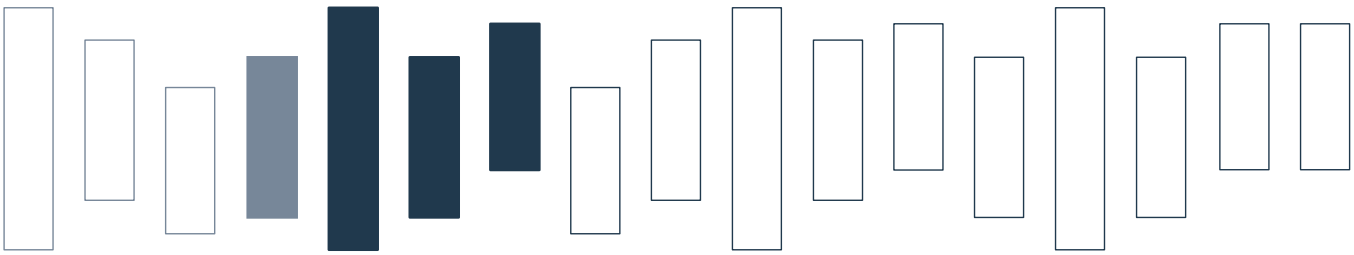
Technical Report AC-TR-17-004

January 2017

Parameterized Complexity Classes Beyond Para-NP

Ronald de Haan, and Stefan Szeider



This is the authors' copy of a paper that appeared in the Journal of Computer and System Sciences, 2017.

www.ac.tuwien.ac.at/tr

Parameterized Complexity Classes Beyond Para-NP*

Ronald de Haan, Stefan Szeider

Algorithms and Complexity Group
TU Wien, Vienna, Austria

[dehaan,sz]@ac.tuwien.ac.at

Abstract

Today's propositional satisfiability (SAT) solvers are extremely powerful and can be used as an efficient back-end for solving NP-complete problems. However, many fundamental problems in logic, in knowledge representation and reasoning, and in artificial intelligence are located at the second level of the Polynomial Hierarchy or even higher, and hence for these problems polynomial-time transformations to SAT are not possible, unless the hierarchy collapses. Recent research shows that in certain cases one can break through these complexity barriers by fixed-parameter tractable (fpt) reductions to SAT which exploit structural aspects of problem instances in terms of problem parameters. These reductions are more powerful because their running times can grow superpolynomially in the problem parameters. In this paper we develop a general theoretical framework that supports the classification of parameterized problems on whether they admit such an fpt-reduction to SAT or not.

Contents

*This paper contains results that have appeared in shortened and preliminary form in the proceedings of SAT 2014 [31], the proceedings of KR 2014 [32], and the proceedings of SOFSEM 2015 [34].

1 Introduction

Over the last two decades, propositional satisfiability (SAT) has become one of the most successful and widely applied techniques for the solution of NP-complete problems. Today's SAT solvers are extremely efficient and robust. Instances with hundreds of thousands of variables and clauses can be solved routinely. In fact, due to the success of SAT, NP-complete problems have lost their scariness, as in many cases one can efficiently encode NP-complete problems to SAT and solve them by means of a SAT solver [6, 24, 39, 41, 47]. However, many important computational problems in artificial intelligence and knowledge representation and reasoning are located above the first level of the Polynomial Hierarchy (PH) and thus considered "harder" than SAT. Hence we cannot hope for polynomial-time reductions from these problems to SAT, as such transformations would cause the (unexpected) collapse of the PH.

Realistic problem instances are not random and often contain some kind of "hidden structure." Recent research succeeded in exploiting such hidden structure to break the complexity barriers between levels of the PH for problems that arise in disjunctive answer set programming [17] and abductive reasoning [46]. The idea is to exploit problem structure in terms of a problem *parameter*, and to develop reductions to SAT that can be computed efficiently as long as the problem parameter is reasonably small. The theory of *parameterized complexity* [8, 10, 11, 19, 44] provides exactly the right type of reduction suitable for this purpose, called *fixed-parameter tractable reductions*, or *fpt-reductions* for short. Now, for a suitable choice of the parameter, one can aim at developing fpt-reductions from the hard problem under consideration to SAT.

Such positive results go significantly beyond the state-of-the-art of current research in parameterized complexity. By shifting the scope from fixed-parameter tractability to fpt-reducibility (to SAT), positive results can be obtained with less restrictive parameters and hence such positive results apply to larger classes of inputs. In fact, there are some known reductions that, in retrospect, can be seen as fpt-reductions to SAT. A prominent example is the technique of bounded model checking [5], which can be employed as an fpt-reduction to SAT from the PSPACE-complete problem of model checking linear temporal logic (LTL) formulas on symbolically represented Kripke structures, where the parameter is the size of the logic formula [35, 38]. Bounded model checking is widely used for hardware and software verification at industrial scale [4].

When studying a problem, together with a choice of the parameter, one can readily use known concepts and techniques from parameterized complexity theory to devise fpt-reductions to SAT. However, evidently,

not in all cases one can establish an fpt-reduction to SAT. In order to adequately analyze, for concrete problems, what choices of the parameter admit fpt-reductions to SAT, one also needs methods to show that in certain cases fpt-reductions to SAT are not possible. Such tools are lacking in the parameterized complexity literature.

Contributions The aim of this paper is to establish a general theoretical framework that supports the classification of parameterized variants of problems at higher levels of the PH on whether they admit an fpt-reduction to SAT or not.

- We develop a new hardness theory that can be used to provide evidence that certain parameterized problems do not admit an fpt-reduction to SAT.

The notion of hardness offered by this new theory is similar to the concepts of NP-hardness which provides evidence against polynomial-time solvability [20] and W[1]-hardness which provides evidence against fixed-parameter tractability [10].

At the center of our theory are two hierarchies of parameterized complexity classes: the $*$-k hierarchy and the k-$*$ hierarchy. (These hierarchies contain different levels that are similar to the levels of the W-hierarchy.) We define the complexity classes in terms of weighted variants of the quantified Boolean satisfiability problem with one quantifier alternation, which is canonical for the second level of the PH. For the classes in the k-$*$ hierarchy, the (Hamming) weight of the assignment to the variables in the first quantifier block is bounded by the parameter k, the weight of the second quantifier block is unrestricted ("$*$"). For the classes in the $*$-k hierarchy it is the other way around, the weight of assignments to the second block is restricted by k and the first block is unrestricted. Both hierarchies span various degrees of hardness between the classes para-NP and para-co-NP at the bottom and the classes para-Σ_2^{P} and para-Π_2^{P} at the top—para-K contains all parameterized problems that, after fpt-time preprocessing, belong to the complexity class K [18]. We show that the parameterized complexity classes of the k-$*$ hierarchy in fact collapse into a single class $\Sigma_2^{\mathrm{p}}[k*]$, whereas the $*$-k hierarchy seems to be a proper hierarchy $\Sigma_2^{\mathrm{p}}[*k,t] \subseteq \Sigma_2^{\mathrm{p}}[*k,t+1]$, for each t (similarly to the W-hierarchy). We use the notation $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k,t]$ to reflect the way in which these classes are based on weighted variants of canonical problems for the class Σ_2^{P}.[1]

We begin by providing several basic structural results that help to enable the use of the new classes for a parameterized complexity analysis of concrete problems.

