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

1 Preliminaries 2

2 Propositional Logic Problems 5

2.1 Weighted Quantified Boolean Satisfiability in the ++-k Hierarchy 7
2.2 Weighted Quantified Boolean Satisfiability for the k-\ast Classes 7
2.3 Quantified Boolean Satisfiability with Bounded Treewidth 8
2.4 Other Quantified Boolean Satisfiability 9
2.5 Minimization for DNF Formulas 9
2.6 Sequences of Propositional Formulas 10

3 Knowledge Representation and Reasoning Problems 10

3.1 Disjunctive Answer Set Programming 10
3.2 Robust Constraint Satisfaction 11
3.3 Abductive Reasoning 12

4 Graph Problems 13

4.1 Clique Extensions 13
4.2 Graph Coloring Extensions 13

5 Other Problems 14

5.1 First-order Model Checking 14
5.2 Bounded Model Checking 14
5.3 Quantified Fagin Definability 15
5.4 Computational Social Choice 16
5.5 Turing Machine Halting 18

Index of Problems by Complexity 23

*Supported by the European Research Council (ERC), project 239962 (COMPLEX REASON), and the Austrian Science Fund (FWF), project P26200 (Parameterized Compilation).
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) \( \text{fpt-} \) 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 \( \text{fpt-} \) 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 [10] [35] [47] [50] contains a hierarchy of increasing complexity classes \( \Sigma^P_i \), 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 \( \bigwedge_{i=1}^m X_i \), where each \( X_i \) is either \( \forall \) or \( \exists \), the \( X_i \) are disjoint sets of propositional variables, and \( \psi \) 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 \( \gamma = \{ x_1 \mapsto d_1, \ldots, x_n \mapsto d_n \} \) be a function that maps some variables of a formula \( \varphi \) to other variables or to truth values. We let \( \varphi[\gamma] \) denote the application of such a substitution \( \gamma \) to the formula \( \varphi \). We also write \( \varphi[x_1 \mapsto d_1, \ldots, x_n \mapsto d_n] \) to denote \( \varphi[\gamma] \). For each \( i \geq 1 \) we define the following decision problem.

**QSat**

**Instance:** A quantified Boolean formula \( \varphi = \exists X_1 \forall X_2 \exists X_3 \ldots Q_i X_i \psi \), where \( Q_i \) is a universal quantifier if \( i \) is even and an existential quantifier if \( i \) is odd.

**Question:** Is \( \varphi \) true?

Input formulas to the problem QSat are called \( \Sigma^P_i \)-formulas. For each nonnegative integer \( i \leq 0 \), the complexity class \( \Sigma^P_i \) can be characterized as the closure of the problem QSat under polynomial-time reductions [47] [50]. The \( \Sigma^P_i \)-hardness of QSat, 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 \( \Sigma^P_1 \) coincides with P, and the class \( \Sigma^P_1 \) coincides with NP. For each \( i \geq 1 \), the class \( \Pi^P_i \) is defined as \( \text{co-} \Sigma^P_{i+1} \).

The classes \( \Sigma^P_i \) and \( \Pi^P_i \) 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 \( \Sigma^P_i \) and \( \Pi^P_i \), for \( i \geq 0 \), can be equivalently defined by letting \( \Sigma^P_0 = \Pi^P_0 = P \), and for each \( i \geq 1 \) letting \( \Sigma^P_i = NP^{\Sigma^P_{i-1}} \) and \( \Pi^P_i = \text{co-NP}^{\Sigma^P_{i-1}} \).

The Polynomial Hierarchy also includes complexity classes between \( \Sigma^P_i \) and \( \Pi^P_i \), on the one hand, and \( \Sigma^P_{i+1} \) and \( \Pi^P_{i+1} \), on the other hand. The class \( \Delta^P_{i+1} \) 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 \( \Sigma^P_i \). Similarly, the class \( \Theta^P_{i+1} \) 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 \( \Sigma^P_i \), 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 \( \Theta^P_2 \) on the other hand. The Boolean Hierarchy (BH) [12] [13] [56] 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 $\Theta^P_2$. Each class $BH_i$ can be characterized as the class of problems that can be reduced to the problem $BH_{i-1}$-SAT, which is defined inductively as follows. The problem $BH_1$-SAT consists of all sequences $(\varphi)$ of length 1, where $\varphi$ is a satisfiable propositional formula. For even $i \geq 2$, the problem $BH_i$-SAT consists of all sequences $(\varphi_1, \ldots, \varphi_i)$ of propositional formulas such that both $(\varphi_1, \ldots, \varphi_{i-1}) \in BH_{(i-1)}$-SAT and $\varphi_i$ is unsatisfiable. For odd $i \geq 2$, the problem $BH_i$-SAT consists of all sequences $(\varphi_1, \ldots, \varphi_i)$ of propositional formulas such that $(\varphi_1, \ldots, \varphi_{i-1}) \in BH_{(i-1)}$-SAT or $\varphi_i$ 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 $NP \cup co-NP \subseteq BH_2 \subseteq BH_3 \subseteq \cdots \subseteq BH \subseteq \Theta^P_2$.

