i \models \varphi_2, & \pi^i \models \mathrm{X}\varphi & \text{iff} & \pi^{i+1} \models \varphi, \\
\pi^i \models \mathrm{F}\varphi & \text{iff} & \text{for some } j \in \mathbb{N},\ \pi^{i+j} \models \varphi, & \pi^i \models \varphi_1 \mathrm{U} \varphi_2 & \text{iff} & \text{for some } j \in \mathbb{N},\ \pi^{i+j} \models \varphi_2 \text{ and} \\
 & & & & & \pi^\ell \models \varphi_1 \text{ for all } i \leq \ell < i+j.
\end{array}
$$

Then, an LTL formula φ holds in a Kripke structure K if and only if $\pi \models \varphi$ for all initialized paths π of K. Related to the model checking problem is the question whether a witness exists: a formula φ has a witness in K if there is an initialized path π of K with $\pi \models \varphi$.

The idea of *bounded model checking* is to consider only those paths that can be represented by a prefix of length at most k, and prefixes of length k. Observe that some infinite paths can be represented by a finite prefix with a "loop": an infinite path is a (k, l)-lasso if $\pi(k + 1 + j) = \pi(l + j)$, for all $j \in \mathbb{N}$. In fact, the search for witnesses can be restricted to lassos if K is finite. This leads to the following bounded semantics. In what cases an LTL formula φ holds along a suffix π^i of a (k, l)-lasso π in the bounded semantics, written $\pi^i \models_k \varphi$, is specified by the following conditions:

$$
\begin{array}{lll}
\pi^i \models_k \mathrm{X}\varphi & \text{iff} & \begin{cases} \pi^{i+1} \models_k \varphi & \text{if } i < k, \\ \pi^l \models_k \varphi & \text{if } i = k, \end{cases} \\
\pi^i \models_k \mathrm{F}\varphi & \text{iff} & \text{for some } j \in \{\min(i,l), \dots, k\},\ \pi^j \models_k \varphi, \\
\pi^i \models_k \varphi_1 \mathrm{U} \varphi_2 & \text{iff} & \text{for some } j \in \{\min(i,l), \dots, k\},\ \pi^j \models_k \varphi_2, \\
 & & \text{and } \begin{cases} \pi^\ell \models_k \varphi_1 \text{ for all } i \leq \ell < k \text{ and all } l \leq \ell < j & \text{if } j < i, \\ \pi^\ell \models_k \varphi_1 \text{ for all } l \leq \ell < j & \text{if } j \geq i. \end{cases}
\end{array}
$$

In the case where π is not a (k, l)-lasso for any l, the bounded semantics only gives an approximation. In what cases an LTL formula φ holds along a suffix π^i of a path π that is not a (k, l)-lasso for any l, written $\pi^i \models_k \varphi$, is specified by the following conditions:

$$
\begin{array}{lll}
\pi^i \models_k \mathrm{X}\varphi & \text{iff} & \pi^{i+1} \models_k \varphi \text{ and } i < k, \\
\pi^i \models_k \mathrm{F}\varphi & \text{iff} & \text{for some } j \in \{i, \dots, k\},\ \pi^j \models_k \varphi, \\
\pi^i \models_k \varphi_1 \mathrm{U} \varphi_2 & \text{iff} & \text{for some } j \in \{i, \dots, k\},\ \pi^j \models_k \varphi_2, \\
 & & \text{and } \pi^\ell \models_k \varphi_1 \text{ for all } i \leq \ell < j.
\end{array}
$$

Note that $\pi \models_k \varphi$ implies $\pi \models \varphi$ for all paths π. However, it might be the case that $\pi \models \varphi$ but not $\pi \models_k \varphi$.

For a detailed definition and discussion of Kripke structures and the syntax and semantics of LTL we refer to other sources [5, 16]. For a detailed definition of the bounded semantics for LTL formulas, we refer to the bounded model checking literature [8, 9]. The following problem, that we consider as a parameterized problem, is central to bounded model checking.

