A Compendium of Parameterized Problems at Higher Levels of the Polynomial Hierarchy

de Haan, R.; Szeider, S.

DOI
10.3390/a12090188

Publication date
2019

Document Version
Final published version

Published in
Algorithms

algorithms

MDPI

Article

A Compendium of Parameterized Problems at Higher Levels of the Polynomial Hierarchy

Ronald de Haan [1,*] **and Stefan Szeider** [2]

[1] Institute for Logic, Language and Computation (ILLC), University of Amsterdam, 1000–1183 Amsterdam, The Netherlands

[2] Algorithms and Complexity Group, Institute for Logic and Computation, Faculty of Informatics, Technische Universität Wien, 1040 Vienna, Austria

* Correspondence: me@ronalddehaan.eu

Received: 19 July 2019; Accepted: 3 September 2019; Published: 9 September 2019

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 problems are parameterized versions of problems whose complexity lies at the second level of the Polynomial Hierarchy or higher.

Keywords: parameterized complexity; polynomial hierarchy; fpt-reductions

1. Introduction

The remarkable performance of today's SAT solvers (see, e.g., [1]) offers a practically successful strategy for solving NP-complete combinatorial search problems by reducing them in polynomial time to the propositional satisfiability problem—also known as 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 this strategy is to use fixed-parameter tractable reductions (or fpt-reductions)—i.e., reductions that are computable in time $f(k)n^c$ for some computable function f and some constant c—as they can exploit some structural aspects of the problem instances in terms of a problem parameter k.

To illustrate this approach, let us consider as an example the problem of deciding the truth of quantified Boolean formulas (QBFs) of the form $\varphi = \exists X.\forall Y.\psi$, where ψ is a DNF formula over the variables in $X \cup Y$. (We define and consider this example problem—as well as the parameterized variants of this problem that we consider in this section—in more detail in Section 3.3.) This problem is Σ_2^{p}-complete [2,3], so there is no polynomial-time reduction from this problem to SAT, unless the Polynomial Hierarchy collapses. In other words, the strategy of first reducing the problem to SAT, and then using a SAT solver to solve the problem seems not viable.

Another strategy would be to consider the problem from a parameterized point of view—i.e., investigating in which cases structural properties of the input allow the problem to be solved efficiently (in fpt-time). For example, if the incidence treewidth of the DNF formula ψ is bounded, we can solve the problem in fpt-time [4,5]. However, this is the case only in very restricted settings, and so this strategy is also only viable to a limited extent.

The approach of using fpt-reductions to SAT combines the merits of the above two strategies. It uses the remarkable performance of today's SAT solvers, and it can exploit structural properties of the problem inputs. As a result, this approach has the potential of being useful in a much wider range of settings than either of the two separate strategies. It can work for problems that lie at higher levels of the Polynomial Hierarchy. Moreover, it can work for choices of problem parameters that are much less restrictive than those needed to get traditional fixed-parameter tractability results.

As a demonstration of this, let us consider two other parameterized variants of our example problem of deciding the truth of formulas of the form $\varphi = \exists X.\forall Y.\psi$. In the first variant, the parameter is the treewidth of the incidence graph of ψ where we delete all nodes corresponding to variables in X—in other words, the incidence graph of ψ with respect to the universally quantified variables. This variant is para-NP-complete [6,7], which means that it allows an fpt-reduction to SAT. Thus, for this variant, we can exploit the problem parameter to efficiently reduce the problem to SAT, and subsequently use a SAT solver to solve the problem. In the second variant that we consider, the parameter is the treewidth of the incidence graph of ψ where we delete all nodes corresponding to variables in Y—in other words, the incidence graph of ψ with respect to the existentially quantified variables. This variant is para-Σ_2^{P}-complete [6,7], indicating that fpt-reductions to SAT are not possible (unless the Polynomial Hierarchy collapses).

For both of these variants of our example problem, both the approach of polynomial-time reducing the problem to SAT and the approach of finding (traditional) fixed-parameter tractable algorithms do not work. However, for only one of them, an fpt-reduction to SAT is possible (under common complexity-theoretic assumptions). This indicates that, to establish whether an fpt-reduction to SAT is possible, a suitable parameterized complexity analysis at higher levels of the Polynomial Hierarchy is needed. Various parameterized complexity classes have been developed for this purpose [7,8] that, together with previously studied classes, enable such analyses. This includes both classes that correspond to various types of fpt-reductions to SAT—e.g., para-NP, para-co-NP, and $\mathrm{FPT}^{\mathrm{NP}}[\mathrm{few}]$—and classes that indicate that no fpt-reductions to SAT are possible, under suitable complexity-theoretic assumptions—e.g., $\Sigma_2^{\mathrm{P}}[k*]$, $\Sigma_2^{\mathrm{P}}[*k,1]$, $\Sigma_2^{\mathrm{P}}[*k,\mathrm{P}]$, $\Pi_2^{\mathrm{P}}[k*]$, $\Pi_2^{\mathrm{P}}[*k,1]$, $\Pi_2^{\mathrm{P}}[*k,\mathrm{P}]$, para-$\Sigma_2^{\mathrm{P}}$, para-$\Pi_2^{\mathrm{P}}$, PH[level], and para-PSPACE. (We define all of these parameterized complexity classes in Sections 2.2 and 2.3.)

Outline

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 indicating 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 [9] and Cesati [10] 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.

The remainder of the paper is structured as follows. In Section 2, we give an overview of the parameterized complexity classes involved in the classification of whether problems allow an fpt-reduction to SAT, as well as some other notions and definitions that are used throughout the paper. Then, in Sections 3–6, we present (i) problems related to (quantified variants of) propositional logic, (ii) problems from the area of Knowledge Representation and Reasoning, (iii) graph problems, and (iv) other problems, respectively. Finally, we conclude in Section 7.

2. Preliminaries

2.1. 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 (see, e.g., [11,12]).

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* [2,3,12,13] contains a hierarchy of increasing complexity classes Σ_i^{P}, for all $i \geq 0$. We give a characterization of these classes based on the satisfiability problem of various classes of quantified Boolean formulas. A *quantified Boolean formula* is a formula of the form $Q_1 X_1 Q_2 X_2 \ldots Q_m X_m \psi$, 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 $\gamma = \{x_1 \mapsto d_1, \ldots, x_n \mapsto d_n\}$ 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 $\varphi[x_1 \mapsto d_1, \ldots, x_n \mapsto d_n]$ to denote $\varphi[\gamma]$. For each $i \geq 1$, the decision problem QSAT$_i$ is defined as follows.

QSAT$_i$
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 φ true?

Input formulas to the problem QSAT$_i$ are called Σ_i^{p}-formulas. For each nonnegative integer $i \geq 0$, the complexity class Σ_i^{p} can be characterized as the closure of the problem QSAT$_i$ under polynomial-time reductions [2,3]—that is, all decision problems that are polynomial-time reducible to QSAT$_i$. The Σ_i^{p}-hardness of QSAT$_i$ holds already when the matrix of the input formula is restricted to 3CNF for odd i, and restricted to 3DNF for even i. Note that the class Σ_0^{p} coincides with P, and the class Σ_1^{p} coincides with NP. For each $i \geq 1$, the class Π_i^{p} is defined as co-Σ_i^{p}—that is, $\Pi_i^{\mathrm{p}} = \{\, \{0,1\}^* \setminus L : L \in \Sigma_i^{\mathrm{p}} \,\}$.

The classes Σ_i^{p} and Π_i^{p} can also be defined by means of nondeterministic Turing machines with an oracle. Intuitively, oracles are black-box machines that can solve a problem in a single time step—for more details, see, e.g., Chapter 3 of [11]. For any complexity class C, we let NP^C be the set of decision problems that are decided in polynomial time by a nondeterministic Turing machine with an oracle for a problem that is in the class C. Then, the classes Σ_i^{p} and Π_i^{p}, for $i \geq 0$, can be equivalently defined by letting $\Sigma_0^{\mathrm{p}} = \Pi_0^{\mathrm{p}} = \mathrm{P}$, and for each $i \geq 1$ letting $\Sigma_i^{\mathrm{p}} = \mathrm{NP}^{\Sigma_{i-1}^{\mathrm{p}}}$ and $\Pi_i^{\mathrm{p}} = \text{co-NP}^{\Sigma_{i-1}^{\mathrm{p}}}$.

The Polynomial Hierarchy also includes complexity classes between Σ_i^{p} and $\Sigma_{i+1}^{\mathrm{p}}$—such as the classes $\Delta_{i+1}^{\mathrm{p}}$ and $\Theta_{i+1}^{\mathrm{p}}$. The class $\Delta_{i+1}^{\mathrm{p}}$ consists of all decision problems that are decided in polynomial time by a deterministic Turing machine with an oracle for a problem that is in the class Σ_i^{p}. Similarly, the class $\Theta_{i+1}^{\mathrm{p}}$ consists of all decision problems that are decided in polynomial time by a deterministic Turing machine with an oracle for a problem that is in the class Σ_i^{p}, with the restriction that the Turing machine is only allowed to make $O(\log n)$ oracle queries, where n denotes the input size [14,15]. It holds that $\Sigma_i^{\mathrm{p}} \cup \Pi_i^{\mathrm{p}} \subseteq \Theta_{i+1}^{\mathrm{p}} \subseteq \Delta_{i+1}^{\mathrm{p}} \subseteq \Sigma_{i+1}^{\mathrm{p}} \cap \Pi_{i+1}^{\mathrm{p}}$.