**Parameterized Complexity** We introduce some core concepts 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 $\Sigma$. 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 called **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 $O(|I|^{|I|^{f(k)}})$, 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 $R : \Sigma^* \times \mathbb{N} \rightarrow (\Sigma')^* \times \mathbb{N}$ from instances of $L$ to instances of $L'$ such that there exist some computable function $g : \mathbb{N} \rightarrow \mathbb{N}$ such that for all $(I, k) \in \Sigma^* \times \mathbb{N}$: (i) $(I, k)$ is a yes-instance of $L$ if and only if $(R(I, k)) = (I', k') \in L'$, (ii) $k' \leq g(k)$, and (iii) $R$ is computable in fpt-time. Similarly, we call reductions that satisfy properties (i) and (iii) but that are computable in time $O(|I|^{|I|^{f(k)}})$, 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 $\leq 2$, 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 width 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 $\leq 1$. 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 weight $t$ by $\text{CIRCU}_t$. We denote the class of all Boolean circuits by $\text{CIRCU}$, and the class of all Boolean formulas by $\text{FORM}$. For any class $\mathcal{C}$ of Boolean circuits, we define the following parameterized problem.

\[ p\text{-WSAT}[^t][\mathcal{C}] \]

**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 $W[t] = [\{ p\text{-WSAT}[^t][\text{CIRCU}_u] : u \geq 1 \}]^\text{fpt}$ for all $t \geq 1$. The classes $W[SAT]$ and $W[P]$ are defined by letting $W[SAT] = [p\text{-WSAT}[^t][\text{FORM}]^\text{fpt}$ and $W[P] = [p\text{-WSAT}[^t][\text{CIRCU}]^\text{fpt}$.

Let $K$ be a classical complexity class, e.g. $NP$. The parameterized complexity class $\text{para-K}$ is defined as the class of all parameterized problems $L \subseteq \Sigma^* \times \mathbb{N}$ for some finite alphabet $\Sigma$, for which there exist an alphabet $\Pi$, a computable function $f : \mathbb{N} \rightarrow \Pi^*$, and a problem $P \subseteq \Pi^* \times \Pi^*$ 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 [20]. 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 $\text{XK}^{\text{nu}}$, 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 $\text{XP}^{\text{nu}}$ 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 $\text{XK}$ of these classes $\text{XK}^{\text{nu}}$. 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] [20] [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. [15]). 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-$\Sigma_2^P$-hard and para-$\Pi_2^P$-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-$\Sigma_2^P$-hard nor para-$\Pi_2^P$-hard either. Recently, several complexity classes have been introduced to classify such intermediate problems [23] [34]. These parameterized complexity classes are dubbed the $k$-class and the $*$-class 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 $\exists^k \forall^* \text{-WSAT}(\mathcal{C})$ provides the foundation for the $k$-class.

\begin{center}
$\exists^k \forall^* \text{-WSAT}$
\begin{itemize}
  \item **Instance:** A quantified Boolean formula $\exists X. \forall Y. \psi$, and an integer $k$.
  \item **Parameter:** $k$.
  \item **Question:** Does there exist a truth assignment $\alpha$ to $X$ with weight $k$ such that for all truth assignments $\beta$ to $Y$ the assignment $\alpha \cup \beta$ satisfies $\psi$?
\end{itemize}
\end{center}

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

\begin{center}
$\exists^* \forall^k \text{-WSAT}(\mathcal{C})$
\begin{itemize}
  \item **Instance:** A Boolean circuit $C \in \mathcal{C}$ over two disjoint sets $X$ and $Y$ of variables, and an integer $k$.
  \item **Parameter:** $k$.
  \item **Question:** Does there exist a truth assignment $\alpha$ to $X$ such that for all truth assignments $\beta$ to $Y$ with weight $k$ the assignment $\alpha \cup \beta$ satisfies $C$?
\end{itemize}
\end{center}
We present only a two canonical forms of the problems in the $k^k$ hierarchy: $\exists^k \forall^*$ and $\forall^k \exists^k$, respectively. These problems are based on the $k$-SAT problem, which is known to be complete for the complexity class $\Sigma_2^p$. Similarly, the classes of the $\exists^k$ hierarchy are defined as follows:

$$\exists^k \forall^* = \{ \exists^k \forall^* - \text{SAT} \}^{\text{fpt}}.$$

Similarly, the classes of the $\forall^k$ hierarchy are defined as follows:

$$\forall^k \exists^k = \{ \forall^k \exists^k - \text{SAT} \}^{\text{fpt}},$$

$$\exists^k \forall^k = \{ \exists^k \forall^k - \text{SAT} \}^{\text{fpt}}.$$.

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

$$\exists^k \forall^k - \text{SAT} \subseteq \exists^k \forall^k - \text{SAT} \subseteq \exists^k \forall^k - \text{SAT},$$

$$\forall^k \exists^k - \text{SAT} \subseteq \forall^k \exists^k - \text{SAT} \subseteq \forall^k \exists^k - \text{SAT}.$$.

Dual to the classical complexity class $\Sigma_2^p$, is the co-class $\Pi_2^p$, whose canonical complete problem is the problem $Q \text{SAT}$. Similarly, we can define dual classes for the $k$-$\forall$ class and for each of the parameterized complexity classes in the $k^k$ hierarchy. Such co-classes are based on problems complementary to the problems $\exists^k \forall^* - \text{SAT}$ and $\forall^k \exists^k - \text{SAT}$, i.e., these problems have as yes-instances exactly the no-instances of $\exists^k \forall^* - \text{SAT}$ and $\forall^k \exists^k - \text{SAT}$, respectively. Equivalently, these complementary problems can be considered as variants of $\exists^k \forall^* - \text{SAT}$ and $\forall^k \exists^k - \text{SAT}$ where the existential and universal quantifiers are swapped, and are therefore denoted with $\forall^* \exists^k - \text{SAT}$ and $\exists^k \forall^* - \text{SAT}$. We use a similar notation for the dual complexity classes, e.g., we denote $\text{co-} \exists^k \forall^* - \text{SAT}$ by $\forall^* \exists^k - \text{SAT}$.