BMC-Witness
Instance: An LTL formula φ, a Kripke structure K, and an integer $k \geq 1$. *Parameter:* k. *Question:* Is there some path π of K such that $\pi \models_k \varphi$?
Complexity: in para-NP [9].

The unparameterized variant of this problem is PSPACE-complete, when the integer k is given in binary [5, Theorem 5.46 and Lemma 5.47]. However, if the integer k is given in unary, the unparameterized variant of this problem is NP-complete [5, 9].

5.3 Quantified Fagin Definability

The W-hierarchy can also be defined by means of Fagin-definable parameterized problems [27], which are based on Fagin's characterization of NP. We provide an additional characterization of the class $\forall^k\exists^*$ by means of some parameterized problems that are quantified analogues of Fagin-defined problems.

Let τ be an arbitrary vocabulary, and let $\tau' \subseteq \tau$ be a subvocabulary of τ. We say that a τ-structure $\mathcal{A}$ *extends* a τ'-structure $\mathcal{B}$ if (i) $\mathcal{A}$ and $\mathcal{B}$ have the same domain, and (ii) $\mathcal{A}$ and $\mathcal{B}$ coincide on the interpretation of all relational symbols in τ', i.e. $R^{\mathcal{A}} = R^{\mathcal{B}}$ for all $R \in \tau'$. We say that $\mathcal{A}$ *extends* $\mathcal{B}$ *with weight* k if $\sum_{R \in \tau \setminus \tau'} |R^{\mathcal{A}}| = k$. Let φ be a first-order formula over τ with a free relation variable X of arity s.

We let Π_2 denote the class of all first-order formulas of the form $\forall y_1, \dots, y_n.\exists x_1, \dots, x_m.\psi$, where ψ is quantifier-free. Let $\varphi(X)$ be a first-order formula over τ, with a free relation variable X with arity s. Consider the following parameterized problem.

$\forall^k\exists^*$-FD$_{\varphi}^{(\tau,\tau')}$ *Instance:* A τ'-structure $\mathcal{B}$, and an integer k. *Parameter:* k. *Question:* Is it the case that for each τ-structure $\mathcal{A}$ extending $\mathcal{B}$ with weight k, there exists some relation $S \subseteq A^s$ such that $\mathcal{A} \models \varphi(S)$?
Complexity: in $\forall^k\exists^*$ for each $\varphi(X)$, τ' and τ; $\forall^k\exists^*$-hard for some $\varphi(X) \in \Pi_2$, τ' and τ [34].

Note that this means the following. We let T denote the set of all relational vocabularies, and for any $\tau \in T$ we let FO_{τ}^{X} denote the set of all first-order formulas over the vocabulary τ with a free relation variable X. We then get the following characterization of $\forall^k\exists^*$ [34]:

$$\forall^k\exists^* = [\ \{\ \forall^k\exists^*\text{-FD}_{\varphi}^{(\tau',\tau)} : \tau \in T, \tau' \subseteq \tau, \varphi \in \mathrm{FO}_{\tau}^{X}\ \}\ ]^{\mathrm{fpt}}.$$

Additionally, the following parameterized problem is hard for $\forall^*\exists^k$-W[1].

$\forall^*\exists^k$-FD$_{\varphi}^{(\tau,\tau')}$ *Instance:* A τ'-structure $\mathcal{B}$, and an integer k. *Parameter:* k. *Question:* Is it the case that for each τ-structure $\mathcal{A}$ extending $\mathcal{B}$, there exists some relation $S \subseteq A^s$ with $|S| = k$ such that $\mathcal{A} \models \varphi(S)$?
Complexity: $\forall^*\exists^k$-W[1]-hard for some $\varphi(X) \in \Pi_2$ [34].

5.4 Computational Social Choice

The following problems are related to judgment aggregation, in the domain of computational social choice. Judgment aggregation studies procedures that combine individuals' opinions into a collective group opinion.