There are also natural decision problems that are located between NP and Θ_2^{p}. The *Boolean Hierarchy* (BH) [16–18] 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 $(\varphi_1, \ldots, \varphi_i)$ of propositional formulas such that both $(\varphi_1, \ldots, \varphi_{i-1}) \in \mathrm{BH}_{(i-1)}$-SAT and φ_i is unsatisfiable. For odd $i \geq 2$, the problem BH$_i$-SAT consists of all sequences $(\varphi_1, \ldots, \varphi_i)$ of propositional formulas such that $(\varphi_1, \ldots, \varphi_{i-1}) \in \mathrm{BH}_{(i-1)}$-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 NP $\cup$ co-NP $\subseteq$ BH$_2$ $\subseteq$ BH$_3$ $\subseteq \cdots \subseteq$ BH $\subseteq \Theta_2^{\mathrm{p}}$.

2.2. Parameterized Complexity

We introduce some core notions from parameterized complexity theory. For an in-depth treatment, we refer to other sources [19–23]. A *parameterized problem* L is a subset of $\Sigma^* \times \mathbb{N}$ for some finite alphabet Σ. For an instance $(x, k) \in \Sigma^* \times \mathbb{N}$, we call x 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 $(x,k) \in L$ in time $f(k) \cdot |x|^c$, where $|x|$ denotes the size of x. Such an algorithm is called an *fpt-algorithm*, and this amount of time is called *fpt-time*. FPT is the class of all parameterized problems that are fixed-parameter tractable. 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 $|x|^k$, which are also polynomial for fixed k, but are already impractical for, say, $k > 5$.

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' for which there exist some computable function $g : \mathbb{N} \to \mathbb{N}$ such that for all $(x,k) \in \Sigma^* \times \mathbb{N}$: (i) (x,k) is a yes-instance of L if and only if $(x',k') = R(x,k)$ is a yes-instance of L', (ii) $k' \leq g(k)$, and (iii) R is computable in fpt-time. Let K be a parameterized complexity class. A parameterized problem L is K-*hard* if for every $L' \in$ K there is an fpt-reduction from L' to L. A problem L is K-*complete* if it is both in K and K-hard. Reductions that satisfy properties (i) and (ii) but that are computable in time $O(|x|^{f(k)})$, for some fixed computable function f, we call *xp-reductions*.

The parameterized complexity classes W[t], for $t \geq 1$, 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 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 $[\ \cdot\ ]^{\text{fpt}}$—that is, for any set $\mathcal{L}$ of parameterized problems, $[\ \mathcal{L}\ ]^{\text{fpt}}$ is the set of all parameterized problems L' that are fpt-reducible to some problem $L \in \mathcal{L}$. 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 a computable function $f : \mathbb{N} \to \Sigma^*$, and a problem $P \subseteq \Sigma^*$ 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 $(x, f(k)) \in P$. (Here, we implicitly use a representation of pairs of strings in $\Sigma^* \times \Sigma^*$ as strings in Σ^*.) 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 [24]. The class para-K can be seen as a direct analogue of the class K in parameterized complexity. In particular, for the case of K = P, we have FPT = para-P.

We consider the following (trivial) parameterization of SAT, the satisfiability problem for propositional logic. We let $\text{SAT}_1 = \{\ (\varphi, 1) : \varphi \in \text{SAT}\ \}$. In other words, SAT_1 is the parameterized variant of SAT where the parameter is the constant value 1. Similarly, we let $\text{UNSAT}_1 = \{\ (\varphi, 1) : \varphi \in$

UNSAT }. The problem SAT_1 is para-NP-complete, and the problem UNSAT_1 is para-co-NP-complete. In other words, the class para-NP consists of all parameterized problems that can be fpt-reduced to SAT_1, and para-co-NP consists of all parameterized problems that can be fpt-reduced to UNSAT_1.

Another analogue to the classical complexity class K is the parameterized complexity class $\mathrm{XK}^{\mathrm{nu}}$, that is defined as the class of those parameterized problems Q whose slices Q_k are in K, i.e., for each positive integer k the classical problem $Q_k = \{ x : (x,k) \in Q \}$ is in K [20]. For instance, the class $\mathrm{XP}^{\mathrm{nu}}$ 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 Q_k is polynomial-time solvable. There are also uniform variants XK of these classes $\mathrm{XK}^{\mathrm{nu}}$. 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 $|x|^{f(k)}$ [20,22,24]. 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_1 and Xco-NP as the class of parameterized problems for which there exists an xp-reduction to UNSAT_1. (For any $L \in$ XNP, we know that L can be xp-reduced to SAT_1 by following a suitable variant of the proof of the Cook-Levin Theorem [25,26]. Conversely, any parameterized problem L that can be xp-reduced to SAT_1 we can solve in nondeterministic time $|x|^{f(k)}$ by first carrying out the xp-reduction, and then solving the resulting instance of SAT_1. The case for Xco-NP and UNSAT_1 is entirely analogous.)

2.3. Fpt-Reductions to SAT and Parameterized Complexity Classes at Higher Levels of the PH

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 (see, e.g., [27]). 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 with respect to 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 [28–31]. The classes para-NP and para-co-NP contain exactly those parameterized problems that admit such a many-one fpt-reduction to SAT_1 and UNSAT_1, 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 fixed 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 [7,8,30]. 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. The problem $\Sigma_2^{\mathrm{p}}[k*]$-$\mathrm{WSAT}(\mathcal{C})$ provides the foundation for the k-$*$ class.

$\Sigma_2^{\mathrm{p}}[k*]$-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 $\Sigma_2^{\mathrm{p}}[*k]$-$\mathrm{WSAT}(\mathcal{C})$ provides the foundation for the $*$-k hierarchy—where $\mathcal{C}$ is a class of Boolean circuits. (The parameterized problems $\Sigma_2^{\mathrm{p}}[*k]$-$\mathrm{WSAT}(\mathcal{C})$ seem not to be fpt-reducible to each other for various classes $\mathcal{C}$ of Boolean circuits—similarly to the problems p-$\mathrm{WSAT}[\mathcal{C}]$ that are

used to define the classes W[t], W[SAT], and W[P]. This is in contrast to the case of $\Sigma_2^{\mathrm{p}}[k*]$-WSAT, where we can use a Tseitin transformation [32] to reduce arbitrary Boolean circuits to equisatisfiable 3CNF formulas.)

$\Sigma_2^{\mathrm{p}}[*k]$-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 $\Sigma_2^{\mathrm{p}}[k*]$ (also called the k-$*$ class) is then defined as follows:

$$\Sigma_2^{\mathrm{p}}[k*] = \quad [\, \Sigma_2^{\mathrm{p}}[k*]\text{-WSAT} \,]^{\mathrm{fpt}}.$$

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

$$\begin{aligned} \Sigma_2^{\mathrm{p}}[*k, t] &= \quad [\, \{\, \Sigma_2^{\mathrm{p}}[*k]\text{-WSAT}(\mathrm{CIRC}_{t,u}) : u \geq 1 \,\} \,]^{\mathrm{fpt}}, \\ \Sigma_2^{\mathrm{p}}[*k, \mathrm{SAT}] &= \quad [\, \Sigma_2^{\mathrm{p}}[*k]\text{-WSAT}(\mathrm{FORM}) \,]^{\mathrm{fpt}}, \text{ and} \\ \Sigma_2^{\mathrm{p}}[*k, \mathrm{P}] &= \quad [\, \Sigma_2^{\mathrm{p}}[*k]\text{-WSAT}(\mathrm{CIRC}) \,]^{\mathrm{fpt}}. \end{aligned}$$

Note that these definitions are entirely analogous to those of the parameterized complexity classes of the W-hierarchy [20]. The following inclusion relations hold between the classes of the $*$-k hierarchy:

$$\Sigma_2^{\mathrm{p}}[*k, 1] \subseteq \Sigma_2^{\mathrm{p}}[*k, 2] \subseteq \cdots \subseteq \Sigma_2^{\mathrm{p}}[*k, \mathrm{SAT}] \subseteq \Sigma_2^{\mathrm{p}}[*k, \mathrm{P}].$$

(See also Figure 1 for a visual overview of these inclusion relations.)

Dual to the classical complexity class Σ_2^{p} is its co-class Π_2^{p}, whose canonical complete problem is complementary to the problem QSAT_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 $\Sigma_2^{\mathrm{p}}[k*]$-WSAT and $\Sigma_2^{\mathrm{p}}[*k]$-WSAT—i.e., these problems have as yes-instances exactly the no-instances of $\Sigma_2^{\mathrm{p}}[k*]$-WSAT and $\Sigma_2^{\mathrm{p}}[*k]$-WSAT, respectively. Equivalently, these complementary problems can be considered as variants of $\Sigma_2^{\mathrm{p}}[k*]$-WSAT and $\Sigma_2^{\mathrm{p}}[*k]$-WSAT where the existential and universal quantifiers are swapped, and are therefore denoted with $\Pi_2^{\mathrm{p}}[k*]$-WSAT and $\Pi_2^{\mathrm{p}}[*k]$-WSAT. We use a similar notation for the dual complexity classes, e.g., we denote co-$\Sigma_2^{\mathrm{p}}[*k, t]$ by $\Pi_2^{\mathrm{p}}[*k, t]$.