The class $\exists^k \forall^*$ includes the class $\text{para-co-} \text{NP}$ as a subset, and is contained in the class $\text{Xco-NP}$ as a subset. Similarly, each of the classes $\exists^k \forall^k - \text{SAT}$ and $\forall^k \exists^k - \text{SAT}$ is the parameter value, 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 $k^k$ hierarchies are based. We present only a two canonical forms of the problems in the $k^k$ hierarchy. For problems in the $k^k$-hierarchy, we let $C$ range over classes of Boolean circuits.

<table>
<thead>
<tr>
<th>$\exists^k \forall^* - \text{SAT}(C)$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instance:</strong> A Boolean circuit $C \in C$ over two disjoint sets $X$ and $Y$ of variables, and an integer $k$.</td>
</tr>
<tr>
<td><strong>Parameter:</strong> $k$.</td>
</tr>
<tr>
<td><strong>Question:</strong> Does there exist a truth assignment $\alpha$ to $X$ of weight $k$, such that for all truth assignments $\beta$ to $Y$ the assignment $\alpha \cup \beta$ satisfies $C$?</td>
</tr>
<tr>
<td><strong>Complexity:</strong> $\exists^k \forall^* - \text{complete}$.$^{[33, 34]}$</td>
</tr>
</tbody>
</table>
Figure 1: An overview of parameterized complexity classes up to the second level of the Polynomial Hierarchy

<table>
<thead>
<tr>
<th>Problem</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\exists^k \forall^* \text{-WSAT}$</td>
<td>Instance: A quantified Boolean formula $\phi = \exists X. \forall Y. \psi$, and an integer $k$. Parameter: $k$. Question: Does there exist a truth assignment $\alpha$ to $X$ with weight $k$, such that $\forall Y. \psi[\alpha]$ evaluates to true? Complexity: $\exists^k \forall^*$-complete [33, 34].</td>
</tr>
<tr>
<td>$\exists^k \forall^* \text{-WSAT(3DNF)}$</td>
<td>Instance: A quantified Boolean formula $\phi = \exists X. \forall Y. \psi$ with $\psi \in 3\text{DNF}$, and an integer $k$. Parameter: $k$. Question: Does there exist a truth assignment $\alpha$ to $X$ with weight $k$, such that $\forall Y. \psi[\alpha]$ evaluates to true? Complexity: $\exists^k \forall^*$-complete [33, 34].</td>
</tr>
</tbody>
</table>
2.1 Weighted Quantified Boolean Satisfiability in the $\exists\forall^*$ Hierarchy

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

\[\exists^{\leq k}\forall^*\text{-WSAT}\]

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

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

\[\exists^{\geq k}\forall^*\text{-WSAT}\]

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

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

\[\exists^n-k\forall^*\text{-WSAT}\]

Instance: A quantified Boolean formula $\phi = \exists X.\forall Y.\psi$, and an integer $k$.
Parameter: $k$.
Question: Does there exist an assignment $\alpha$ 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 $k$-$\forall^*$ Classes

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

\[\exists^k\forall^*\text{-WSAT(2DNF)}\]

Instance: A quantified Boolean formula $\phi = \exists X.\forall Y.\psi$ with $\psi \in 2\text{DNF}$, and an integer $k$.
Parameter: $k$.
Question: Does there exist an assignment $\alpha$ to $X$, such that for all assignments $\beta$ to $Y$ of weight $k$ the assignment $\alpha \cup \beta$ satisfies $\psi$?

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

Let $d \geq 2$ be an arbitrary constant. Then the following problem is also $\exists^k\forall^*$-complete.
The following four types: fails otherwise \[10\]. We call a tree decomposition \((G, T)\) of minimum \(t\) the minimum of the widths of all tree decompositions of \(G\) with \(k\) a nonnegative integer. There is an fpt-algorithm that computes a tree decomposition of \(G\) with \(k\) if it exists, and fails otherwise \[10\]. We call a tree decomposition \((T, (B_t)_{t \in T})\) nice if every node \(t \in T\) is of one of the following four types:

- leaf node: \(t\) has no children and \(|B_t| = 1\);