- We show that the parameterized complexity classes of the k-$*$ hierarchy collapse into a single class $\Sigma_2^{\mathrm{p}}[k*]$.
- We provide normalization results for the classes $\Sigma_2^{\mathrm{p}}[*k,1]$ and $\Sigma_2^{\mathrm{p}}[*k,\mathrm{P}]$, that show that the canonical problems for these classes are hard even when restricted to instances that are in a normal form—for $\Sigma_2^{\mathrm{p}}[*k,1]$, the normal form consists of 2DNF formulas, and for $\Sigma_2^{\mathrm{p}}[*k,\mathrm{P}]$, the normal form consists of circuits in negation normal form where the universally quantified variables occur only positively.

Moreover, we illustrate how these normalization results are helpful by pointing out several cases where these results are used to establish hardness for the newly developed parameterized complexity classes.

Then, we demonstrate the robustness of the new theory by showing that the new parameterized complexity classes can be characterized in terms of several other fundamental concepts from the domain of theoretical computer science.

- We show that the class $\Sigma_2^{\mathrm{p}}[k*]$ can be characterized as those parameterized problems that are fpt-reducible to a natural parameterized variant of first-order logic model checking.
- We show that the class $\Sigma_2^{\mathrm{p}}[k*]$ can also be characterized in several ways using alternating Turing machines (ATMs) with appropriate bounds on the number of alternations and the number of nondeterministic steps.

[1] In previous work [14, 15, 16, 27, 28, 30, 31, 32, 33, 34] the class $\Sigma_2^{\mathrm{p}}[k*]$ appeared under the names $\exists^k\forall$ and $\exists^k\forall^*$. Similarly, the classes $\Sigma_2^{\mathrm{p}}[*k,t]$ appeared under the names $\exists\forall^k$-W$[t]$ and $\exists^*\forall^k$-W$[t]$.

We give evidence that the parameterized complexity classes that we introduce are different from classes that are known from the literature, and that they are distinct from each other. We establish separation results that are based on various complexity-theoretic assumptions.

- Assuming that NP $\neq$ co-NP, we show that $\Sigma_2^{\mathrm{p}}[k*]$ is different from para-NP, para-Σ_2^{P} and para-Π_2^{P}, and that the classes $\Sigma_2^{\mathrm{p}}[*k, t]$ are different from para-co-NP, para-Σ_2^{P} and para-Π_2^{P}.
- We show that $\Sigma_2^{\mathrm{p}}[k*]$ is different from para-co-NP and that the classes $\Sigma_2^{\mathrm{p}}[*k, t]$ are different from para-NP, unless there is a subexponential-time reduction from the Σ_2^{P}-complete problem QSAT_2(3DNF) to SAT or UNSAT.
- Using a combination of these complexity-theoretic assumptions, we show that the class $\Sigma_2^{\mathrm{p}}[k*]$ (and its co-class $\Pi_2^{\mathrm{p}}[k*]$), on the one hand, and the classes $\Sigma_2^{\mathrm{p}}[*k, t]$ (and their co-classes $\Pi_2^{\mathrm{p}}[*k, t]$), on the other hand, are distinct.

Finally, we substantiate the usefulness of the new theory by indicating how the parameterized complexity classes that we introduced can be employed in the computational complexity analysis of many natural parameterized problems.

- We illustrate the crucial role of the classes $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k, t]$ in the parameterized complexity analysis of problems at higher levels of the PH by conducting a case study.
- We present completeness results for parameterized problems from a wide range of areas for the various parameterized complexity classes that we developed.

The Potential of Fpt-Reductions to SAT Due to the spectacular performance of modern SAT solvers in practice, fixed-parameter tractable reductions to SAT offer the possibility of algorithmic methods that could perform well in many cases in practice. Because of this, one could optimistically view fpt-reductions to SAT as "tractability" results, especially for problems whose complexity lies at the second level of the PH or higher. However, it should be pointed out that the notion of fixed-parameter tractability offers much stronger promises for the efficiency of algorithms than fpt-reductions to SAT. In particular, fixed-parameter tractability offers worst-case running time guarantees, whereas algorithms based on fpt-reductions to SAT rely on the performance of SAT solvers (for which we have no good worst-case running time guarantees).

Instead of considering fpt-reductions to SAT as unreserved tractability results, one should regard them as promising results that could serve as a theoretical starting point for engineering efforts that might lead to productive algorithmic methods that are based on the combination of fixed-parameter tractable algorithms and optimized SAT solving methodology. Such theoretical results have particular potential in cases where the parameters that have small values in the application domain at hand are too restrictive to lead to fixed-parameter tractability results. In other words, fpt-reductions to SAT are best viewed as a type of positive results that is complementary to the traditional notion of fixed-parameter tractability.

Similar arguments as the ones we discussed above could be put forward to argue for the merits of fpt-reductions to problems at higher levels of the PH or even to PSPACE-complete problems. In fact, there are off-the-shelf solvers available also for such problems. For example, for the PSPACE-complete problem of QBF satisfiability there are many solving algorithms available, that work reasonably well in many settings. For problems that are beyond PSPACE (e.g., EXPTIME-hard problems) it does make sense to consider the possibility of fpt-reductions to problems in PSPACE. However, available algorithms for problems at higher levels of the PH or PSPACE-complete problems do not perform nearly as spectacularly well in practice as SAT solvers. As a result, fpt-reductions to the former type of problems are much less promising for the development of practically efficient algorithmic methods. For this reason, in this paper, we focus our attention on the development of theoretical tools that are beneficial for the analysis of parameterized problems on whether they admit fpt-reductions to SAT.

Roadmap We begin in Section 2 by reviewing basic notions from complexity theory and parameterized complexity theory. In Section 3, we define the parameterized complexity classes of the k-$*$ and $*$-k hierarchies, and we establish basic structural results for these classes. Then, in Section 4, we show how the new parameterized complexity classes can be characterized using first-order logic model checking and alternating Turing machines. In Section 5, we provide evidence that the new classes are different from each other and from other parameterized complexity classes known from the literature. In Section 6, we illustrate how the parameterized complexity classes of the k-$*$ and $*$-k hierarchies are useful for the complexity analysis of parameterized variants of problems at higher levels of the PH, and we present completeness results for the new classes for a wide range of natural problems. Finally, in Section 7, we conclude and suggest directions for future research.