The class $\Sigma_2^{\mathrm{p}}[k*]$ 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 $\Sigma_2^{\mathrm{p}}[*k, t]$ include the the class para-NP as a subset, and is contained in the class XNP. Under some common complexity-theoretic assumptions, the class $\Sigma_2^{\mathrm{p}}[k*]$ can be separated from para-NP on the one hand, and para-Σ_2^{p} on the other hand. In particular, assuming that NP $\neq$ co-NP, it holds that $\Sigma_2^{\mathrm{p}}[k*] \not\subseteq$ para-NP, that para-NP $\not\subseteq \Sigma_2^{\mathrm{p}}[k*]$ and that $\Sigma_2^{\mathrm{p}}[k*] \subsetneq$ para-Σ_2^{p} [7,8]. Similarly, the classes $\Sigma_2^{\mathrm{p}}[*k, t]$ can be separated from para-co-NP and para-Σ_2^{p}. Assuming that NP $\neq$ co-NP, it holds that $\Sigma_2^{\mathrm{p}}[*k, 1] \not\subseteq$ para-co-NP, that para-co-NP $\not\subseteq \Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$ and thus in particular that para-co-NP $\not\subseteq \Sigma_2^{\mathrm{p}}[*k, 1]$, and that $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}] \subsetneq$ para-Σ_2^{p} [7,8].

One can also enhance the power of polynomial-time SAT encodings by considering polynomial-time algorithms that can query a SAT solver multiple times—that is, polynomial-time Turing reductions. Such an approach has been shown to be quite effective in practice (see, e.g., [33–35]) and extends the scope of SAT solvers to problems in the class Δ_2^{p}, but not to problems that are Σ_2^{p}-hard or Π_2^{p}-hard. In addition, here, switching from polynomial-time to fpt-time provides a significant increase in power. The class para-Δ_2^{p} contains all parameterized problems that can be decided by an fpt-algorithm that can query a SAT oracle multiple times—i.e., by an fpt-time Turing reduction to SAT. (One can prove this by following the proof of Theorem 4 in [24] that FPT = para-P, with the modification that the algorithms are given access to a SAT oracle.) In addition, one could restrict the number of queries that

the algorithm is allowed to make. The class para-Θ_2^p consists of all parameterized problems that can be decided by an fpt-algorithm that can query a SAT oracle 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. (This statement one can prove by following the proof of Theorem 4 in [24] that FPT = para-P, with the modification that the algorithms can query a SAT oracle an amount of times that depends logarithmically on the input size.) Restricting the number of queries even further, we define the parameterized complexity class $\text{FPT}^{\text{NP}}[\text{few}]$ as the class of all parameterized problems that can be decided by an fpt-algorithm that can query a SAT oracle at most $f(k)$ times, where k is the parameter value and f is some computable function [7,8].

We get the parameterized analogue para-PSPACE of the class PSPACE by using the definition of para-K for K = PSPACE. Similarly, we can define the parameterized complexity class XPSPACE, consisting of all parameterized problems Q for which there exists a computable function f and an algorithm A that decides whether $(x,k) \in Q$ in space $|x|^{f(k)}$. We also consider another parameterized variant of PSPACE, which is based on parameterizing the number of quantifier alternations in QSAT. An unbounded number of quantifier alternations in this problem results in the class PSPACE, and bounding the number of quantifier alternations by a constant leads to some fixed level of the PH. The parameterized complexity class PH[level] is based on bounding the number of quantifier alternations by the problem parameter [7,36]. Formally, we consider the following parameterized problem QSAT(level).

QSAT(level)
Instance: A quantified Boolean formula $\varphi = \exists X_1 \forall X_2 \exists X_3 \ldots Q_k X_k \psi$, where Q_k is a universal quantifier if k is even and an existential quantifier if k is odd, and where ψ is quantifier-free.
Parameter: k.
Question: Is φ true?

The parameterized complexity class PH[level] is defined to be the class of all parameterized problems that can be fpt-reduced to QSAT(level). We have that para-$\Sigma_2^p \cup$ para-$\Pi_2^p \subseteq$ PH[level] $\subseteq$ para-PSPACE.

An overview of the parameterized complexity classes relevant for this paper can be found in Figure 1.

In the early literature on this topic [6,28,30,37–41], the class $\Sigma_2^p[k*]$ appeared under the names $\exists^k\forall$ and $\exists^k\forall^*$. Similarly, the classes $\Sigma_2^p[*k,t]$ appeared under the names $\exists\forall^k$-W$[t]$ and $\exists^*\forall^k$-W$[t]$. In addition, the class $\text{FPT}^{\text{NP}}[\text{few}]$ appeared under the name $\text{FPT}^{\text{NP}}[f(k)]$.

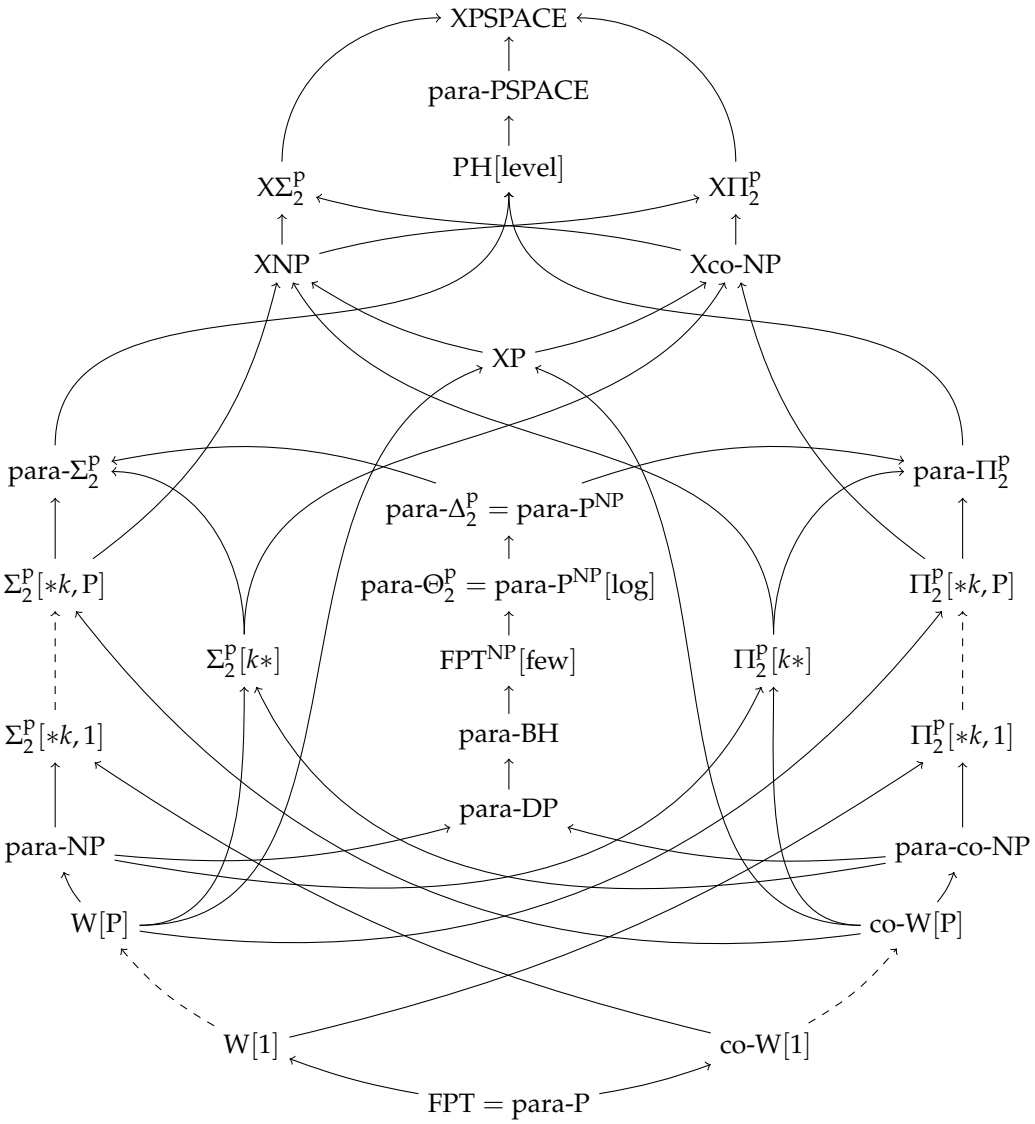

Figure 1. An overview of parameterized complexity classes up to the second level of the Polynomial Hierarchy, and higher. Dashed lines indicate a hierarchy of classes—e.g., between W[1] and W[P] lies the hierarchy $\mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots \subseteq \mathrm{W}[\mathrm{P}]$.

2.4. Treewidth and Tree Decompositions

We conclude this section with explaining the notions of tree decompositions and treewidth that we use in various places through the paper. For more details on these notions, we refer to textbooks—e.g., [19–23,42].

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 [43]. 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 $B_t = B_{t'} \setminus \{v\}$ 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 [42].

3. Propositional Logic Problems

We start with the quantified circuit satisfiability problems on which the k-$*$ and $*$-k hierarchies are based. We present 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.

$\Sigma_2^{\mathrm{p}}[k*]$-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 of weight k, such that for all truth assignments β to Y the assignment $\alpha \cup \beta$ satisfies C?
Complexity: $\Sigma_2^{\mathrm{p}}[k*]$-complete [7,8].

$\Sigma_2^{\mathrm{p}}[k*]$-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: $\Sigma_2^{\mathrm{p}}[k*]$-complete [7,8].

$\Sigma_2^{\mathrm{p}}[*k]$-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: $\Sigma_2^{\mathrm{p}}[*k, t]$-complete when restricted to circuits of weft t, for any $t \geq 1$ (by definition); $\Sigma_2^{\mathrm{p}}[*k, \mathrm{SAT}]$-complete if $\mathcal{C} = \mathrm{FORM}$ (by definition); $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$-complete if $\mathcal{C} = \mathrm{CIRC}$ (by definition).

3.1. Weighted Quantified Boolean Satisfiability for the k-$*$ Classes

Consider the following variants of $\Sigma_2^{\mathrm{p}}[k*]$-WSAT, most of which are $\Sigma_2^{\mathrm{p}}[k*]$-complete.