An *agenda* is a finite nonempty set $\Phi = \{\varphi_1, \dots, \varphi_n, \neg\varphi_1, \dots, \neg\varphi_n\}$ of formulas that is closed under complementation. A *judgment set* J for an agenda Φ is a subset $J \subseteq \Phi$. We call a judgment set J *complete* if $\varphi_i \in J$ or $\neg\varphi_i \in J$ for all formulas φ_i, and we call it *consistent* if there exists an assignment that makes all formulas in J true. Let $\mathcal{J}(\Phi)$ denote the set of all complete and consistent subsets of Φ. We call a sequence $\boldsymbol{J} \in \mathcal{J}(\Phi)^n$ of complete and consistent subsets a *profile*. A (resolute) *judgment aggregation procedure* for the agenda Φ and n individuals is a function $F : \mathcal{J}(\Phi)^n \to \mathcal{P}(\Phi \backslash \emptyset) \backslash \emptyset$ that returns for each profile $\boldsymbol{J}$ a non-empty set $F(\boldsymbol{J})$ of non-empty judgment sets. An example is the *majority rule* F^{maj}, where $F^{\mathrm{maj}}(\boldsymbol{J}) = \{J^*\}$ and where $\varphi \in J^*$ if and only if φ occurs in the majority of judgment sets in $\boldsymbol{J}$, for each $\varphi \in \Phi$. We call F *complete* and *consistent*, if each $J^* \in F(\boldsymbol{J})$ is complete and consistent, respectively, for every $\boldsymbol{J} \in \mathcal{J}(\Phi)^n$. For instance, the majority rule F^{maj} is complete, whenever the number n of individuals is odd. An agenda Φ is *safe* with respect to an aggregation procedure F, if F is consistent when applied to profiles of judgment sets over Φ. We say that an agenda Φ satisfies the *median property (MP)* if every inconsistent subset of Φ has itself an inconsistent subset of size at most 2. Safety for the majority rule can be characterized in terms of the median property as follows: an agenda Φ is safe for the majority rule if and only if Φ satisfies the MP [21, 41]. The problem of deciding whether an agenda satisfies the MP is Π_2^{P}-complete [21].

Maj-Agenda-Safety(formula size) *Instance:* An agenda Φ. *Parameter:* $\ell = \max\{\, \lvert\varphi\rvert : \varphi \in \Phi \,\}$. *Question:* Is Φ safe for the majority rule?
Complexity: para-Π_2^{P}-complete [22, 23].

Maj-Agenda-Safety(degree) *Instance:* An agenda Φ containing only CNF formulas. *Parameter:* The degree d of Φ. *Question:* Is Φ safe for the majority rule?
Complexity: para-Π_2^{P}-complete [22, 23].

Maj-Agenda-Safety(degree + formula size) *Instance:* An agenda Φ containing only CNF formulas, where $\ell = \max\{\, \lvert\varphi\rvert : \varphi \in B(\Phi) \,\}$, and where d is the degree of Φ. *Parameter:* $\ell + d$. *Question:* Is Φ safe for the majority rule?
Complexity: para-Π_2^{P}-complete [22, 23].

The above three parameterized problems remain para-Π_2^{P}-hard even when restricted to agendas based on formulas that are Horn formulas containing only clauses of size at most 2 [22, 23].

Maj-Agenda-Safety(agenda size) *Instance:* An agenda Φ. *Parameter:* $\lvert\Phi\rvert$. *Question:* Is Φ safe for the majority rule?
Complexity: $\mathrm{FPT}^{\mathrm{NP}}[f(k)]$-complete [22, 23].

Moreover, the following upper and lower bounds on the number of oracle queries are known for the above problem. Maj-Agenda-Safety(agenda size) can be decided in fixed-parameter tractable time using $2^{O(k)}$ queries to an NP oracle, where $k = \lvert\Phi\rvert$ [23]. In addition, there is no fpt-algorithm that decides Maj-Agenda-Safety(agenda size) using $o(\log k)$ queries to an NP oracle, unless the Polynomial Hierarchy collapses [22, 23].

Maj-Agenda-Safety(counterexample size) *Instance:* An agenda Φ, and an integer k. *Parameter:* k. *Question:* Does every inconsistent subset Φ' of Φ of size k have itself an inconsistent subset of size at most 2?
Complexity: $\forall^k\exists^*$-hard [22, 23].