2 Preliminaries

In this section, we review some notions from complexity theory and parameterized complexity theory. We expect the reader to be familiar with the basics of computational complexity theory (such as the notion of decision problems, the complexity classes P and NP, and the concept of NP-completeness). For more details, we refer to textbooks on the topic [3, 22, 23, 45, 51]. We firstly survey the definition of and some foundational results related to the Polynomial Hierarchy. Secondly, we consider some concepts and definitions from the theory of parameterized complexity that we use in the remainder of the paper.

2.1 The Polynomial Hierarchy (PH)

The *Polynomial Hierarchy (PH)* [42, 45, 52, 55] contains a hierarchy of complexity classes $\Sigma_i^{\mathrm{P}} \subseteq \Sigma_{i+1}^{\mathrm{P}}$, for all $i \geq 0$. We give a characterization of these classes based on the satisfiability problem of various classes of quantified Boolean formulas. A *quantified Boolean formula* is a formula of the form $Q_1 X_1 Q_2 X_2 \dots Q_m X_m . \psi$, where each Q_i is either $\forall$ or $\exists$, the X_i are disjoint sets of propositional variables, and ψ is a Boolean formula over the variables in $\bigcup_{i=1}^{m} X_i$. The quantifier-free part of such formulas is called the *matrix* of the formula. Truth of such formulas is defined in the usual way. Let $\gamma = \{x_1 \mapsto d_1, \dots, x_n \mapsto d_n\}$ be a function that maps some variables $x_1, \dots, x_n$ of a formula φ to other variables or to truth values. We let $\varphi[\gamma]$ denote the application of such a substitution γ to the formula φ. We also write $\varphi[x_1 \mapsto d_1, \dots, x_n \mapsto d_n]$ to denote $\varphi[\gamma]$. For each $i \geq 1$ we define the following decision problem.

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

Input formulas to the problem QSAT_i are called Σ_i^{P}-formulas. For each nonnegative integer $i \leq 0$, the complexity class Σ_i^{P} can be characterized as the closure of the problem QSAT_i under polynomial-time reductions [52, 55]. 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}.

The classes Σ_i^{P} and Π_i^{P} can also be defined by means of nondeterministic Turing machines with an oracle. For any complexity class C, we let NP^C be the set of decision problems that is decidable in polynomial time by a nondeterministic Turing machine with an oracle for a problem that is complete for the class C. Then, the classes Σ_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 letting $\Sigma_i^{\mathrm{P}} = \mathrm{NP}^{\Sigma_{i-1}^{\mathrm{P}}}$ and $\Pi_i^{\mathrm{P}} = \text{co-NP}^{\Sigma_{i-1}^{\mathrm{P}}}$ for each $i \geq 1$.

Alternating Turing Machines The classes of the PH can also be characterized using *alternating Turing machines*. We use the same notation as Flum and Grohe [19, Appendix A.1].

Let $m \geq 1$ be a positive integer. An *alternating Turing machine (ATM)* with m tapes is a 6-tuple $\mathbb{M} = (S_\exists, S_\forall, \Sigma, \Delta, s_0, F)$, where:

- $S_\exists$ and $S_\forall$ are disjoint sets;
- $S = S_\exists \cup S_\forall$ is the finite, non-empty set of *states*;
- Σ is the finite, non-empty *alphabet*;
- $s_0 \in S$ is the *initial state*;
- $F \subseteq S$ is the set of *accepting states*;
- $\Delta \subseteq S \times (\Sigma \cup \{\$, \square\})^m \times S \times (\Sigma \cup \{\$\})^m \times \{\mathbf{L}, \mathbf{R}, \mathbf{S}\}^m$ is the *transition relation*. The elements of Δ are the *transitions*.
- $\$, \square \notin \Sigma$ are special symbols. "$" marks the left end of each tape. It cannot be overwritten and only allows **R**-transitions. "$\square$" is the *blank symbol*.

Intuitively, the tapes of our machine are bounded to the left and unbounded to the right. The leftmost cell, the 0-th cell, of each tape carries a "$", and initially, all other tape cells carry the blank symbol. The input is written on the first tape, starting with the first cell, the cell immediately to the right of the "$".

A *configuration* is a tuple $C = (s, x_1, p_1, \ldots, x_m, p_m)$, where $s \in S$, $x_i \in \Sigma^*$, and $0 \leq p_i \leq |x_i| + 1$ for each $1 \leq i \leq m$. Intuitively, $\$x_i\square\square\ldots$ is the sequence of symbols in the cells of tape i, and the head of tape i scans the p_i-th cell. The *initial configuration* for an input $x \in \Sigma^*$ is $C_0(x) = (s_0, x, 1, \epsilon, 1, \ldots, \epsilon, 1)$, where ϵ denotes the empty word.

A *computation step* of $\mathbb{M}$ is a pair (C, C') of configurations such that there is a transition transformating C into C'. Intuitively, a tuple $(s, (a_1, \ldots, a_m), s', (d_1, \ldots, d_m)) \in \Delta$ encodes that it is possible to go from one configuration with state s where the head of tape i scans symbol a_i, for each $1 \leq i \leq m$, into another configuration that differs only from the first configuration in that (1) the new configuration has state s' and (2) the head of each tape i moves according to $d_i \in \{\mathbf{L}, \mathbf{R}, \mathbf{S}\}$—here **L** represent a move of one cell to the left, **R** represents a move of one cell to the right, and **S** represents the head staying at the same location. We omit the formal details. We write $C \to C'$ to denote that (C, C') is a computation step of $\mathbb{M}$. If $C \to C'$, we call C' a *successor configuration* of C. A *halting configuration* is a configuration that has no successor configuration. A halting configuration is *accepting* if its state is in F. A configuration is called *existential* if it is not a halting configuration and its state is in $S_\exists$, and *universal* if it is not a halting configuration and its state is in $S_\forall$. A step $C \to C'$ is *nondeterministic* if there is a configuration $C'' \neq C'$ such that $C \to C''$, and is *existential* if C is an existential configuration. A state $s \in S$ is called *deterministic* if for any $a_1, \ldots, a_m \in \Sigma \cup \{\$, \square\}$, there is at most one $(s, (a_1, \ldots, a_m), s', (a'_1, \ldots, a'_m), (d_1, \ldots, d_m)) \in \Delta$. Similarly, we call a non-halting configuration *deterministic* if its state is deterministic, and *nondeterministic* otherwise.

Intuitively, in an existential configuration, there must be one possible run that leads to acceptance, whereas in a universal configuration, all runs must lead to acceptance. Formally, a *run* of an ATM $\mathbb{M}$ is a directed tree where each node is labeled with a configuration of $\mathbb{M}$ such that:

- The root is labeled with an initial configuration.
- If a vertex is labeled with an existential configuration C, then the vertex has precisely one child that is labeled with a successor configuration of C.
- If a vertex is labeled with a universal configuration C, then for every successor configuration C' of C the vertex has a child that is labeled with C'.