Let \(ψ = δ_1 ∨ ⋯ ∨ δ_6\) be a DNF formula. For any subset \(Z \subseteq \text{Var}(ψ)\) of variables, we define the incidence graph \(IG(Z, ψ)\) of \(ψ\) with respect to \(Z\) to be the graph \(IG(Z, ψ) = (V, E)\), where \(V = Z ⋃ \{δ_1, \ldots, δ_6\}\) and \(E = \{\{δ_j, z\} : 1 \leq j \leq u, z \in Z, z \text{ occurs in the clause } δ_j\}\). If \(ψ\) is a DNF formula, \(Z \subseteq \text{Var}(ψ)\) is a subset of variables, and \((T, (B_t)_{t \in T})\) is a tree decomposition of \(IG(Z, ψ)\), 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 \((T, (B_t)_{t \in T})\) where \(T = (T, F)\) 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 \(T\); 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 \((T, (B_t)_{t \in T})\) 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 \(G\) with width \(k\) if it exists, and fails otherwise \[10\]. We call a tree decomposition \((T, (B_t)_{t \in T})\) 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 $B_t = B_{t'} \setminus \{v\}$ 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 $Q\text{Sat}_2$, where the treewidth of the incidence graph for certain subsets of variables is bounded.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Description</th>
<th>Parameter</th>
<th>Question</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\exists\forall$-SAT (incid.tw)</td>
<td>A quantified Boolean formula $\varphi = \exists X.\forall Y.\psi$, with $\psi$ in DNF.</td>
<td>The treewidth of the incidence graph $IG(X \cup Y, \psi)$ with respect to $X \cup Y$.</td>
<td>Is $\varphi$ satisfiable?</td>
<td>fixed-parameter tractable [15] [21].</td>
</tr>
<tr>
<td>$\exists\forall$-SAT (3-incid.tw)</td>
<td>A quantified Boolean formula $\varphi = \exists X.\forall Y.\psi$, with $\psi$ in DNF.</td>
<td>The treewidth of the incidence graph $IG(X, \psi)$ with respect to $X$.</td>
<td>Is $\varphi$ satisfiable?</td>
<td>para-$\Sigma_2^P$-complete [32] [34].</td>
</tr>
<tr>
<td>$\exists\forall$-SAT (Y-incid.tw)</td>
<td>A quantified Boolean formula $\varphi = \exists X.\forall Y.\psi$, with $\psi$ in DNF.</td>
<td>The treewidth of the incidence graph $IG(Y, \psi)$ with respect to $Y$.</td>
<td>Is $\varphi$ satisfiable?</td>
<td>para-NP-complete [32] [34].</td>
</tr>
</tbody>
</table>

The above problems are parameterized by the treewidth of the incidence graph of the formula $\psi$ (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.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Description</th>
<th>Parameter</th>
<th>Question</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>QBF-SAT (#\forall-vars)</td>
<td>A quantified Boolean formula $\varphi$.</td>
<td>The number of universally quantified variables of $\varphi$.</td>
<td>Is $\varphi$ true?</td>
<td>para-NP-complete [1] [7] [32].</td>
</tr>
</tbody>
</table>

### 2.5 Minimization for DNF Formulas

Let $\varphi$ be a propositional formula in DNF. We say that a set $C$ of literals is an *implicant* of $\varphi$ if all assignments that satisfy $\bigwedge_{t \in C} t$ also satisfy $\varphi$. Moreover, we say that a DNF formula $\varphi'$ is a *term-wise subformula* of $\varphi$ 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 $\Sigma_2^P$-complete by Umans [48].
Shortest-Implicant-Core (core size)
Instance: A DNF formula \( \varphi \), an implicant \( C \) of \( \varphi \), and an integer \( k \).
Parameter: \( k \).
Question: Does there exists an implicant \( C' \subseteq C \) of \( \varphi \) of size \( k \)?
Complexity: \( \exists \forall^* \)-complete \[32, 34\].

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

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

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

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\(_k\)-Sat, which is canonical for the different levels of the Boolean Hierarchy (see Section 1). The problem is complete for the class FPT\(^{\text{NP}}[f(k)]\).

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 \( (\varphi_1, \ldots, \varphi_k) \in \text{BH}_k\)-Sat?
Complexity: FPT\(^{\text{NP}}[f(k)]\)-complete \[23\].

The above problem is used to show the following lower bound result for FPT\(^{\text{NP}}[f(k)]\)-complete problems. No FPT\(^{\text{NP}}[f(k)]\)-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 \( r = (a_1 \lor \ldots \lor a_k \leftarrow b_1, \ldots, b_m, \not c_1, \ldots, \not c_n) \), 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 \( \text{At}(P) \) denote the set of all atoms occurring in \( P \). By *literals* we mean atoms \( a \) or their negations \( \text{not} \ a \).

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 \( \text{not} \ a \) in the body, for each \( a \in M \), and (ii) removing literals \( \text{not} \ a \) 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.

<table>
<thead>
<tr>
<th>Parameterization</th>
<th>Instance</th>
<th>Parameter</th>
<th>Question</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASP-CONSISTENCY(#cont.atoms)</td>
<td>A disjunctive logic program ( P ).</td>
<td>The number of contingent atoms of ( P ).</td>
<td>Does ( P ) have an answer set?</td>
<td>para-co-NP-complete [33, 34].</td>
</tr>
<tr>
<td>ASP-CONSISTENCY(#cont.rules)</td>
<td>A disjunctive logic program ( P ).</td>
<td>The number of contingent rules of ( P ).</td>
<td>Does ( P ) have an answer set?</td>
<td>( \exists^k \forall^* )-complete [34].</td>
</tr>
<tr>
<td>ASP-CONSISTENCY(#disj.rules)</td>
<td>A disjunctive logic program ( P ).</td>
<td>The number of disjunctive rules of ( P ).</td>
<td>Does ( P ) have an answer set?</td>
<td>( \exists^\forall^k )-W[( P )]-complete [34].</td>
</tr>
<tr>
<td>ASP-CONSISTENCY(#dual-Horn.rules)</td>
<td>A disjunctive logic program ( P ).</td>
<td>The number of rules of ( P ) that are dual-Horn.</td>
<td>Does ( P ) have an answer set?</td>
<td>( \exists^\forall^k )-W[( P )]-complete [30].</td>
</tr>
<tr>
<td>ASP-CONSISTENCY(str.norm.bd-size)</td>
<td>A disjunctive logic program ( P ).</td>
<td>The size of the smallest normality-backdoor for ( P ).</td>
<td>Does ( P ) have an answer set?</td>
<td>para-NP-complete [25].</td>
</tr>
<tr>
<td>ASP-CONSISTENCY(max.atom.occ.)</td>
<td>A disjunctive logic program ( P ).</td>
<td>The maximum number of times that any atom occurs in ( P ).</td>
<td>Does ( P ) have an answer set?</td>
<td>para-( \Sigma_2^P )-complete [33, 34].</td>
</tr>
</tbody>
</table>