$\Sigma_2^{\mathrm{p}}[k*]$-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: $\Sigma_2^{\mathrm{p}}[k*]$-complete [7,8].

$\Sigma_2^{\mathrm{p}}[k*]$-WSAT$^{\leq k}$
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: $\Sigma_2^{\mathrm{p}}[k*]$-complete [7,8].

$\Sigma_2^{\mathrm{p}}[k*]$-WSAT$^{\geq k}$
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 [7].

$\Sigma_2^{\mathrm{p}}[k*]$-WSAT^{n-k}
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 $\lvert X\rvert - k$, such that $\forall Y.\psi[\alpha]$ evaluates to true?
Complexity: $\Sigma_2^{\mathrm{p}}[k*]$-complete [7].

3.2. Weighted Quantified Boolean Satisfiability in the $*$-k Hierarchy

Let $d \geq 2$ be an arbitrary constant. Then, the following problem is also $\Sigma_2^{\mathrm{p}}[*k, 1]$-complete.

$\Sigma_2^{\mathrm{p}}[*k]$-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: $\Sigma_2^{\mathrm{p}}[*k, 1]$-complete for any $d \geq 2$ [7,8].

The problem $\Sigma_2^{\mathrm{p}}[*k]$-WSAT(2-DNF) is $\Sigma_2^{\mathrm{p}}[*k, 1]$-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 [7,8].

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 $Z \setminus Y$, 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 nodes in Y as input are negation nodes, i.e., all input nodes in Y appear only negatively in the circuit. The following problems are $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$-complete.

$\Sigma_2^{\mathrm{p}}[*k]$-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: $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$-complete [7,8].

$\Sigma_2^{\mathrm{p}}[*k]$-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: $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$-complete [7].

It remains open what the exact parameterized complexity is of $\exists$-monotone and $\exists$-anti-monotone variants of $\Sigma_2^{\mathrm{p}}[*k]$-WSAT—i.e., variants of $\Sigma_2^{\mathrm{p}}[*k]$-WSAT that are based on circuits $C \in$ CIRC over two disjoint sets X and Y of variables that are in negation normal form and that are (anti-)monotone in X. The proofs used to show $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$-completeness of the $\forall$-monotone and $\forall$-anti-monotone variants [7,8] do not immediately carry over to this case.

3.3. Quantified Boolean Satisfiability with Bounded Treewidth

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

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.

QSAT_2(itw)
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 [4,5].

$\mathrm{QSAT}_2(\exists$-itw)
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 [6,7].

$\mathrm{QSAT}_2(\forall$-itw)
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 [6,7].

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-hard, it is unlikely that the parameter value can be computed in polynomial time for these problems. Alternatively, one could consider a variant of the problem where a tree decomposition of width k is given as part of the input.

3.4. Other Quantified Boolean Satisfiability

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

QSAT(#∀-vars) *Instance:* A quantified Boolean formula φ. *Parameter:* The number of universally quantified variables of φ. *Question:* Is φ true?
Complexity: para-NP-complete [6,44,45].

3.5. 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 [46].

SHORTEST-IMPLICANT-CORE(core size) *Instance:* A DNF formula φ, an implicant C of φ, and an integer k. *Parameter:* k. *Question:* Does there exist an implicant $C' \subseteq C$ of φ of size k?
Complexity: $\Sigma_2^{\mathrm{p}}[k*]$-complete [6,7].

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 exist an implicant $C' \subseteq C$ of φ of size $n - k$?
Complexity: $\Sigma_2^{\mathrm{p}}[k*]$-complete [6,7].

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: $\Sigma_2^{\mathrm{p}}[k*]$-complete [6,7].

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 $\mathrm{FPT}^{\mathrm{NP}}$[few], and in $\Sigma_2^{\mathrm{p}}[k*]$ [6,7].

3.6. 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

$\mathrm{FPT}^{\mathrm{NP}}$[few].

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: $\mathrm{FPT}^{\mathrm{NP}}$[few]-complete [7,28,37].

The above problem is used to show the following lower bound result for $\mathrm{FPT}^{\mathrm{NP}}$[few]-complete problems. No $\mathrm{FPT}^{\mathrm{NP}}$[few]-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 to the third level [18,37].

The next $\mathrm{FPT}^{\mathrm{NP}}$[few]-complete problem is based on the problem SAT-UNSAT $= \{ (\varphi_1, \varphi_2) : \varphi_1 \in \mathrm{SAT}, \varphi_2 \in \mathrm{UNSAT} \}$.

BOUNDED-SAT-UNSAT-DISJUNCTION *Instance:* A family $(\varphi_i, \varphi'_i)_{i \in [k]}$ of pairs of propositional formulas. *Parameter:* k. *Question:* Is there some $\ell \in [k]$ such that $(\varphi_\ell, \varphi'_\ell) \in$ SAT-UNSAT?
Complexity: $\mathrm{FPT}^{\mathrm{NP}}$[few]-complete [7,28,37].

3.7. Maximal Models

The following $\mathrm{FPT}^{\mathrm{NP}}$[few]-complete problems are based on various notions of (local) maximality for models of propositional formulas. Let φ be a (satisfiable) propositional formula, and let $X \subseteq \mathrm{Var}(\varphi)$ be a subset of variables. Then, a truth assignment $\alpha : \mathrm{Var}(\varphi) \to \{0,1\}$ is an *X-maximal model* of φ if (i) α satisfies φ and (ii) there is no truth assignment $\beta : \mathrm{Var}(\varphi) \to \{0,1\}$ that satisfies φ and that sets more variables among X to true than α. Moreover, take an arbitrary ordering over the variables $X \subseteq \mathrm{Var}(\varphi)$—for the sake of presentation, let $X = \{x_1, \ldots, x_k\}$ and let the ordering $<$ specify $x_1 < \cdots < x_k$. We say that a truth assignment $\alpha : \mathrm{Var}(\varphi) \to \{0,1\}$ is the *lexicographically X-maximal model* of φ (with respect to the given ordering) if (i) α satisfies φ and (ii) each truth assignment $\beta : \mathrm{Var}(\varphi) \to \{0,1\}$ that satisfies φ (with $\alpha \neq \beta$) has the property that there is some $1 \leq \ell \leq k$ such that $\alpha(x_\ell) = 1$ and $\beta(x_\ell) = 0$ and for all $1 \leq \ell' < \ell$ it holds that $\alpha(x_{\ell'}) = \beta(x_{\ell'})$.

LOCAL-MAX-MODEL *Instance:* A satisfiable propositional formula φ, a subset $X \subseteq \mathrm{Var}(\varphi)$ of variables, and a variable $w \in X$. *Parameter:* $\lvert X \rvert$. *Question:* Is there a X-maximal model of φ that sets w to true?
Complexity: $\mathrm{FPT}^{\mathrm{NP}}$[few]-complete [7,47].

LOCAL-LEX-MAX-MODEL *Instance:* A satisfiable propositional formula φ, a subset $X \subseteq \mathrm{Var}(\varphi)$ of variables, and a variable $w \in X$. *Parameter:* $\lvert X \rvert$. *Question:* Is there a lexicographically X-maximal model of φ that sets w to true?
Complexity: $\mathrm{FPT}^{\mathrm{NP}}$[few]-complete [7]. (The problem LOCAL-LEX-MAX-MODEL isn't considered explicitly in the literature. $\mathrm{FPT}^{\mathrm{NP}}$[few]-completeness for this problem follows immediately from proofs in the literature—i.e., Propositions 73 and 74 in [7]).

ODD-LOCAL-MAX-MODEL *Instance:* A propositional formula φ, and a subset $X \subseteq \mathrm{Var}(\varphi)$ of variables. *Parameter:* $\lvert X \rvert$. *Question:* Do the X-maximal models of φ set an odd number of variables in X to true?
Complexity: $\mathrm{FPT}^{\mathrm{NP}}$[few]-complete [7].

ODD-LOCAL-LEX-MAX-MODEL *Instance:* A propositional formula φ, a subset $X \subseteq \mathrm{Var}(\varphi)$ of variables, and an ordering $<$ over the variables in X. *Parameter:* $\lvert X \rvert$. *Question:* Does the lexicographically X-maximal model of φ (w.r.t. $<$) set an odd number of variables in X to true?
Complexity: $\mathrm{FPT}^{\mathrm{NP}}$[few]-complete [7].

4. Knowledge Representation and Reasoning Problems

4.1. 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. [48,49]). 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. For each such rule, $a_1 \vee \cdots \vee a_k$ is called the *head* of the rule, and $b_1, \ldots, b_m, \mathit{not}\ c_1, \ldots, \mathit{not}\ c_n$ is called the *body* of the rule. 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-normal* 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-normal if all its rules are dual-normal. We let $\mathrm{At}(P)$ denote the set of all atoms occurring in P. By *literals*, we mean atoms a or their negations $\mathit{not}\ a$. The *(Gelfond–Lifschitz) reduct* of a program P with respect to a set M of atoms, denoted P^M, is the program obtained from P by: (i) removing rules with $\mathit{not}\ a$ in the body, for each $a \in M$, and (ii) removing literals $\mathit{not}\ a$ from all other rules [50]. 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 various parameterizations of this problem. Two of these are related to atoms that must be part of any answer set of a program P. We identify a subset $\mathrm{Comp}(P)$ of *compulsory atoms*, that any answer set must include. Given a program P, we let $\mathrm{Comp}(P)$ be the smallest set such that: (i) if $(w \leftarrow \mathit{not}\ w)$ is a rule of P, then $w \in \mathrm{Comp}(P)$; and (ii) if $(b \leftarrow a_1, \ldots, a_n)$ is a rule of P, and $a_1, \ldots, a_n \in \mathrm{Comp}(P)$, then $b \in \mathrm{Comp}(P)$. We then let the set $\mathrm{Cont}(P)$ of *contingent atoms* be those atoms that occur in P but are not in $\mathrm{Comp}(P)$. We call a rule *contingent* if all the atoms that appear in the head are contingent. Another of the parameterizations that we consider is based on the notion of backdoors to normality for disjunctive logic programs. A set B of atoms is a *normality-backdoor* for a program P if deleting the atoms $b \in B$ from the rules of P results in a normal program. Deciding if a program P has a normality-backdoor of size at most k can be decided in fixed-parameter tractable time [29].