We often identify nodes of the tree with the configurations with which they are labeled. The run is *finite* if the tree is finite, and *infinite* otherwise. The *length* of the run is the height of the tree. The run is *accepting* if it

is finite and every leaf is labeled with an accepting configuration. If the root of a run ρ is labeled with $C_0(x)$, then ρ is a run *with input* x. Any path from the root of a run ρ to a leaf of ρ is called a *computation path*.

The *language (or problem) accepted by* $\mathbb{M}$ is the set $Q_{\mathbb{M}}$ of all $x \in \Sigma^*$ such that there is an accepting run of $\mathbb{M}$ with initial configuration $C_0(x)$. $\mathbb{M}$ *runs in time* $t : \mathbb{N} \to \mathbb{N}$ if for every $x \in \Sigma^*$ the length of every run of $\mathbb{M}$ with input x is at most $t(|x|)$.

A *step* $C \to C'$ is an *alternation* if either C is existential and C' is universal, or vice versa. A run ρ of $\mathbb{M}$ is *ℓ-alternating* for an $\ell \in \mathbb{N}$, if on every path in the tree associated with ρ, there are less than ℓ alternations between existential and universal configurations. The machine $\mathbb{M}$ is *ℓ-alternating* if every run of $\mathbb{M}$ is ℓ-alternating.

The classes Σ_i^{P} can be characterized using ATMs as follows. Let $i \geq 1$. The class Σ_i^{P} consists of all problems that are decidable by an i-alternating polynomial-time ATM $\mathbb{M} = (S_\exists, S_\forall, \Sigma, \Delta, s_0, F)$ such that $s_0 \in S_\exists$.

2.2 Parameterized Complexity Theory

We introduce some core notions from parameterized complexity theory. For an in-depth treatment we refer to textbooks [8, 10, 11, 19, 44]. 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*. For each positive integer $k \geq 1$, we define the k-th slice of L as the unparameterized problem $L_k = \{\, x : (x, k) \in L \,\}$. The following generalization of polynomial time computability is commonly regarded as the tractability notion of parameterized complexity theory. A parameterized problem L is *fixed-parameter tractable* if there exists a computable function f and a constant c such that there exists an algorithm that decides whether $(x, k) \in L$ in time $f(k)|x|^c$, 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 fixed-parameter tractable decision problems. If the parameter is constant, then fpt-algorithms run in polynomial time where the order of the polynomial is independent of the parameter. This provides a good scalability in the parameter, in contrast to running times of the form $|x|^k$, which are also polynomial for fixed k, but can already be impractical for, say, $k > 3$. The class of all parameterized problems that can be solved in time $O(|x|^{f(k)})$, for some computable function f, is denoted by XP.

Parameterized Intractability Parameterized complexity also offers a hardness theory, similar to the theory of NP-hardness, which allows researchers to give strong theoretical evidence that some parameterized problems are not fixed-parameter tractable. Central to this hardness theory is the *W-hierarchy* of complexity classes $\mathrm{FPT} \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots \subseteq \mathrm{W}[\mathrm{SAT}] \subseteq \mathrm{W}[\mathrm{P}] \subseteq \mathrm{XP}$, where all inclusions are believed to be strict. The classes of the W-hierarchy are considered to be parameterized *intractability classes*.

For this hardness theory, the following notion of reductions is used. Let $L \subseteq \Sigma^* \times \mathbb{N}$ and $L' \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. An *fpt-reduction* (or *fixed-parameter tractable reduction*) from L to L' is a mapping $R : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ from instances of L to instances of L' such that there exist some computable function $g : \mathbb{N} \to \mathbb{N}$ such that for all $(x, k) \in \Sigma^* \times \mathbb{N}$: (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, i.e., in time $f(k)|x|^c$ for some computable function f and some constant c. We write $L \leq_{\mathrm{fpt}} L'$ if there is an fpt-reduction from L to L'. Similarly, we call reductions that satisfy properties (i) and (ii) but that are computable in time $O(|x|^{f(k)})$, for some fixed computable function f, *xp-reductions*.

The parameterized complexity classes $\mathrm{W}[t]$, $t \geq 1$, $\mathrm{W}[\mathrm{SAT}]$ and $\mathrm{W}[\mathrm{P}]$ are based on the satisfiability problems of Boolean circuits and formulas. We consider *Boolean circuits* with a single output gate. Boolean circuits are directed acyclic graphs, where each node with no ingoing edges is called an input node (or a *variable*), and where all other nodes are labelled with a Boolean operator (and are called *gates*). If there is an edge from a node r to a node r', we say that r is an *input* (or a *parent*) of r'. Gates that are labelled with a negation have exactly one input, and gates that are labelled with conjunction or negation can have more inputs. The number of inputs of a gate is called the *fan-in* of that gate. Similarly, the *fan-out* of a gate is the number of gates that have that gate as input. We distinguish between *small* gates, with fan-in at most 2, and *large gates*, with fan-in greater than 2. The *depth* of a circuit is the length of a longest path from any variable to the output gate. The *weft* of a circuit is the largest number of large gates on any path from

a variable to the output gate. We let Nodes(C) denote the set of all nodes of a circuit C. 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 at most 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 $\textsc{circ}_{t,u}$. 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.

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 parameterized problem Q, $[\ Q\ ]_{\text{fpt}}$ denotes the class of parameterized problems that are fpt-reducible to Q. The classes W[t] are defined by letting $\text{W}[t] = [\ \{\ \text{WSAT}[\textsc{circ}_{t,u}] : u \geq 1\ \}\ ]_{\text{fpt}}$, for each $t \geq 1$. The classes W[SAT] and W[P] are defined by letting $\text{W}[\text{SAT}] = [\ \text{WSAT}[\text{FORM}]\ ]_{\text{fpt}}$ and $\text{W}[\text{P}] = [\ \text{WSAT}[\textsc{circ}]\ ]_{\text{fpt}}$.

In addition, the completeness theory of parameterized complexity contains the A-hierarchy [19], containing the intractability classes A[t], for $t \geq 1$. (We give a formal definition of these parameterized complexity classes below, after defining the basic concepts of first-order logic.) The class A[1] coincides with W[1], and for each $t \geq 2$ it holds that $\text{W}[t] \subseteq \text{A}[t] \subseteq \text{A}[t+1] \subseteq \cdots \subseteq \text{XP}$. It is very unlikely that any parameterized problem that is hard for any of these parameterized intractability classes is fixed-parameter tractable, as this would violate commonly-believed assumptions in complexity theory, such as the Exponential Time Hypothesis [19, 37] (i.e., fixed-parameter tractability of any W[1]-hard problem would imply the existence of a $2^{o(n)}$ algorithm for n-variable 3SAT).