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

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 \( \alpha \) 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 \( \Sigma^P_2 \)-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:** \( \exists^* \forall^k \text{-W}[1] \)-complete [41].

**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 $P = (V, H, M, T)$, and a positive integer $m$.

**Parameter:** The number of clauses in $T$ that are not Horn.

**Question:** Does there exist a solution $S$ of $P$ 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 $(|C| + k)$-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 $\Pi^p_2$-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 $\Pi^p_2$-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-$\Pi^p_2$-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].
The following problem is concerned with the problem of verifying whether a linear temporal logic formula \( \phi \) involves variables, (ii) \( I \) and functions in terms of \( S \). Defined in terms of the operators \( X \) and \( U \). Properties related to the future of paths. LTL formulas is satisfied on all paths in a Kripke structure. This problem is of importance in the area of software and hardware verification \[8\].

### 5.2 Bounded Model Checking

A first-order logic sentence \( \psi \) is quantifier-free, and a finite vocabulary \( \tau \) is a finite set of relation symbols. Each relation symbol \( R \) has an arity \( \text{arity}(R) \geq 1 \). A structure \( A \) of vocabulary \( \tau \), or \( \tau \)-structure (or simply \( \tau \)-structure), consists of a set \( A \) called the domain and an interpretation \( A^\tau \subseteq A^{\text{arity}(R)} \) for each relation symbol \( R \in \tau \). We use the usual definition of truth of a first-order logic sentence \( \varphi \) over the vocabulary \( \tau \) in a \( \tau \)-structure \( A \). We let \( A \models \varphi \) denote that the sentence \( \varphi \) is true in structure \( A \). If \( \varphi \) is a first-order formula with free variables \( \text{Free}(\varphi) \), and \( \mu : \text{Free}(\varphi) \to A \) is an assignment, we use the notation \( A, \mu \models \varphi \) to denote that \( \varphi \) is true in structure \( A \) under the assignment \( \mu \).

### 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^*\text{-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**: \( \text{para-}\Pi_2^P\text{-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 \( 3^k\forall^* \). We introduce a few notions that we need for defining the model checking perspective on \( 3^k\forall^* \). A (relational) vocabulary \( \tau \) is a finite set of relation symbols.

Each relation symbol \( R \) has an arity \( \text{arity}(R) \geq 1 \). A structure \( A \) of vocabulary \( \tau \), or \( \tau \)-structure (or simply \( \tau \)-structure), consists of a set \( A \) called the domain and an interpretation \( A^\tau \subseteq A^{\text{arity}(R)} \) for each relation symbol \( R \in \tau \). We use the usual definition of truth of a first-order logic sentence \( \varphi \) over the vocabulary \( \tau \) in a \( \tau \)-structure \( A \). We let \( A \models \varphi \) denote that the sentence \( \varphi \) is true in structure \( A \). If \( \varphi \) is a first-order formula with free variables \( \text{Free}(\varphi) \), and \( \mu : \text{Free}(\varphi) \to A \) is an assignment, we use the notation \( A, \mu \models \varphi \) to denote that \( \varphi \) is true in structure \( A \) under the assignment \( \mu \).

\( 3^k\forall^*\text{-MC} \)

**Instance**: A first-order logic sentence \( \varphi = \exists x_1, \ldots, x_k, \forall y_1, \ldots, y_n, \psi \) over a vocabulary \( \tau \), where \( \psi \) is quantifier-free, and a finite \( \tau \)-structure \( A \).

**Parameter**: \( k \).

**Question**: Is it the case that \( A \models \varphi \)?

**Complexity**: \( 3^k\forall^*\text{-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 \( \varphi_1, \varphi_2 \in \text{LTL} \), then so are \( \varphi_1 \lor \varphi_2, F\varphi_1 \) (Future), \( X\varphi_1 \) (neXt), \( \varphi_1 U \varphi_2 \) (\( \varphi_1 \) Until \( \varphi_2 \)). (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 \( \pi \) of \( K \) is an infinite sequence \( (s_0, s_1, s_2, \ldots) \) 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 \( \pi \). A
suffixed a path is defined as \( \pi^i = (s_i, s_{i+1}, \ldots) \). 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 \( \varphi \) holds along a path \( \pi^i \), written \( \pi^i \models \varphi \), is specified by the following conditions:

\[
\begin{align*}
\pi^i \models v \in V & \quad \text{iff} \quad v \in L(\pi(i)), \\
\pi^i \models \varphi_1 \lor \varphi_2 & \quad \text{iff} \quad \pi^i \models \varphi_1 \text{ or } \pi^i \models \varphi_2, \\
\pi^i \models F\varphi & \quad \text{iff} \quad \text{for some } j \in \mathbb{N}, \pi^{i+j} \models \varphi, \\
\pi^i \models \varphi_1 \cup \varphi_2 & \quad \text{iff} \quad \text{for some } j \in \mathbb{N}, \pi^{i+j} \models \varphi_2 \quad \text{and} \quad \pi^i \models \varphi_1 \text{ for all } i \leq \ell < i + j.
\end{align*}
\]