Let $\Phi = \{\varphi_1, \dots, \varphi_m, \neg\varphi_1, \dots, \neg\varphi_m\}$ be an agenda, where each φ_i is a CNF formula. We define the following graphs that are intended to capture the interaction between formulas in Φ. The *formula primal graph* of Φ has as vertices the variables $\mathrm{Var}(\Phi)$ occurring in the agenda, and two variables are connected by an edge if there exists a formula φ_i in which they both occur. The *formula incidence graph* of Φ is a bipartite graph whose vertices consist of (1) the variables $\mathrm{Var}(\Phi)$ occurring in the agenda and (2) the formulas $\varphi_i \in \Phi$. A variable $x \in \mathrm{Var}(\Phi)$ is connected by an edge with a formula $\varphi_i \in \Phi$ if x occurs in φ_i, i.e., $x \in \mathrm{Var}(\varphi_i)$. The *clausal primal graph* of Φ has as vertices the variables $\mathrm{Var}(\Phi)$ occurring in the agenda, and two variables are connected by an edge if there exists a formula φ_i and a clause $c \in \varphi_i$ in which they both occur. The *clausal incidence graph* of Φ is a bipartite graph whose vertices consist of (1) the variables $\mathrm{Var}(\Phi)$ occurring in the agenda and (2) the clauses c occurring in formulas $\varphi_i \in \Phi$.

A variable $x \in \mathrm{Var}(\Phi)$ is connected by an edge with a clause c of the formula $\varphi_i \in \Phi$ if x occurs in c, i.e., $x \in \mathrm{Var}(c)$. Consider the following parameterizations of the problem Maj-Agenda-Safety.

Maj-Agenda-Safety(f-tw) *Instance:* An agenda Φ containing only CNF formulas. *Parameter:* The treewidth of the formula primal graph of Φ. *Question:* Is Φ safe for the majority rule?
Complexity: fixed-parameter tractable [22].

Maj-Agenda-Safety(f-tw*) *Instance:* An agenda Φ containing only CNF formulas. *Parameter:* The treewidth of the formula incidence graph of Φ. *Question:* Is Φ safe for the majority rule?
Complexity: para-Π_2^{P}-complete [22].

Maj-Agenda-Safety(c-tw) *Instance:* An agenda Φ containing only CNF formulas. *Parameter:* The treewidth of the clausal primal graph of Φ. *Question:* Is Φ safe for the majority rule?
Complexity: para-co-NP-complete [22].

Maj-Agenda-Safety(c-tw*) *Instance:* An agenda Φ containing only CNF formulas. *Parameter:* The treewidth of the clausal incidence graph of Φ. *Question:* Is Φ safe for the majority rule?
Complexity: para-co-NP-complete [22].

5.5 Turing Machine Halting

The following problems are related to alternating Turing machines (ATMs), possibly with multiple tapes. ATMs are nondeterministic Turing machines where the states are divided into existential and universal states, and where each configuration of the machine is called existential or universal according to the state that the machine is in. A run ρ of the ATM $\mathbb{M}$ on an input x is a tree whose nodes correspond to configurations of $\mathbb{M}$ in such a way that (1) for each non-root node v of the tree with parent node v', the machine $\mathbb{M}$ can transition from the configuration corresponding to v' to the configuration corresponding to v, (2) the root node corresponds to the initial configuration of $\mathbb{M}$, and (3) each child node corresponds to a halting configuration. A computation path in a run of $\mathbb{M}$ is a root-to-leaf path in the run. Moreover, the nodes of a run ρ are labelled accepting or rejecting, according to the following definition. A leaf of ρ is labelled accepting if the configuration corresponding to it is an accepting configuration, and the leaf is labelled rejecting if it is a rejecting configuration. A non-leaf node of ρ that corresponds to an existential configuration is labelled accepting if at least one of its children is labelled accepting. A non-leaf node of ρ that corresponds to a universal configuration is labelled accepting if all of its children is labelled accepting. An ATM $\mathbb{M}$ is 2-alternating if for each input x, each computation path in the run of $\mathbb{M}$ on input x switches at most once from an existential configuration to a universal configuration, or vice versa. For more details on the terminology, we refer to the work of De Haan and Szeider [34, 35] and to the work of Flum and Grohe [27, Appendix A.1].