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 [7,30].

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: $\Sigma_2^{\mathrm{P}}[k*]$-complete [7,30].

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: $\Sigma_2^{\mathrm{P}}[*k, \mathrm{P}]$-complete [7,30].

ASP-CONSISTENCY(#dual-normal.rules) *Instance:* A disjunctive logic program P. *Parameter:* The number of rules of P that are dual-normal. *Question:* Does P have an answer set?
Complexity: $\Sigma_2^{\mathrm{P}}[*k, \mathrm{P}]$-complete [7].

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 [29].

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 [7,30].

4.2. Robust Constraint Satisfaction

The following problem is based on the class of robust constraint satisfaction problems introduced by Gottlob [51] and Abramsky, Gottlob and Kolaitis [52]. 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 $S = \mathrm{Var}(c)$, the *constraint scope*, is a finite sequence of distinct variables from X, and R, the *constraint relation*, is a relation over D whose arity matches the length of S, i.e., $R \subseteq D^r$, where r is the length of S.

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

Let k be a 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: $\Pi_2^p[k*]$-complete [7,30].

4.3. 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 [53].

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 [31].

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: $\Sigma_2^p[*k, 1]$-complete [7].

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 [31].

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: $\Sigma_2^p[*k, \mathrm{P}]$-complete [7].

5. Graph Problems

5.1. 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 $\Pi_2^p[*k, 1]$.

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: $\Pi_2^p[*k, 1]$-complete [7].

5.2. 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 [54].

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 $e = \{v_1, v_2\} \in E$ it 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 [54]. 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 [7,41].

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 [7,41].

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: $\Pi_2^p[k*]$-complete [7,41].

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 [7,41].

6. Other Problems

6.1. First-Order Logic Model Checking

First-order logic model checking is at the basis of a well-known hardness theory in parameterized complexity theory [22]. The following problem, also based on first-order logic model checking, offers another characterization of the parameterized complexity class $\Sigma_2^{\mathrm{p}}[k*]$. We introduce a few notions that we need for defining the model checking perspective on $\Sigma_2^{\mathrm{p}}[k*]$. 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 $\mathcal{A}$.

$\Sigma_2^{\mathrm{p}}[k*]$-MC *Instance:* A first-order logic sentence $\varphi = \exists x_1, \ldots, x_k. \forall y_1, \ldots, y_n. \psi$ 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: $\Sigma_2^{\mathrm{p}}[k*]$-complete [41].

6.2. Quantified Fagin Definability

The W-hierarchy can also be defined by means of Fagin-definable parameterized problems [22], which are based on Fagin's characterization of NP. We provide an additional characterization of the class $\Pi_2^{\mathrm{p}}[k*]$ 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 logic formulas of the form $\forall y_1, \ldots, y_n. \exists x_1, \ldots, x_m. \psi$, where ψ is quantifier-free. Let $\varphi(X)$ be a first-order logic formula over τ, with a free relation variable X with arity s. Consider the following parameterized problem.

$\Pi_2^{\mathrm{p}}[k*]$-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 $\mathcal{B}$ with weight k, there exists some relation $S \subseteq A^s$ such that $\mathcal{A} \models \varphi(S)$?
Complexity: in $\Pi_2^{\mathrm{p}}[k*]$ for each $\varphi(X)$, τ' and τ; $\Pi_2^{\mathrm{p}}[k*]$-hard for some $\varphi(X) \in \Pi_2$, τ' and τ [7].

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 logic formulas over the vocabulary τ with a free relation variable X. We then get the following characterization of $\Pi_2^{\mathrm{p}}[k*]$ [7]:

$$\Pi_2^{\mathrm{p}}[k*] = [\ \{\ \Pi_2^{\mathrm{p}}[k*]\text{-FD}_\varphi^{(\tau',\tau)} : \tau \in T, \tau' \subseteq \tau, \varphi \in \mathrm{FO}_\tau^X\ \}\ ]^{\mathrm{fpt}}.$$

Additionally, the following parameterized problem is hard for $\Pi_2^{\mathrm{p}}[*k, 1]$.

$\Pi_2^{\mathrm{p}}[*k]$-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 $\lvert S\rvert = k$ such that $\mathcal{A} \models \varphi(S)$?
Complexity: $\Pi_2^{\mathrm{p}}[*k, 1]$-hard for some $\varphi(X) \in \Pi_2$ [7].

6.3. Symbolic Model Checking for Temporal Logics

The following problems are concerned with the task of verifying whether a temporal logic formula is true in a Kripke structure. This task is of importance in the area of software and hardware verification (see, e.g., [55,56]). We consider three different temporal logics: *linear-time temporal logic* (LTL), *computation tree logic* (CTL), and CTL*, which is a superset of both LTL and CTL.

Truth of formulas φ in these logics is defined over Kripke structures $\mathcal{M}$—we write $\mathcal{M} \models \varphi$ if φ is true in $\mathcal{M}$. We consider Kripke structures that are represented symbolically using propositional formulas. Moreover, for each of the logics $\mathcal{L} \in \{\mathrm{LTL}, \mathrm{CTL}, \mathrm{CTL}^\star\}$, we consider various restricted fragments $\mathcal{L}\backslash\mathrm{X}$, $\mathcal{L}\backslash\mathrm{U}$ and $\mathcal{L}\backslash\mathrm{U,X}$, where certain operators are disallowed. For a full description of the syntax and semantics of these logics, how Kripke structures can be represented symbolically and the restricted fragments that we consider, we refer to Appendix A.

In general, the problem of deciding whether a given temporal logic formula φ is true in a given symbolically represented Kripke structure $\mathcal{M}$ is PSPACE-hard, even when restricted to formulas φ of constant size [7,36]. We consider a variant of this problem where the Kripke structures have limited recurrence diameter. The *recurrence diameter* $rd(\mathcal{M})$ of a Kripke structure $\mathcal{M}$ is the length of the longest simple (non-repeating) path in $\mathcal{M}$.

SYMBOLIC-MC$^\star[\mathcal{L}]$ *Input:* A symbolically represented Kripke structure $\mathcal{M}$, $rd(\mathcal{M})$ in unary, and an $\mathcal{L}$ formula φ. *Parameter:* $\lvert\varphi\rvert$. *Question:* $\mathcal{M} \models \varphi$?
Complexity: • para-co-NP-complete if $\mathcal{L} = \mathrm{LTL}\backslash\mathrm{U,X}$ [7,36], • PH[level]-complete if $\mathcal{L} \in \{\mathrm{CTL}, \mathrm{CTL}\backslash\mathrm{X}, \mathrm{CTL}\backslash\mathrm{U}, \mathrm{CTL}\backslash\mathrm{U,X}, \mathrm{CTL}^\star\backslash\mathrm{U,X}\}$ [7,36], and • para-PSPACE-complete if $\mathcal{L} \in \{\mathrm{CTL}^\star, \mathrm{CTL}^\star\backslash\mathrm{X}, \mathrm{CTL}^\star\backslash\mathrm{U}\}$ [7,36].

6.4. Judgment Aggregation

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 $\Phi = \{\varphi_1, \ldots, \varphi_n, \neg\varphi_1, \ldots, \neg\varphi_n\}$ 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 $F^{\text{maj}}(\boldsymbol{J}) = \{J^*\}$ 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 [57,58]. The problem of deciding whether an agenda satisfies the MP is Π_2^{p}-complete [57].

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 [7,37].

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 [7,37].

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 [7,37].

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 [7,37].

MAJ-AGENDA-SAFETY(agenda size) *Instance:* an agenda Φ. *Parameter:* $\lvert\Phi\rvert$. *Question:* Is Φ safe for the majority rule?
Complexity: FPT$^{\text{NP}}$[few]-complete [7,37].

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 fpt-time using $2^{O(k)}$ queries to an NP oracle, where $k = |\Phi|$ [7,37]. 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 [7,37].

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: $\Pi_2^p[k*]$-hard [7,37].

Let $\Phi = \{\varphi_1, \ldots, \varphi_m, \neg\varphi_1, \ldots, \neg\varphi_m\}$ 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 [7,37].

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 [7,37].

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 [7,37].

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 [7,37].

6.5. Planning

The following parameterized problems are related to planning in the face of uncertainty. We begin by describing the framework of SAS$^+$ planning (see, e.g., [59]). Let $V = \{v_1, \ldots, v_n\}$ be a finite set of *variables* over a finite *domain* D. Furthermore, let $D^+ = D \cup \{\mathbf{u}\}$, where $\mathbf{u}$ is a special *undefined* value not present in D. Then D^n is the set of *total states* and $(D^+)^n$ is the set of *partial states* over V and D. Intuitively, a state $(d_1, \ldots, d_n) \in D^n$ corresponds to an assignment that assigns to each variable $v_i \in V$ the value $d_i \in D$, and a partial state corresponds to a partial assignment that assigns a value to some variables $v_i \in V$. Clearly, $D^n \subseteq (D^+)^n$—that is, each total state is also a partial state. Let $(d_1, \ldots, d_n) = s \in (D^+)^n$ be a state. Then, the value of a variable v_i in state s is denoted by $s[v_i] = d_i$.