Then, an LTL formula \( \varphi \) holds in a Kripke structure \( K \) if and only if \( \pi \models \varphi \) for all initialized paths \( \pi \) of \( K \). Related to the model checking problem is the question whether a witness exists: a formula \( \varphi \) has a witness in \( K \) if there is an initialized path \( \pi \) 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 \( \varphi \) holds along a suffix \( \pi^i \) of a \((k, l)\)-lasso \( \pi \) in the bounded semantics, written \( \pi^i \models_k \varphi \), is specified by the following conditions:

\[
\begin{align*}
\pi^i \models_k X\varphi & \quad \text{iff} \quad \pi^{i+1} \models_k \varphi \quad \text{if } i < k, \\
\pi^i \models_k F\varphi & \quad \text{iff} \quad \text{for some } j \in \{0, \ldots, k\}, \pi^{i+j} \models_k \varphi, \\
\pi^i \models_k \varphi_1 \cup \varphi_2 & \quad \text{iff} \quad \text{for some } j \in \{0, \ldots, k\}, \pi^{i+j} \models_k \varphi_2, \quad \text{and} \quad \pi^i \models_k \varphi_1 \text{ for all } i \leq \ell < k \text{ and all } l \leq \ell < j \quad \text{if } j < i, \\
& \quad \text{if } j \geq i.
\end{align*}
\]

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

\[
\begin{align*}
\pi^i \models_k X\varphi & \quad \text{iff} \quad \pi^{i+1} \models_k \varphi \quad \text{and} \quad i < k, \\
\pi^i \models_k F\varphi & \quad \text{iff} \quad \text{for some } j \in \{0, \ldots, k\}, \pi^{i+j} \models_k \varphi, \\
\pi^i \models_k \varphi_1 \cup \varphi_2 & \quad \text{iff} \quad \text{for some } j \in \{0, \ldots, k\}, \pi^{i+j} \models_k \varphi_2, \quad \text{and} \quad \pi^i \models_k \varphi_1 \text{ for all } i \leq \ell < j.
\end{align*}
\]

Note that \( \pi \models_k \varphi \) implies \( \pi \models \varphi \) for all paths \( \pi \). 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, 10]. 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.

<table>
<thead>
<tr>
<th><strong>BMC-Witness</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instance:</strong> An LTL formula ( \varphi ), a Kripke structure ( K ), and an integer ( k \geq 1 ).</td>
<td></td>
</tr>
<tr>
<td><strong>Parameter:</strong> ( k ).</td>
<td></td>
</tr>
<tr>
<td><strong>Question:</strong> Is there some path ( \pi ) of ( K ) such that ( \pi \models_k \varphi )?</td>
<td></td>
</tr>
<tr>
<td><strong>Complexity:</strong> in para-NP [9].</td>
<td></td>
</tr>
</tbody>
</table>

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

We let \( \Pi_2 \) denote the class of all first-order formulas of the form \( \forall y_1, \ldots, y_n \exists x_1, \ldots, x_m \psi \), where \( \psi \) is quantifier-free. Let \( \varphi(X) \) be a first-order formula over \( \tau \), with a free relation variable \( X \) with arity \( s \). Consider the following parameterized problem.

\[
\forall^k \exists^*-FD(\tau',\tau)
\]

**Instance:** A \( \tau' \)-structure \( B \), and an integer \( k \).

**Parameter:** \( k \).

**Question:** Is it the case that for each \( \tau \)-structure \( A \) extending \( B \) with weight \( k \), there exists some relation \( S \subseteq A^s \) such that \( A \models \varphi(S) \)?

**Complexity:** in \( \forall^k \exists^* \) for each \( \varphi(X) \), \( \tau' \) and \( \tau \); \( \forall^k \exists^* \)-hard for some \( \varphi(X) \in \Pi_2 \), \( \tau' \) and \( \tau \) [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 \( \text{FO}_\tau \) denote the set of all first-order formulas over the vocabulary \( \tau \) with a free relation variable \( X \). We then get the following characterization of \( \forall^k \exists^* \) [34]:

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

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

\[
\forall^* \exists^k \cdot FD(\tau',\tau)
\]

**Instance:** A \( \tau' \)-structure \( B \), and an integer \( k \).

**Parameter:** \( k \).

**Question:** Is it the case that for each \( \tau \)-structure \( A \) extending \( B \), there exists some relation \( S \subseteq A^s \) with \( |S| = k \) such that \( A \models \varphi(S) \)?