We consider the following restrictions on ATMs. An $\exists\forall$-*Turing machine* (or simply $\exists\forall$-machine) is a 2-alternating ATM, where the initial state is an existential state. Let $\ell, t \geq 1$ be positive integers. We say that an $\exists\forall$-machine $\mathbb{M}$ *halts (on the empty string) with existential cost ℓ and universal cost t* if: (1) there is an accepting run of $\mathbb{M}$ with the empty input ϵ, and (2) each computation path of $\mathbb{M}$ contains at most ℓ existential configurations and at most t universal configurations. The following problem, where the number of Turing machine tapes is given as part of the input, is $\exists^k\forall^*$-complete.

$\exists^k\forall^*$-TM-HALT*.
Instance: Positive integers $m, k, t \geq 1$, and a $\exists\forall$-machine $\mathbb{M}$ with m tapes. *Parameter:* k. *Question:* Does $\mathbb{M}$ halt on the empty string with existential cost k and universal cost t?
Complexity: $\exists^k\forall^*$-complete [34, 35].

Let $m \geq 1$ be a constant integer. Then the following parameterized decision problem, where the number of Turing machine tapes is fixed, is also $\exists^k\forall^*$-complete.

$\exists^k\forall^*$-TM-HALT^m.
Instance: Positive integers $k, t \geq 1$, and an $\exists\forall$-machine $\mathbb{M}$ with m tapes. *Parameter:* k. *Question:* Does $\mathbb{M}$ halt on the empty string with existential cost k and universal cost t?
Complexity: $\exists^k\forall^*$-complete [34, 35].

In addition, the parameterized complexity class $\exists^k\forall^*$ can also be characterized by means of alternating Turing machines in the following way. Let P be a parameterized problem. An $\exists^k\forall^*$*-machine for* P is a $\exists\forall$-machine $\mathbb{M}$ such that there exists a computable function f and a polynomial p such that: (1) $\mathbb{M}$ decides P in time $f(k) \cdot p(|x|)$; and (2) for all instances (x, k) of P and each computation path R of $\mathbb{M}$ with input (x, k), at most $f(k) \cdot \log |x|$ of the existential configurations of R are nondeterministic. We say that a parameterized problem P *is decided by some* $\exists^k\forall^*$*-machine* if there exists a $\exists^k\forall^*$-machine for P. Then, $\exists^k\forall^*$ is exactly the class of parameterized decision problems that are decided by some $\exists^k\forall^*$-machine [34, 35].

References

[1] Samson Abramsky, Georg Gottlob, and Phokion G. Kolaitis. Robust constraint satisfaction and local hidden variables in quantum mechanics. In Francesca Rossi, editor, *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI 2013*. AAAI Press/IJCAI, 2013.

[2] Miklós Ajtai, Ronald Fagin, and Larry J. Stockmeyer. The closure of monadic NP. *J. of Computer and System Sciences*, 60(3):660–716, 2000.

[3] Sanjeev Arora and Boaz Barak. *Computational Complexity – A Modern Approach*. Cambridge University Press, 2009.

[4] 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, 2002.

[5] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

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

[7] 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.

[8] 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.

[9] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In Rance Cleaveland, editor, *Tools and Algorithms for Construction and Analysis of Systems, 5th International Conference, TACAS '99, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'99, Amsterdam, The Netherlands, March 22-28, 1999, Proceedings*, volume 1579 of *Lecture Notes in Computer Science*, pages 193–207. Springer Verlag, 1999.

[10] Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 25(6):1305–1317, 1996.

[11] Gerhard Brewka, Thomas Eiter, and Miroslaw Truszczynski. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2011.

[12] Jin-yi Cai, Thomas Gundermann, Juris Hartmanis, Lane A. Hemachandra, Vivian Sewelson, Klaus W. Wagner, and Gerd Wechsung. The Boolean hierarchy I: Structural properties. *SIAM J. Comput.*, 17(6):1232–1252, 1988.