An *SAS$^+$ instance* is a tuple $\mathbb{P} = (V, D, A, I, G)$, where V is a set of variables, D is a domain, A is a set of *actions*, $I \in D^n$ is the *initial state* and $G \in (D^+)^n$ is the (partial) *goal state*. Each action $a \in A$ has a *precondition* $\mathsf{pre}(a) \in (D^+)^n$ and an *effect* $\mathsf{eff}(a) \in (D^+)^n$.

We will frequently use the convention that a variable has the value $\mathbf{u}$ in a precondition/effect unless a value is explicitly specified. Furthermore, by a slight abuse of notation, we denote actions and partial states such as preconditions, effects, and goals as follows. Let $a \in A$ be an action, and let $\{p_1, \ldots, p_m\} \subseteq V$ be the set of variables that are not assigned by $\mathsf{pre}(a)$ to the value $\mathbf{u}$—that is, $\{\, v \in V : \mathsf{pre}(a)[v] \neq \mathbf{u} \,\} = \{p_1, \ldots, p_m\}$. Moreover, suppose that $\mathsf{pre}(a)[p_1] = d_1, \ldots, \mathsf{pre}(a)[p_m] = d_m$. Then, we denote the precondition $\mathsf{pre}(a)$ by $\mathsf{pre}(a) = \{p_1 \mapsto d_1, \ldots, p_m \mapsto d_m\}$. In particular, if $\mathsf{pre}(a)$ is the partial state such that $\mathsf{pre}(a)[v] = \mathbf{u}$ for each $v \in V$, we denote $\mathsf{pre}(a)$ by $\emptyset$. We use a similar notation for effects. Let a be the action with $\mathsf{pre}(a) = \{p_1 \mapsto d_1, \ldots, p_m \mapsto d_m\}$ and $\mathsf{eff}(a) = \{e_1 \mapsto d'_1, \ldots, e_\ell \mapsto d'_\ell\}$. We then use the notation $a : \{p_1 \mapsto d_1, \ldots, p_m \mapsto d_m\} \to \{e_1 \mapsto d'_1, \ldots, e_{m'} \mapsto d'_{m'}\}$ to describe the action a.

Let $a \in A$ be an action and $s \in D^n$ be a state. Then, a is *valid in* s if for all $v \in V$, either $\mathsf{pre}(a)[v] = s[v]$ or $\mathsf{pre}(a)[v] = \mathbf{u}$. The *result of* a *in* s is the state $t \in D^n$ defined as follows. For all $v \in V$, $t[v] = \mathsf{eff}(a)[v]$ if $\mathsf{eff}(a)[v] \neq \mathbf{u}$ and $t[v] = s[v]$ otherwise. Let $s_0, s_\ell \in D^n$ and let $\omega = (a_1, \ldots, a_\ell)$ be a sequence of actions (of length ℓ). We say that ω is a *plan from* s_0 *to* s_ℓ if either (i) ω is the empty sequence (and $\ell = 0$, and thus $s_0 = s_\ell$), or (ii) there are states $s_1, \ldots, s_{\ell-1} \in D^n$ such that for each $i \in [\ell]$, a_i is valid in s_{i-1} and s_i is the result of a_i in s_{i-1}. A state $s \in D^n$ is a *goal state* if for all $v \in V$, either $G[v] = s[v]$ or $G[v] = \mathbf{u}$. An action sequence ω is a *plan for* $\mathbb{P}$ if ω is a plan from I to a goal state.

We also allow actions to have *conditional effects* (see, e.g., [60]). A conditional effect is of the form $s \triangleright t$, where $s, t \in (D^+)^n$ are partial states. Intuitively, such a conditional effect ensures that the variable assignment t is only applied if the condition s is satisfied. When allowing conditional effects, the effect of an action is not a partial state $\mathsf{eff}(a) \in (D^+)^n$, but a set $\mathsf{eff}(a) = \{s_1 \triangleright t_1, \ldots, s_\ell \triangleright t_\ell\}$ of conditional effects. (For the sake of simplicity, we assume that the partial states $t_1, \ldots, t_\ell$ are non-conflicting—that is, there exist no $v \in V$ and no $i_1, i_2 \in [\ell]$ with $i_1 < i_2$ such that $\mathbf{u} \neq t_{i_1}[v] \neq t_{i_2}[v] \neq \mathbf{u}$.) The result of an action a with $\mathsf{eff}(a) = \{s_1 \triangleright t_1, \ldots, s_\ell \triangleright t_\ell\}$ in a state s (in which a is valid) is the state $t \in D^n$ that is defined as follows. For all $v \in V$, $t[v] = t_i[v]$ if there exists some $i \in [\ell]$ such that s_i is satisfied in s and $t_i[v] \neq \mathbf{u}$, and $t[v] = s[v]$ otherwise.

For the parameterized planning problems that we consider, in addition to the variables V, we consider a set V_u of n' variables—constituting an uncertain planning instance $\mathbb{P} = (V, V_u, D, A, I, G)$. Intuitively, the value of the variables in V_u in the initial state is unknown. The question is whether there exists an action sequence ω such that, for each state $I_0 \in D^{n+n'}$ that extends I—that is, $I_0[v] = I[v]$ for each $v \in V$—it holds that ω is a plan for the SAS$^+$ instance $(V \cup V_u, D, A, I_0, G)$. If there exists such an action sequence ω, we say that ω is a plan that *works* for each complete initial state I_0 that extends I.

BOUNDED-UNCERTAIN-PLANNING
Instance: an uncertain planning instance $\mathbb{P} = (V, V_u, D, A, I, G)$, and an integer k.
Parameter: k.
Question: Is there a plan of length k for $\mathbb{P}$ that works for each complete initial state I_0 that extends I?
Complexity: $\Sigma_2^{\mathrm{p}}[k*]$-complete [7,39].

POLYNOMIAL-PLANNING(bounded-deviation)
Instance: an uncertain planning instance $\mathbb{P} = (V, V_u, D, A, I, G)$, an integer d, and an integer u given in unary.
Parameter: d.
Question: Is there a plan for $\mathbb{P}$ of length $\leq u$ that works for each complete initial state I_0 that extends I and for which at most d unknown variables deviate from the base value?
Complexity: $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$-complete [7,39].

6.6. 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 leaf 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 are 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 [8,41] and to the work of Flum and Grohe—Appendix A.1 in [22].

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 $\Sigma_2^{\mathrm{p}}[k*]$-complete.

$\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT*.
Instance: Positive integers $m, k, t \geq 1$, 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: $\Sigma_2^{\mathrm{p}}[k*]$-complete [7,8].

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 $\Sigma_2^{\mathrm{p}}[k*]$-complete.

$\Sigma_2^{\mathrm{P}}[k*]$-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: $\Sigma_2^{\mathrm{P}}[k*]$-complete [7,8].

In addition, the parameterized complexity class $\Sigma_2^{\mathrm{P}}[k*]$ can also be characterized by means of alternating Turing machines in the following way. Let P be a parameterized problem. An $\Sigma_2^{\mathrm{P}}[k*]$-*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* $\Sigma_2^{\mathrm{P}}[k*]$-*machine* if there exists a $\Sigma_2^{\mathrm{P}}[k*]$-machine for P. Then, $\Sigma_2^{\mathrm{P}}[k*]$ is exactly the class of parameterized decision problems that are decided by some $\Sigma_2^{\mathrm{P}}[k*]$-machine [7,8].

7. Conclusions

In this paper, we provided a list of parameterized problems that are based on problems at higher levels of the Polynomial Hierarchy, together with a complexity classification indicating whether they allow a (many-to-one or Turing) fpt-reduction to SAT or not. These complexity classifications are based in part on recently developed parameterized complexity classes—e.g., $\Sigma_2^{\mathrm{P}}[k*]$ and $\Sigma_2^{\mathrm{P}}[*k, \mathrm{P}]$ [7,8], $\mathrm{FPT}^{\mathrm{NP}}[\mathrm{few}]$ [6,7] and $\mathrm{PH}[\mathrm{level}]$ [7,36]. The problems that we considered are related to propositional logic, quantified Boolean satisfiability, disjunctive answer set programming, constraint satisfaction, (propositional) abductive reasoning, cliques, graph coloring, first-order logic model checking, temporal logic model checking, judgment aggregation, planning, and (alternating) Turing machines.

Author Contributions: Conceptualization, Methodology, Formal analysis, Investigation, Project administration, R.d.H. and S.S; Resources, Funding acquisition, S.S.; Writing—original draft preparation, R.d.H., Writing—review and editing, R.d.H. and S.S.; Supervision, S.S.

Funding: This research was funded by the European Research Council (ERC), project 239962 (COMPLEX REASON), and the Austrian Science Fund (FWF), project P26200 (Parameterized Compilation).

Acknowledgments: Open Access Funding by the Austrian Science Fund (FWF).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Appendix A

We begin by defining the syntax of the logic LTL. LTL formulas φ are formed according to the following grammar (here p ranges over a fixed set P of propositional variables), given by:

$$\varphi ::= p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \mathrm{X}\varphi \mid \mathrm{F}\varphi \mid (\varphi \mathrm{U} \varphi).$$

(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 $\mathcal{M} = (S, R, V, s_0)$, where S is a finite set of *states*, where $R \subseteq S \times S$ is a binary relation on the set of states called the *transition relation*, where $V : S \to 2^P$ is a *valuation function* that assigns each state to a set of propositions, and where $s_0 \in S$ is the *initial state*. We say that a finite sequence $s_1 \ldots s_\ell$ of states $s_i \in S$ is a *finite path* in $\mathcal{M}$ if $(s_i, s_{i+1}) \in R$ for each $i \in [\ell - 1]$. Similarly, we say that an infinite sequence $s_1 s_2 s_3 \ldots$ of states $s_i \in S$ is an *infinite path* in $\mathcal{M}$ if $(s_i, s_{i+1}) \in R$ for each $i \geq 1$.