First-Order Logic Model Checking We define the basic concepts of first-order logic. A *(relational) vocabulary* τ is a finite set of relation symbols. Each relation symbol R has an *arity* $\text{arity}(R) \in \mathbb{N}$. A *structure* $\mathcal{A}$ of vocabulary τ, or *τ-structure* (or simply *structure*), consists of a set A called the *domain* (or *universe*) and an interpretation $R^{\mathcal{A}} \subseteq A^{\text{arity}(R)}$ for each relation symbol $R \in \tau$. In first-order logic, formulas are built from a countably infinite set $\{x_1, x_2, \ldots\}$ of variables, relation symbols, existential and universal quantification, and the Boolean operators $\neg, \wedge$, and $\vee$. That is, if $R \in \tau$ is a relation symbol of arity a, and $x_1, \ldots, x_a$ are variables, then $R(x_1, \ldots, x_a)$ is a formula. Moreover, if φ_1 and φ_2 are formulas and x is a variable, then $\exists x.\varphi_1$, $\forall x.\varphi_1$, $\neg\varphi_1$, $(\varphi_1 \wedge \varphi_2)$, and $(\varphi_1 \vee \varphi_2)$ are also formulas. We use $(\varphi_1 \rightarrow \varphi_2)$ as an abbreviation for $(\neg\varphi_1 \vee \varphi_2)$, and we use $(\varphi_1 \leftrightarrow \varphi_2)$ as an abbreviation for $((\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1))$. For a formula φ, we call the variables occurring in φ that are not bound by any quantifier the *free variables* of φ, and we write Free(φ) to denote the set of free variables in a formula φ. Formally, Free(φ) is defined inductively as follows:

$$\begin{aligned}
\text{Free}(R(x_1, \ldots, x_a)) &= \{x_1, \ldots, x_a\}, \\
\text{Free}(\neg\varphi) &= \text{Free}(\varphi), \\
\text{Free}(\varphi_1 \wedge \varphi_2) &= \text{Free}(\varphi_1) \cup \text{Free}(\varphi_2), \\
\text{Free}(\varphi_1 \vee \varphi_2) &= \text{Free}(\varphi_1) \cup \text{Free}(\varphi_2), \\
\text{Free}(\exists x.\varphi) &= \text{Free}(\varphi) \backslash \{x\}, \text{ and} \\
\text{Free}(\forall x.\varphi) &= \text{Free}(\varphi) \backslash \{x\}.
\end{aligned}$$

Truth of first-order formulas given a structure and an assignment to the free variables of the formula is defined in the usual way. Let $\mathcal{A}$ be a τ-structure with universe A, let φ be a first-order formula over the vocabulary τ, and let $\alpha : \text{Free}(\varphi) \rightarrow A$ be an assignment. We often consider the assignment α as a set of mappings, i.e., $\alpha = \{\ x \mapsto \alpha(x) : x \in \text{Free}(\varphi)\ \}$. Then the following conditions define when φ is true in $\mathcal{A}$

given α, written $\mathcal{A}, \alpha \models \varphi$.

$$\begin{aligned}
\mathcal{A}, \alpha \models R(x_1, \ldots, x_a) &\quad \text{if and only if} \quad (x_1, \ldots, x_a) \in R^{\mathcal{A}}, \\
\mathcal{A}, \alpha \models \neg\varphi &\quad \text{if and only if} \quad \mathcal{A}, \alpha \not\models \varphi, \\
\mathcal{A}, \alpha \models (\varphi_1 \wedge \varphi_2) &\quad \text{if and only if} \quad \mathcal{A}, \alpha \models \varphi_1 \text{ and } \mathcal{A}, \alpha \models \varphi_2, \\
\mathcal{A}, \alpha \models (\varphi_1 \vee \varphi_2) &\quad \text{if and only if} \quad \mathcal{A}, \alpha \models \varphi_1 \text{ or } \mathcal{A}, \alpha \models \varphi_2, \\
\mathcal{A}, \alpha \models \exists x.\varphi &\quad \text{if and only if} \quad \mathcal{A}, \alpha \cup \{x \mapsto a\} \models \varphi \text{ for some } a \in A, \text{ and} \\
\mathcal{A}, \alpha \models \forall x.\varphi &\quad \text{if and only if} \quad \mathcal{A}, \alpha \cup \{x \mapsto a\} \models \varphi \text{ for each } a \in A.
\end{aligned}$$

For more details, we refer to textbooks (see, e.g., [19, Section 4.2]). A *first-order logic sentence* is a first-order logic formula that contains no free variables, i.e., a formula φ such that $\text{Free}(\varphi) = \emptyset$. For sentences φ, we write $\mathcal{A} \models \varphi$ to denote $\mathcal{A}, \emptyset \models \varphi$.

The parameterized complexity class A[2] is defined on the basis of a first-order logic model checking problem [19]. In particular, we consider the problem A[2]-MC. Instances of this problem consist of a first-order structure $\mathcal{A}$ (over a signature τ), and a first-order logic sentence of the form $\varphi = \exists x_1, \ldots, x_{k_1} \forall y_1, \ldots, y_{k_2}.\psi$ (over the same signature τ), where ψ is quantifier-free. The parameter is $|\varphi|$, and the question is to decide whether $\mathcal{A} \models \varphi$. The parameterized complexity class A[2] consists of all parameterized problems that are fpt-reducible to A[2]-MC. Consequently, the problem A[2]-MC is A[2]-complete by definition. The parameterized complexity classes A[t], for $t \geq 3$, are defined in a similar way, using the variants A[t]-MC of A[2]-MC, where t quantifier alternations are allowed in the first-order logic sentence φ, rather than just 2.[2]

Parameterized Analogues of Classical Complexity Classes The following parameterized complexity classes are analogues of classical complexity classes. Let K be a classical complexity class, e.g., NP. The parameterized complexity class para-K is then defined as the class of all parameterized problems $L \subseteq \Sigma^* \times \mathbb{N}$ for which there exist a computable function $f : \mathbb{N} \to \Sigma^*$ and a problem $L' \in$ 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 $(x, f(k)) \in L'$—here we suppose that the pair $(x, f(k))$ is encoded as a string 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 characterized as the class of all parameterized problems that are solvable by a nondeterministic fpt-algorithm [18].

Besides the analogues para-NP and para-co-NP of the classical complexity classes NP and co-NP, we consider another parameterized analogue of these classes. Remember that XP is the class of parameterized problems P for which there exists a computable function f and an algorithm A that decides whether $(x, k) \in P$ in time $|x|^{f(k)}$. Similarly, we define XNP to be the class of parameterized problems that are decidable in nondeterministic time $|x|^{f(k)}$. Its co-class we denote by Xco-NP.[3]

Fpt-Reductions to SAT Another way to look at the parameterized complexity classes para-NP and para-co-NP is as the class of all parameterized problems that are fpt-reducible to SAT or UNSAT. Formally, we consider SAT as the decision problem $\{ \varphi : \varphi \text{ is a satisfiable propositional formula } \}$ and UNSAT as the decision problem $\{ \varphi : \varphi \text{ is an unsatisfiable propositional formula } \}$. By a slight abuse of notation, we will often also use SAT to refer to the (trivial) parameterized variant of the problem where the parameter value $k = 1$ is a fixed constant for all instances, i.e., to refer to the language $\{ (\varphi, 1) : \varphi \text{ is a satisfiable propositional formula } \}$. We use a similar convention for the problem UNSAT. In all cases, it is clear from the context whether the parameterized or the non-parameterized variant is meant.