[13] Marco Cesati. Compendium of parameterized problems. `http://bravo.ce.uniroma2.it/home/cesati/research/compendium.pdf`, September 2006.

[14] Richard Chang and Jim Kadin. The Boolean hierarchy and the polynomial hierarchy: a closer connection. *SIAM J. Comput.*, 25:169–178, 1993.

[15] Hubie Chen. Quantified constraint satisfaction and bounded treewidth. In *Proceedings of ECAI 2004, the 16th European Conference on Artificial Intelligence*, pages 161–165. IOS Press, 2004.

[16] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT Press, Cambridge, MA, USA, 1999.

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

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

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

[20] Thomas Eiter and Georg Gottlob. The complexity of logic-based abduction. *J. of the ACM*, 42(1):3–42, 1995.

[21] Ulle Endriss, Umberto Grandi, and Daniele Porello. Complexity of judgment aggregation. *J. Artif. Intell. Res.*, 45:481–514, 2012.

[22] Ulle Endriss, Ronald de Haan, and Stefan Szeider. Parameterized complexity results for agenda safety in judgment aggregation. Unpublished manuscript, 2014.

[23] Ulle Endriss, Ronald de Haan, and Stefan Szeider. Parameterized complexity results for agenda safety in judgment aggregation. In *Proceedings of the 5th International Workshop on Computational Social Choice (COMSOC-2014)*. Carnegie Mellon University, June 2014.

[24] Tomás Feder and Phokion G. Kolaitis. Closures and dichotomies for quantified constraints. Electronic Colloquium on Computational Complexity *(ECCC)*, 13(160), 2006.

[25] 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.

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

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

[28] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4):365–386, 1991.

[29] Georg Gottlob. On minimal constraint networks. *Artificial Intelligence*, 191–192:42–60, 2012.

[30] Ronald de Haan. Unpublished manuscript, 2014.

[31] Ronald de Haan, Andreas Pfandler, Stefan Rümmele, and Stefan Szeider. Backdoors to abduction. Unpublished manuscript, 2014.

[32] Ronald de Haan and Stefan Szeider. Fixed-parameter tractable reductions to SAT. In Uwe Egly and Carsten Sinz, editors, *Proceedings of the 17th International Symposium on the Theory and Applications of Satisfiability Testing (SAT 2014) Vienna, Austria, July 14–17, 2014*, volume 8561 of *Lecture Notes in Computer Science*, pages 85–102. Springer, 2014.

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

[34] Ronald de Haan and Stefan Szeider. The parameterized complexity of reasoning problems beyond NP. Technical Report 1312.1672v3, arXiv.org, 2014.

[35] Ronald de Haan and Stefan Szeider. Machine characterizations for parameterized complexity classes beyond para-NP. In *Proceedings of the 41st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2015)*, Lecture Notes in Computer Science. Springer Verlag, 2015. To appear.

[36] Jim Kadin. The polynomial time hierarchy collapses if the Boolean hierarchy collapses. *SIAM J. Comput.*, 17(6):1263–1282, December 1988.

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

[38] V. Wiktor Marek and Miroslaw Truszczynski. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-Year Perspective*, pages 169–181. Springer, 1999.

[39] 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, 2013.

[40] Albert R. Meyer and Larry J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *SWAT*, pages 125–129. IEEE Computer Soc., 1972.

[41] Klaus Nehring and Clemens Puppe. The structure of strategy-proof social choice - part I: General characterization and possibility results on median spaces. *J. of Economic Theory*, 135(1):269–305, 2007.

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

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

[44] 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.

[45] 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.

[46] Marcus Schaefer and Christopher Umans. Completeness in the polynomial-time hierarchy: A compendium. *SIGACT News*, 33(3):32–49, September 2002.

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

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

[49] Moshe Y. Vardi. Boolean satisfiability: theory and engineering. *Communications of the ACM*, 57(3):5, March 2014.

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

Index of Problems by Complexity

ECCC ISSN 1433-8092

http://eccc.hpi-web.de