Let $\mathcal{M} = (S, R, V, s_0)$ be a Kripke structure, and $\bar{s}_1 = s_1 s_2 s_3 \dots$ be a path in $\mathcal{M}$. Moreover, let $\bar{s}_i = s_i s_{i+1} s_{i+2} \dots$ for each $i \geq 2$. Truth of LTL formulas φ on paths $\bar{s}$ (denoted $\bar{s} \models \varphi$) is defined inductively as follows:

$$\begin{array}{lll}
\bar{s}_i \models p & \text{if} & p \in V(s_i), \\
\bar{s}_i \models \varphi_1 \wedge \varphi_2, & \text{if} & \bar{s}_i \models \varphi_1 \text{ and } \bar{s}_i \models \varphi_2, \\
\bar{s}_i \models \neg\varphi, & \text{if} & \bar{s}_i \not\models \varphi, \\
\bar{s}_i \models \mathrm{X}\varphi, & \text{if} & \bar{s}_{i+1} \models \varphi, \\
\bar{s}_i \models \mathrm{F}\varphi, & \text{if} & \text{for some } j \geq 0, \bar{s}_{i+j} \models \varphi, \\
\bar{s}_i \models \varphi_1 \mathrm{U} \varphi_2 & \text{if} & \text{there is some } j \geq 0 \text{ such that } \bar{s}_{i+j} \models \varphi_2 \text{ and } \bar{s}_{i+j'} \models \varphi \text{ for each } j' \in [0, j-1].
\end{array}$$

Then, we say that an LTL formula φ is true in the Kripke structure $\mathcal{M}$ (denoted $\mathcal{M} \models \varphi$) if for all infinite paths $\bar{s}$ starting in s_0 it holds that $\bar{s} \models \varphi$.

Next, we define the syntax of the logic CTL*, which consists of two different types of formulas: state formulas and path formulas. When we refer to CTL* formulas without specifying the type, we refer to state formulas. Given the set P of atomic propositions, the syntax of CTL* formulas is defined by the following grammar (here Φ denotes CTL* *state formulas*, φ denotes CTL* *path formulas*, and p ranges over P), given by:

$$\Phi ::= p \mid \neg\Phi \mid (\Phi \wedge \Phi) \mid \exists\varphi,$$

$$\varphi ::= \Phi \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid \mathrm{X}\varphi \mid \mathrm{F}\varphi \mid (\varphi \mathrm{U} \varphi).$$

Path formulas have the same intended meaning as LTL formulas. State formulas, in addition, allow explicit quantification over paths, which is not possible in LTL.

Formally, the semantics of CTL* formulas are defined inductively as follows. Let $\mathcal{M} = (S, R, V, s_0)$ be a Kripke structure, $s \in S$ be a state in $\mathcal{M}$ and $\bar{s}_1 = s_1 s_2 s_3 \dots$ be a path in $\mathcal{M}$. Again, let $\bar{s}_i = s_i s_{i+1} s_{i+2} \dots$ for each $i \geq 2$. The truth of CTL* state formulas Φ on states $s \in S$ (denoted $s \models \Phi$) is defined as follows:

$$\begin{array}{lll}
s \models p & \text{if} & p \in V(s), \\
s \models \Phi_1 \wedge \Phi_2, & \text{if} & s \models \Phi_1 \text{ and } s \models \Phi_2, \\
s \models \neg\Phi, & \text{if} & s \not\models \Phi, \\
s \models \exists\varphi, & \text{if} & \text{there is some path } \bar{s} \text{ in } \mathcal{M} \text{ starting in } s \text{ such that } \bar{s} \models \varphi.
\end{array}$$

The truth of CTL* path formulas φ on paths $\bar{s}$ (denoted $\bar{s} \models \varphi$) is defined as follows:

$$\begin{array}{lll}
\bar{s}_i \models \Phi & \text{if} & s_i \models \Phi, \\
\bar{s}_i \models \varphi_1 \wedge \varphi_2, & \text{if} & \bar{s}_i \models \varphi_1 \text{ and } \bar{s}_i \models \varphi_2, \\
\bar{s}_i \models \neg\varphi, & \text{if} & \bar{s}_i \not\models \varphi, \\
\bar{s}_i \models \mathrm{X}\varphi, & \text{if} & \bar{s}_{i+1} \models \varphi, \\
\bar{s}_i \models \mathrm{F}\varphi, & \text{if} & \text{for some } j \geq 0, \bar{s}_{i+j} \models \varphi, \\
\bar{s}_i \models \varphi_1 \mathrm{U} \varphi_2, & \text{if} & \text{there is some } j \geq 0 \text{ such that } \bar{s}_{i+j} \models \varphi_2 \text{ and } \bar{s}_{i+j'} \models \varphi \text{ for each } j' \in [0, j-1].
\end{array}$$

Then, we say that a CTL* formula Φ is true in the Kripke structure $\mathcal{M}$ (denoted $\mathcal{M} \models \Phi$) if $s_0 \models \Phi$.

The syntax of the logic CTL is defined similarly to the syntax of CTL*. Only the grammar for path formulas φ differs, namely:

$$\varphi ::= \mathrm{X}\Phi \mid \mathrm{F}\Phi \mid (\Phi \mathrm{U} \Phi).$$

In particular, this means that every CTL state formula, (CTL formula for short) is also a CTL* formula. The semantics for CTL formulas is defined as for their CTL* counterparts. Moreover, we say that a CTL formula Φ is true in the Kripke structure $\mathcal{M}$ (denoted $\mathcal{M} \models \Phi$) if $s_0 \models \Phi$.

For each of the logics $\mathcal{L} \in \{\mathrm{LTL}, \mathrm{CTL}, \mathrm{CTL}^\star\}$, we consider the fragments $\mathcal{L}\backslash\mathrm{X}$, $\mathcal{L}\backslash\mathrm{U}$ and $\mathcal{L}\backslash\mathrm{U,X}$. In the fragment $\mathcal{L}\backslash\mathrm{X}$, the X-operator is disallowed. Similarly, in the fragment $\mathcal{L}\backslash\mathrm{U}$, the U-operator is disallowed. In the fragment $\mathcal{L}\backslash\mathrm{U,X}$, neither the X-operator nor the U-operator is allowed.

Next, we define how Kripke structures can be represented symbolically using propositional formulas. Let $P = \{p_1, \dots, p_m\}$ be a finite set of propositional variables. A *symbolically represented Kripke*

structure over P is a tuple $\mathcal{M} = (\varphi_R, \alpha_0)$, where $\varphi_R(x_1, \ldots, x_m, x'_1, \ldots, x'_m)$ is a propositional formula over the variables $x_1, \ldots, x_m, x'_1, \ldots, x'_m$, and where $\alpha_0 \in \{0,1\}^m$ is a truth assignment to the variables in P. The Kripke structure associated with $\mathcal{M}$ is (S, R, V, α_0), where $S = \{0,1\}^m$ consists of all truth assignments to P, where $(\alpha, \alpha') \in R$ if and only if $\varphi_R[\alpha, \alpha']$ is true, and where $V(\alpha) = \{\, p_i : \alpha(p_i) = 1 \,\}$.