The parameterized problem SAT is para-NP-complete, and the parameterized problem UNSAT is para-co-NP-complete [18, Proposition 14]. This means that para-NP consists of all parameterized problems that can be fpt-reduced to SAT, and that para-co-NP consists of all parameterized problems that can be fpt-reduced to UNSAT. Similarly, we can view XNP as the class of parameterized problems for which there

[2]The problems A[t]-MC are known in the literature under the name MC(Σ_t) (see, e.g., [19]). We use the name A[t]-MC in this paper to avoid confusion with the problem $\Sigma_2^{\text{p}}[k*]$-MC, that we will define in Section 4.1.

[3]Alternatively, one could denote this class by co-XNP.

exists an xp-reduction to SAT and Xco-NP as the class of parameterized problems for which there exists an xp-reduction to UNSAT—this can be proven with an analogous argument.

The parameterized complexity class para-NP can equivalently be characterized as the class of all parameterized problems that are fpt-reducible to any NP-complete problem.[4] The polynomial-time reductions from one NP-complete problem to another can be composed with fpt-reductions. Therefore, the existence of an fpt-reduction to any NP-complete problem implies the existence of an fpt-reduction to any other NP-complete problem. Consequently, it makes no difference whether we consider fpt-reductions to the satisfiability problem for propositional formulas, to the satisfiability problem for Boolean circuits, or to the satisfiability problem of propositional formulas in conjunctive normal form (CNF), as these problems are all NP-complete.

3 The Hierarchies and Basic Results

As we have seen in the previous section, a parameterized problem is fpt-reducible to SAT if and only if it is contained in the class para-NP (and similarly, it is fpt-reducible to UNSAT if and only if it is in para-co-NP). On the other hand, one can use hardness for the classes para-Σ_2^{P} or para-Π_2^{P} to give evidence that a parameterized problem is not fpt-reducible to SAT—for instance, a para-Σ_2^{P}-hard problem is not in para-NP, unless the PH collapses. However, it turns out that there are many natural parameterized problems that seem to be neither in para-NP or para-co-NP, nor hard for para-Σ_2^{P} or para-Π_2^{P}. That is, the computational complexity of these problems lies at an intermediate level. In order to adequately characterize the parameterized complexity of these problems, we need new parameterized complexity classes beyond para-NP and para-co-NP, on the one hand, and below para-Σ_2^{P} and para-Π_2^{P}, on the other hand.

In this section, we define two hierarchies of parameterized complexity classes: the k-$*$ and the $*$-k hierarchies, consisting of the parameterized complexity classes $\Sigma_2^{\mathrm{p}}[k*, t]$ and $\Sigma_2^{\mathrm{p}}[*k, t]$, respectively. Moreover, we provide basic structural results for these parameterized complexity classes.

The classes $\Sigma_2^{\mathrm{p}}[k*, t]$ and $\Sigma_2^{\mathrm{p}}[*k, t]$ are based on weighted variants of QSAT_2, the satisfiability problem for quantified Boolean formulas with an $\exists\forall$ quantifier prefix. That is, an instance of the problem QSAT_2 has both an existential quantifier and a universal quantifier block. Therefore, there are several ways of restricting the weight of assignments. Restricting the weight of assignments to the existential quantifier block results in the k-$*$ hierarchy, and restricting the weight of assignments to the universal quantifier block results in the $*$-k hierarchy. Incidentally, restricting the weight of assignments to both quantifier blocks simultaneously results in a hierarchy of classes (dubbed "k-k") that are closely related to the classes of the A-hierarchy.

After defining the classes $\Sigma_2^{\mathrm{p}}[k*, t]$ and $\Sigma_2^{\mathrm{p}}[*k, t]$, we show that the k-$*$ hierarchy in fact collapses to a single class $\Sigma_2^{\mathrm{p}}[k*]$—that is, $\Sigma_2^{\mathrm{p}}[k*, 1] = \Sigma_2^{\mathrm{p}}[k*, 2] = \cdots = \Sigma_2^{\mathrm{p}}[k*, \mathrm{P}] = \Sigma_2^{\mathrm{p}}[k*]$. Moreover, we give normalization results for the classes $\Sigma_2^{\mathrm{p}}[*k, 1]$ and $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$. Concretely, we show that hardness of the canonical problem for $\Sigma_2^{\mathrm{p}}[*k, 1]$ already holds when the problem is restricted to quantified Boolean formulas whose matrix is in 2DNF, and we show that hardness of the canonical problem for $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$ already holds when the problem is restricted to quantified Boolean circuits that are monotone in the universally quantified variables.

A graphical overview of the relation of the parameterized complexity classes $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k, t]$ (and their co-classes $\Pi_2^{\mathrm{p}}[k*]$ and $\Pi_2^{\mathrm{p}}[*k, t]$) is provided in Figure 1.

Example: a natural $\Sigma_2^{\mathrm{p}}[k*]$-complete problem Before we dive into the technical details of defining and developing the parameterized complexity classes $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k, t]$, we briefly discuss an example of a natural parameterized problem whose complexity lies beyond para-NP and para-co-NP, on the one hand, and below para-Σ_2^{P} and para-Π_2^{P}, on the other hand. One aim of considering this example is to give the reader some intuition about the kind of parameterized problems that motivate the development of the classes $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k, t]$. Another aim of this example is to help convey the message that the parameterized complexity classes that we develop are not just the result of a theoretical exercise, but that they capture the level of computational complexity of natural problems that occur in various settings. To further pursue this latter

[4] To put it more precisely, para-NP can be characterized as the class of all parameterized problems that are fpt-reducible to any problem Q_c, where Q is an NP-complete problem, $c \in \mathbb{N}$ is an arbitrary constant, and $Q_c = \{ (x, c) : x \in Q \}$ is the parameterized variant of Q where the parameter value is the constant c.