**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 \( \Phi \) is a finite nonempty set \( \Phi = \{ \varphi_1, \ldots, \varphi_n, \neg \varphi_1, \ldots, \neg \varphi_n \} \) of formulas that is closed under complementation. A judgment set \( J \) for an agenda \( \Phi \) 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 \( \varphi_i \), and we call it consistent if there exists an assignment that makes all formulas in \( J \) true. Let \( J(\Phi) \) denote the set of all complete and consistent subsets of \( \Phi \). We call a sequence \( J \in J(\Phi)^n \) of complete and consistent subsets a profile. A (resolute) judgment aggregation procedure for the agenda \( \Phi \) and \( n \) individuals is a function \( F : J(\Phi)^n \to P(\Phi) \setminus \emptyset \) that returns for each profile \( J \) a non-empty set \( F(J) \) of non-empty judgment sets. An example is the majority rule \( F^\text{maj} \), where \( F^\text{maj}(J) = \{ J' \} \) and where \( \varphi \in J' \) if and only if \( \varphi \) occurs in the majority of judgment sets in \( J \), for each \( \varphi \in \Phi \). We call \( F \) complete and consistent, if each \( J' \in F(J) \) is complete and consistent, respectively, for every \( J \in J(\Phi)^n \). For instance, the majority rule \( F^\text{maj} \) is complete, whenever the number \( n \) of individuals is odd. An agenda \( \Phi \) is safe with respect to an aggregation procedure \( F \), if \( F \) is consistent when applied to profiles of judgment sets over \( \Phi \). We say that an agenda \( \Phi \) satisfies the median property (MP) if every inconsistent subset of \( \Phi \) 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 \( \Phi \) is safe for the majority rule if and only if \( \Phi \) satisfies the MP [21] [41]. The problem of deciding whether an agenda satisfies the MP is \( \Pi_2^P \)-complete [21].
Maj-Agenda-Safety(formula size)

Instance: An agenda Φ.
Parameter: ℓ = max{|ϕ| : ϕ ∈ Φ}.
Question: Is Φ safe for the majority rule?

Complexity: para-Π^2-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-complete [22, 23].

Maj-Agenda-Safety(degree + formula size)

Instance: An agenda Φ containing only CNF formulas, where ℓ = max{|ϕ| : ϕ ∈ B(Φ)}, and where d is the degree of Φ.
Parameter: ℓ + d.
Question: Is Φ safe for the majority rule?

Complexity: para-Π^2-complete [22, 23].

The above three parameterized problems remain para-Π^2-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: |Φ|.
Question: Is Φ safe for the majority rule?

Complexity: FPT^{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 = |Φ| [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: ∀^k∃^* 23].

Let Φ = {φ₁, ..., φᵣ, ¬φ₁, ..., ¬φᵣ} be an agenda, where each φᵢ 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 Var(Φ) occurring in the agenda, and two variables are connected by an edge if there exists a formula φᵢ in which they both occur. The formula incidence graph of Φ is a bipartite graph whose vertices consist of (1) the variables Var(Φ) occurring in the agenda and (2) the formulas φᵢ ∈ Φ. A variable x ∈ Var(Φ) is connected by an edge with a formula φᵢ ∈ Φ if x occurs in φᵢ, i.e., x ∈ Var(φᵢ). The clausal primal graph of Φ has as vertices the variables Var(Φ) occurring in the agenda, and two variables are connected by an edge if there exists a formula φᵢ and a clause c ∈ φᵢ in which they both occur. The clausal incidence graph of Φ is a bipartite graph whose vertices consist of (1) the variables Var(Φ) occurring in the agenda and (2) the clauses c occurring in formulas φᵢ ∈ Φ.
A variable \( x \in \text{Var}(\Phi) \) is connected by an edge with a clause \( c \) of the formula \( \varphi_i \in \Phi \) if \( x \) occurs in \( c \), i.e., \( x \in \text{Var}(c) \). Consider the following parameterizations of the problem \textsc{Maj-Agenda-Safety}.

**\textsc{Maj-Agenda-Safety}(l-tw)**

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

**\textsc{Maj-Agenda-Safety}(l-tw*)**

*Instance:* An agenda \( \Phi \) containing only CNF formulas.
*Parameter:* The treewidth of the formula incidence graph of \( \Phi \).
*Question:* Is \( \Phi \) safe for the majority rule?
*Complexity:* para-\( \Pi^P_2 \)-complete [22].

**\textsc{Maj-Agenda-Safety}(c-tw)**

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

**\textsc{Maj-Agenda-Safety}(c-tw*)**

*Instance:* An agenda \( \Phi \) containing only CNF formulas.
*Parameter:* The treewidth of the clausal incidence graph of \( \Phi \).
*Question:* Is \( \Phi \) 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 \( \rho \) of the ATM \( M \) on an input \( x \) is a tree whose nodes correspond to configurations of \( M \) in such a way that (1) for each non-root node \( v \) of the tree with parent node \( v' \), the machine \( M \) can transition from the configuration corresponding to \( v' \) to the configuration corresponding to \( v \), (2) the root node corresponds to the initial configuration of \( M \), and (3) each child node corresponds to a halting configuration. A computation path in a run of \( M \) is a root-to-leaf path in the run. Moreover, the nodes of a run \( \rho \) are labelled accepting or rejecting, according to the following definition. A leaf of \( \rho \) 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 \( \rho \) that corresponds to an existential configuration is labelled accepting if at least one of its children is labelled accepting. A non-leaf node of \( \rho \) that corresponds to a universal configuration is labelled accepting if all of its children is labelled accepting. An ATM \( M \) is 2-alternating if for each input \( x \), each computation path in the run of \( M \) 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 \( \exists^\ell\forall \)-Turing machine (or simply \( \exists^\ell\forall \)-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 \( \exists^\ell\forall \)-machine \( M \) halts (on the empty string) with existential cost \( \ell \) and universal cost \( t \) if: (1) there is an accepting run of \( M \) with the empty input \( \epsilon \), and (2) each computation path of \( M \) contains at most \( \ell \) 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$^m$. |
|-------------------------|------------------|
| **Instance:** Positive integers $m, k, t \geq 1$, and a $\exists \forall$-machine $M$ with $m$ tapes. |
| **Parameter:** $k$. |
| **Question:** Does $M$ 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 a $\exists \forall$-machine $M$ with $m$ tapes. |
| **Parameter:** $k$. |
| **Question:** Does $M$ 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 $M$ such that there exists a computable function $f$ and a polynomial $p$ such that: (1) $M$ decides $P$ in time $f(k) \cdot p(|x|)$; and (2) for all instances $(x, k)$ of $P$ and each computation path $R$ of $M$ 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