References

1. Vardi, M.Y. Boolean satisfiability: Theory and engineering. *Commun. ACM* **2014**, *57*, 5.
2. Stockmeyer, L.J. The polynomial-time hierarchy. *Theor. Comput. Sci.* **1976**, *3*, 1–22.
3. Wrathall, C. Complete Sets and the Polynomial-Time Hierarchy. *Theor. Comput. Sci.* **1976**, *3*, 23–33.
4. Chen, H. Quantified Constraint Satisfaction and Bounded Treewidth. In Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004), Valencia, Spain, 22–27 August 2004; pp. 161–165.
5. Feder, T.; Kolaitis, P.G. Closures and dichotomies for quantified constraints. In *Electronic Colloquium on Computational Complexity (ECCC)*; Technical Report TR06-160; Weizmann Institute of Science: Rehovot, Israel, 2006.
6. De Haan, R.; Szeider, S. Fixed-parameter tractable reductions to SAT. In Proceedings of the 17th International Symposium on the Theory and Applications of Satisfiability Testing (SAT 2014), Vienna, Austria, 14–17 July 2014; Egly, U., Sinz, C., Eds.; Springer: Berlin, Germany, 2014; Volume 8561, pp. 85–102.
7. De Haan, R. Parameterized Complexity in the Polynomial Hierarchy. Ph.D. Thesis, Technische Universität Wien, Vienna, Austria, 2016.
8. De Haan, R.; Szeider, S. Parameterized complexity classes beyond para-NP. *J. Comput. Syst. Sci.* **2017**, *87*, 16–57.
9. Schaefer, M.; Umans, C. Completeness in the Polynomial-Time hierarchy: A Compendium. *SIGACT News* **2002**, *33*, 32–49.
10. Cesati, M. Compendium of Parameterized Problems. Available online: http://cesati.sprg.uniroma2.it/research/compendium/ (accessed on 4 September 2019).
11. Arora, S.; Barak, B. *Computational Complexity—A Modern Approach*; Cambridge University Press: Cambridge, UK, 2009; pp. I–XXIV, 1–579.
12. Papadimitriou, C.H. *Computational Complexity*; Addison-Wesley: Boston, MA, USA, 1994.
13. Meyer, A.R.; Stockmeyer, L.J. The Equivalence Problem for Regular Expressions with Squaring Requires Exponential Space. In Proceedings of the 13th Annual Symposium on Switching & Automata Theory (SWAT), College Park, MD, USA, 25–27 October 1972; pp. 125–129.
14. Hemachandra, L.A. The strong exponential hierarchy collapses. *J. Comput. Syst. Sci.* **1989**, *39*, 299–322.
15. Buss, S.R.; Hay, L. On truth-table reducibility to SAT. *Inf. Comput.* **1991**, *91*, 86–102.
16. Cai, J.; Gundermann, T.; Hartmanis, J.; Hemachandra, L.A.; Sewelson, V.; Wagner, K.W.; Wechsung, G. The Boolean Hierarchy I: Structural Properties. *SIAM J. Comput.* **1988**, *17*, 1232–1252.
17. Chang, R.; Kadin, J. The Boolean Hierarchy and the Polynomial Hierarchy: A Closer Connection. *SIAM J. Comput.* **1993**, *25*, 169–178.
18. Kadin, J. The Polynomial Time Hierarchy Collapses if the Boolean Hierarchy Collapses. *SIAM J. Comput.* **1988**, *17*, 1263–1282.
19. Cygan, M.; Fomin, F.V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; Saurabh, S. *Parameterized Algorithms*; Springer: Berlin, Germany, 2015.
20. Downey, R.G.; Fellows, M.R. Parameterized Complexity. In *Monographs in Computer Science*; Springer: New York, NY, USA, 1999.
21. Downey, R.G.; Fellows, M.R. Texts in Computer Science. *Fundamentals of Parameterized Complexity*; Springer: Berlin, Germany, 2013.
22. Flum, J.; Grohe, M. Parameterized Complexity Theory. In *Texts in Theoretical Computer Science. An EATCS Series*; Springer: Berlin, Germany, 2006; Volume XIV.
23. Niedermeier, R. Invitation to Fixed-Parameter Algorithms. In *Oxford Lecture Series in Mathematics and Its Applications*; Oxford University Press: Oxford, UK, 2006.
24. Flum, J.; Grohe, M. Describing parameterized complexity classes. *Inf. Comput.* **2003**, *187*, 291–319.
25. Cook, S.A. The Complexity of Theorem-Proving Procedures. In Proceedings of the Third Annual ACM Symposium on Theory of Computing, Shaker Heights, OH, USA, 3–5 May 1971; pp. 151–158.

26. Levin, L. Universal sequential search problems. *Probl. Inf. Transm.* **1973**, *9*, 265–266.
27. Prestwich, S.D. CNF Encodings. In *Handbook of Satisfiability*; Biere, A., Heule, M., van Maaren, H., Walsh, T., Eds.; IOS Press: Amsterdam, The Netherlands, 2009; pp. 75–97.
28. Endriss, U.; De Haan, R.; Szeider, S. Parameterized Complexity Results for Agenda Safety in Judgment Aggregation. In Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Istanbul, Turkey, 4–8 May 2015.
29. Fichte, J.K.; Szeider, S. Backdoors to Normality for Disjunctive Logic Programs. *ACM Trans. Comput. Log.* **2015**, *17*, 7:1–7:23.
30. De Haan, R.; Szeider, S. The Parameterized Complexity of Reasoning Problems Beyond NP. In Proceedings of the Fourteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2014), Vienna, Austria, 20–24 July 2014; Baral, C., De Giacomo, G., Eiter, T., Eds.; AAAI Press: Palo Alto, CA, USA, 2014.
31. Pfandler, A.; Rümmele, S.; Szeider, S. Backdoors to Abduction. In Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013), Beijing, China, 3–9 August 2013; Rossi, F., Ed.; AAAI Press: Palo Alto, CA, USA, 2013.
32. Tseitin, G.S. Complexity of a Derivation in the Propositional Calculus. *Zap. Nauchn. Sem. Leningrad Otd. Mat. Inst. Akad. Nauk SSSR* **1968**, *8*, 23–41.
33. Belov, A.; Lynce, I.; Marques-Silva, J. Towards efficient MUS extraction. *AI Commun.* **2012**, *25*, 97–116.
34. Dvořák, W.; Järvisalo, M.; Wallner, J.P.; Woltran, S. Complexity-sensitive decision procedures for abstract argumentation. *Artif. Intell.* **2014**, *206*, 53–78.
35. Marques-Silva, J.; Janota, M.; Belov, A. Minimal Sets over Monotone Predicates in Boolean Formulae. In Proceedings of the 25th International Conference Computer Aided Verification (CAV 2013), Saint Petersburg, Russia, 13–19 July 2013; Sharygina, N., Veith, H., Eds.; Springer: Berlin, Germany, 2013; Volume 8044, pp. 592–607.
36. De Haan, R.; Szeider, S. Parameterized Complexity Results for Symbolic Model Checking of Temporal Logics. In Proceedings of the Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning (KR 2016), Cape Town, South Africa, 25–29 April 2016; pp. 453–462.
37. Endriss, U.; De Haan, R.; Szeider, S. Parameterized Complexity Results for Agenda Safety in Judgment Aggregation. In Proceedings of the 5th International Workshop on Computational Social Choice (COMSOC-2014), Istanbul, Turkey, 4–8 May 2014.
38. De Haan, R. An Overview of Non-Uniform Parameterized Complexity. In *Electronic Colloquium on Computational Complexity (ECCC)*; Technical Report TR15-130; Weizmann Institute of Science: Rehovot, Israel, 2015.
39. De Haan, R.; Kronegger, M.; Pfandler, A. Fixed-parameter Tractable Reductions to SAT for Planning. In Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, 25–31 July 2015.
40. De Haan, R.; Szeider, S. Compendium of Parameterized Problems at Higher Levels of the Polynomial Hierarchy. In *Electronic Colloquium on Computational Complexity (ECCC)*; Technical Report TR14–143; Weizmann Institute of Science: Rehovot, Israel, 2014.
41. De Haan, R.; Szeider, S. 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), Pec pod Sněžkou, Czech Republic, 24–29 January 2015; Springer: Berlin, Germany, 2015; Volume 8939.
42. Kloks, T. *Treewidth: Computations and Approximations*; Springer: Berlin, Germany, 1994.
43. Bodlaender, H.L. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.* **1996**, *25*, 1305–1317.
44. Ayari, A.; Basin, D.A. QUBOS: Deciding Quantified Boolean Logic Using Propositional Satisfiability Solvers. In Proceedings of the 4th International Conference on Formal Methods in Computer-Aided Design (FMCAD 2002), Portland, OR, USA, 6–8 November 2002; Aagaard, M., O'Leary, J.W., Eds.; Springer: Berlin, Germany, 2002; Volume 2517, pp. 187–201.
45. Biere, A. Resolve and Expand. In Proceedings of the Seventh International Conference on Theory and Applications of Satisfiability Testing (SAT 2004), Vancouver, BC, Canada, 10–13 May 2004; pp. 59–70.

46. Umans, C. Approximability and Completeness in the Polynomial Hierarchy. Ph.D. Thesis, University of California, Berkeley, CA, USA, 2000.
47. De Haan, R. Parameterized Complexity Results for the Kemeny Rule in Judgment Aggregation. In Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI 2016), The Hague, The Netherlands, 29 August–2 September 2016; Kaminka, G.A., Fox, M., Bouquet, P., Hüllermeier, E., Dignum, V., Dignum, F., van Harmelen, F., Eds.; IOS Press: Amsterdam, The Netherlands, 2016; Volume 285, pp. 1502–1510.
48. Brewka, G.; Eiter, T.; Truszczynski, M. Answer set programming at a glance. *Commun. ACM* **2011**, *54*, 92–103.
49. Marek, V.W.; Truszczynski, M. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: A 25-Year Perspective*; Springer: Berlin, Germany, 1999; pp. 169–181.
50. Gelfond, M.; Lifschitz, V. Classical Negation in Logic Programs and Disjunctive Databases. *New Gener. Comput.* **1991**, *9*, 365–386.
51. Gottlob, G. On minimal constraint networks. *Artif. Intell.* **2012**, *191–192*, 42–60.
52. Abramsky, S.; Gottlob, G.; Kolaitis, P.G. Robust Constraint Satisfaction and Local Hidden Variables in Quantum Mechanics. In Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013), Beijing, China, 3–9 August 2013; Rossi, F., Ed.; AAAI Press: Palo Alto, CA, USA, 2013.
53. Eiter, T.; Gottlob, G. The complexity of logic-based abduction. *J. ACM* **1995**, *42*, 3–42.
54. Ajtai, M.; Fagin, R.; Stockmeyer, L.J. The Closure of Monadic NP. *J. Comput. Syst. Sci.* **2000**, *60*, 660–716.
55. Baier, C.; Katoen, J.P. *Principles of Model Checking*; MIT Press: Cambridge, MA, USA, 2008.
56. Clarke, E.M.; Grumberg, O.; Peled, D.A. *Model Checking*; MIT Press: Cambridge, MA, USA, 1999.
57. Endriss, U.; Grandi, U.; Porello, D. Complexity of Judgment Aggregation. *J. Artif. Intell. Res.* **2012**, *45*, 481–514.
58. Nehring, K.; Puppe, C. The structure of strategy-proof social choice—Part I: General characterization and possibility results on median spaces. *J. Econ. Theory* **2007**, *135*, 269–305.
59. Bäckström, C.; Nebel, B. Complexity Results for SAS+ Planning. *Comput. Intell.* **1995**, *11*, 625–656.
60. Pednault, E.P.D. ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus. In Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning (KR 1989), Toronto, ON, Canada, 15–18 May 1989; pp. 324–332.