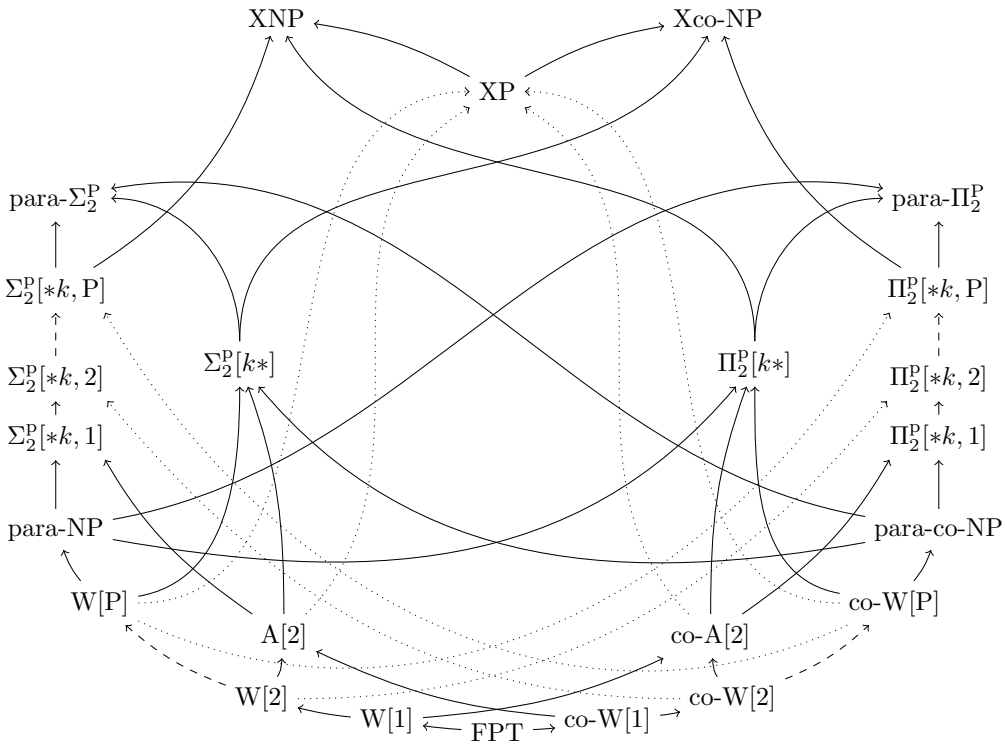


Figure 1: The parameterized complexity classes $\Sigma_2^{\mathrm{p}}[k*]$, $\Pi_2^{\mathrm{p}}[k*]$, $\Sigma_2^{\mathrm{p}}[*k, t]$, and $\Pi_2^{\mathrm{p}}[*k, t]$, and their relation to parameterized complexity classes known from the literature.

goal, in Section 6 we provide a list of natural parameterized problems from many domains that are complete for the classes that we develop.

The example problem that we consider here is related to the task of minimizing DNF formulas. In this problem, one is given a DNF formula φ and a positive integer k. The problem is to decide if there is a DNF formula φ' that is equivalent to φ and that can be obtained from φ by deleting k literals. This problem is Σ_2^{P}-complete in general [54]. We consider the parameterized variant of this problem where the parameter is the number k of literals that is to be deleted. This parameterized problem is $\Sigma_2^{\mathrm{p}}[k*]$-complete [31, Proposition 3]. The structure of this problem nicely corresponds to the intuition behind the class $\Sigma_2^{\mathrm{p}}[k*]$: the question is whether there is a set of k literals (among the n literals occuring in the formula) that can be deleted—a choice between $\binom{n}{k} = O(n^k)$ possible sets—so that all $2^{O(n)}$ truth assignments yield the same value. Problems complete for $\Sigma_2^{\mathrm{p}}[k*]$ are generally of this shape. Similarly, problems complete for the classes $\Sigma_2^{\mathrm{p}}[*k, t]$ typically involve finding a solution among one of 2^n candidate solutions, and for each such candidate, checking whether it is indeed a solution corresponds to verifying a property for all sets of k objects.

3.1 The k-$*$ and $*$-k Hierarchies

We now turn to formally defining the k-$*$ and $*$-k hierarchies, consisting of parameterized complexity classes $\Sigma_2^{\mathrm{p}}[k*, t]$ and $\Sigma_2^{\mathrm{p}}[*k, t]$, respectively. These classes are based on the following two parameterized decision problems. Let $\mathcal{C}$ be a class of Boolean circuits. The problem $\Sigma_2^{\mathrm{p}}[k*]$-WSAT($\mathcal{C}$) provides the foundation for the k-$*$ hierarchy.

$\Sigma_2^{\mathrm{p}}[k*]$-WSAT($\mathcal{C}$)
Instance: A Boolean circuit $C \in \mathcal{C}$ over two disjoint sets X and Y of variables, and a positive 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?

Similarly, the problem $\Sigma_2^{\mathrm{p}}[*k]$-WSAT($\mathcal{C}$) provides the foundation for the $*$-k hierarchy.

$\Sigma_2^{\mathrm{p}}[*k]$-WSAT($\mathcal{C}$)
Instance: A Boolean circuit $C \in \mathcal{C}$ over two disjoint sets X and Y of variables, and a positive 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?

For the sake of convenience, instances to these two problems consisting of a circuit C over sets X and Y of variables and a positive integer k, we will denote by $(\exists X.\forall Y.C, k)$.

We will now define the following parameterized complexity classes, that together form the k-$*$ hierarchy. Remember that the notation $[\ \cdot\ ]_{\mathrm{fpt}}$ denotes the class of all parameterized problems that are fpt-reducible to the referenced (set of) problem(s). Remember also that $\textsc{circ}_{t,u}$ denotes the class of all Boolean circuits of weft t and depth u, that FORM denotes the class of all Booelan circuits that represent a propositional formula, and that CIRC denotes the class of all Boolean circuits. The classes of the k-$*$ hierarchy are defined as follows:

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

Similarly, we define the classes of the $*$-k hierarchy as follows:

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

$$\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}] = [\ \Sigma_2^{\mathrm{p}}[*k]\text{-WSAT}(\text{CIRC})\ ]_{\mathrm{fpt}}.$$

These definitions are entirely analogous to those of the parameterized complexity classes W[t] of the W-hierarchy [10, 11].

By definition of the classes $\Sigma_2^{\mathrm{p}}[k*, t]$ and $\Sigma_2^{\mathrm{p}}[k*, t]$, we directly get the following inclusions:

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

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

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 each of the parameterized complexity classes in the k-$*$ and $*$-k hierarchies. 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. These complementary problems are denoted by $\Pi_2^{\mathrm{p}}[k*]$-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 $\Pi_2^{\mathrm{p}}[*k, t]$.

3.2 The k-k Hierarchy

Before we continue with developing structural results for the k-$*$ and $*$-k hierarchies, we briefly digress and consider another similar hierarchy of complexity classes (that we call the k-k hierarchy). We will use one of the parameterized complexity classes $\Sigma_2^{\mathrm{p}}[kk, t]$ in this additional hierarchy to establish some results in Section 5.3.