Index of Problems by Complexity

fixed-parameter tractable

\∃∀-\text{Sat}(\text{incid.tw}), 9
\text{Maj-Agenda-Safety}(f\text{-tw}), 18

in para-NP

\text{BMC-Witness}, 15

para-NP-complete

\text{Abduction}(\text{Horn-bd-size}), 12
\text{Abduction}(\text{Krom-bd-size}), 12
\text{ASP-consistency}(\text{str.norm.bd-size}), 11
\∃∀-\text{Sat}(\forall\text{-incid.tw}), 9
\text{QBF-Sat}(\#\forall\text{-vars}), 9
3\text{-Coloring-Extension}(\#\text{leaves}), 13

para-co-NP-complete

\text{ASP-consistency}(\#\text{cont.atoms}), 11
\text{Maj-Agenda-Safety}(c\text{-tw}), 18
\text{Maj-Agenda-Safety}(c\text{-tw}^*), 18

para-co-NP-hard

\text{DNF-Minimization}(\text{core size}), 10

in FPT\text{NP}[f(k)]

\text{DNF-Minimization}(\text{core size}), 10
FPT\text{NP}[f(k)]-complete

\text{BH-Sat}(\text{level}), 10
\text{Maj-Agenda-Safety}(\text{agenda size}), 17

in \∃∀^*

\text{DNF-Minimization}(\text{core size}), 10
\∃∀^*-complete

\text{ASP-consistency}(\#\text{cont.rules}), 11
\text{DNF-Minimization}(\text{reduction size}), 10
\∃∀^*-\text{MC}, 14
\∃∀^*-\text{TM-halt}^+, 19
\∃∀^*-\text{TM-halt}^-, 19
\∃∀^*-\text{WSat}, 6
\exists∀^*-\text{WSat}(3\text{DNF}), 6
\exists^k∀^*-\text{WSat}, 7
\exists^k∀^*-\text{WSat}, 7
\∃∀^*-\text{WSat}, 7
\∃∀^*-\text{WSat}, 7
\text{Shortest-Implicant-Core}(\text{core size}), 10
\text{Shortest-Implicant-Core}(\text{red. size}), 10

\∃∀^k-W[1]-complete

\text{Abduction}(\#\text{non-Krom-clauses}), 12
\exists∀^k\text{-WSat}(2\text{DNF}), 7
\exists∀^k\text{-WSat}(d\text{-DNF}), 8
\exists∀^k-W[t]-complete, t \geq 1
\exists∀^k\text{-WSat}(\text{chrc}_{t,u}), 7
\exists∀^k-W[\text{SAT}]-complete
\exists∀^k\text{-WSat}(\text{form}), 7
\exists∀^k-W[P]-complete

\text{Abduction}(\#\text{non-Horn-clauses}), 13
\text{ASP-consistency}(\#\text{disj.rules}), 11
\text{ASP-consistency}(\#\text{dual-Horn.rules}), 11
\exists∀^k\text{-WSat}(\text{chrc}), 7

\exists∀^k-\text{WSat}(\forall\text{-anti-monotone}), 8
\exists∀^k-\text{WSat}(\forall\text{-monotone}), 8

\forall^k∀^*-complete

\forall^k∀^*-\text{FD}(\forall^*, \forall^*), 16
\text{Robust-CSP-Sat}, 12
3\text{-Coloring-Extension}(\#\text{col.leaves}), 14
\forall^k∀^*-hard

\text{Maj-Agenda-Safety}(\text{counterex. size}), 17
\forall^k∀^-[W[1]]-complete

\text{Small-Clique-Extension}, 13
\forall^k∀^-[W[1]]-hard

\forall^k∀^*-\text{FD}(\forall^*, \forall^*), 16

para-\Sigma_2^P-complete

\text{ASP-consistency}(\text{max. atom.occ}), 11
\exists∀-\text{Sat}(3\text{incid.tw}), 9
\exists∀^k∀^*-\text{WSat}, 7

para-\Pi_2^P-complete

\text{Maj-Agenda-Safety}(\text{degree}), 17
\text{Maj-Agenda-Safety}(\text{degree+form. size}), 17
\text{Maj-Agenda-Safety}(f\text{-tw}^*), 18
\text{Maj-Agenda-Safety}(\text{formula size}), 17
3\text{-Coloring-Extension}(\text{degree}), 13
3\text{-Coloring-Extension}(\#\text{uncol.leaves}), 14