Similarly to the definition of the complexity classes of the k-$*$ and $*$-k hierarchies, one can define weighted variants of the problem QSAT$_2$ with weight restrictions on both quantifier blocks. This results in the parameterized complexity classes $\Sigma_2^{\mathrm{p}}[kk, t]$, whose definition is based on the following parameterized problem. Let $\mathcal{C}$ be a class of Boolean circuits. The problem $\Sigma_2^{\mathrm{p}}[kk]$-WSAT($\mathcal{C}$) provides the foundation for the k-k hierarchy.

$\Sigma_2^{\mathrm{p}}[kk]$-WSAT($\mathcal{C}$)
Instance: A Boolean circuit $C \in \mathcal{C}$ over two disjoint sets X and Y of variables, and a positive 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 of weight k the assignment $\alpha \cup \beta$ satisfies C?

The classes $\Sigma_2^{\mathrm{p}}[kk, t]$, for $t \in \mathbb{N} \cup \{\mathrm{SAT}, \mathrm{P}\}$ are then defined as follows:

$$\Sigma_2^{\mathrm{p}}[kk, t] = [\ \{\ \Sigma_2^{\mathrm{p}}[kk]\text{-WSAT}(\text{CIRC}_{t,u}) : u \geq 1\ \}\ ]_{\mathrm{fpt}},$$
$$\Sigma_2^{\mathrm{p}}[kk, \mathrm{SAT}] = [\ \Sigma_2^{\mathrm{p}}[kk]\text{-WSAT}(\text{FORM})\ ]_{\mathrm{fpt}}, \text{ and}$$
$$\Sigma_2^{\mathrm{p}}[kk, \mathrm{P}] = [\ \Sigma_2^{\mathrm{p}}[kk]\text{-WSAT}(\text{CIRC})\ ]_{\mathrm{fpt}}.$$

The complexity class $\Sigma_2^{\mathrm{p}}[kk, \mathrm{SAT}]$ has been defined and considered by Gottlob, Scarcello and Sideri [26] under the name Σ_2W[SAT]. Also, for each $t \in \mathbb{N}$, variants of the problems $\Sigma_2^{\mathrm{p}}[kk, t]$ have been studied in the literature (see, e.g., [19, Chapter 8]). Based on these problems, the parameterized complexity classes A[2,t] (for $t \geq 1$) have been defined. These classes generalize A[2], because A[2] = A[2,1]. Due to fact that the classes A[2,t] and the classes $\Sigma_2^{\mathrm{p}}[kk, t]$ are defined in a very similar way—in fact, the canonical problems for the classes A[2,t] are a special case of the problems $\Sigma_2^{\mathrm{p}}[kk]$-WSAT($\text{CIRC}_{t,u}$)—it is straightforward to verify that for all $t \geq 1$ it holds that A[2,t] $\subseteq \Sigma_2^{\mathrm{p}}[kk, t]$.

Moreover, it can also routinely be proved that for each $t \in \mathbb{N} \cup \{\mathrm{SAT}, \mathrm{P}\}$ it holds that $\Sigma_2^{\mathrm{p}}[kk, t] \subseteq \Sigma_2^{\mathrm{p}}[k*, t]$ and that $\Sigma_2^{\mathrm{p}}[kk, t] \subseteq \Sigma_2^{\mathrm{p}}[*k, t]$. Therefore, we directly get the following result (that we state without proof), that relates A[2] and the classes of the k-$*$ and $*$-k hierarchies.

Proposition 1. *Let* $t \in \mathbb{N} \cup \{\mathrm{SAT}, \mathrm{P}\}$. *Then* $\mathrm{A}[2] \subseteq \Sigma_2^{\mathrm{p}}[kk, t] \subseteq \Sigma_2^{\mathrm{p}}[k*, t] \cap \Sigma_2^{\mathrm{p}}[*k, t]$.

3.3 The Parameterized Complexity Class $\Sigma_2^{\mathrm{p}}[k*]$

We consider the classes $\Sigma_2^{\mathrm{p}}[k*, t]$ of the k-$*$ hierarchy in more detail. It turns out that this hierarchy collapses entirely into a single parameterized complexity class, that we denote by $\Sigma_2^{\mathrm{p}}[k*]$. We show that the classes of the k-$*$ hierarchy all coincide. We do so by showing that $\Sigma_2^{\mathrm{p}}[k*, 1] = \Sigma_2^{\mathrm{p}}[k*, \mathrm{P}]$.

Theorem 2 (Collapse of the k-$*$ hierarchy)**.** $\Sigma_2^{\mathrm{p}}[k*, 1] = \Sigma_2^{\mathrm{p}}[k*, \mathrm{P}]$. *Moreover,* $\Sigma_2^{\mathrm{p}}[k*]$-WSat(3DNF) *is complete for this class.*

Proof. Since by definition $\Sigma_2^{\mathrm{p}}[k*, 1] \subseteq \Sigma_2^{\mathrm{p}}[k*, 2] \subseteq \ldots \subseteq \Sigma_2^{\mathrm{p}}[k*, \mathrm{P}]$, it suffices to show that $\Sigma_2^{\mathrm{p}}[k*, \mathrm{P}] \subseteq \Sigma_2^{\mathrm{p}}[k*, 1]$. We show this by giving an fpt-reduction from $\Sigma_2^{\mathrm{p}}[k*]$-WSat(circ) to $\Sigma_2^{\mathrm{p}}[k*]$-WSat(3DNF). Since 3DNF $\subseteq$ $\mathrm{circ}_{1,3}$, this suffices. We remark that this reduction is based on the standard Tseitin transformation that transforms arbitrary Boolean circuits into 3CNF by means of additional variables [53].

Let (φ, k) be an instance of $\Sigma_2^{\mathrm{p}}[k*]$-WSat(circ) with $\varphi = \exists X.\forall Y.C$. Assume without loss of generality that C contains only binary conjunctions and negations. Let o denote the output gate of C. We construct an instance (φ', k) of $\Sigma_2^{\mathrm{p}}[k*]$-WSat(3DNF) as follows. The formula φ' will be over the set of variables $X \cup Y \cup Z$, where $Z = \{\, z_r : r \in \mathrm{Nodes}(C) \,\}$. For each $r \in \mathrm{Nodes}(C)$, we define a subformula χ_r. We distinguish three cases. If $r = r_1 \wedge r_2$, then we let $\chi_r = (z_r \wedge \neg z_{r_1}) \vee (z_r \wedge \neg z_{r_2}) \vee (z_{r_1} \wedge z_{r_2} \wedge \neg z_r)$. If $r = \neg r_1$, then we let $\chi_r = (z_r \wedge z_{r_1}) \vee (\neg z_r \wedge \neg z_{r_1})$. If $r = w$, for some $w \in X \cup Y$, then we let $\chi_r = (z_r \wedge \neg w) \vee (\neg z_r \wedge w)$. Now we define $\varphi' = \exists X.\forall Y \cup Z.\psi$, where $\psi = \bigvee_{r \in \mathrm{Nodes}(C)} \chi_r \vee z_o$. We prove the correctness of this reduction.

($\Rightarrow$) Assume that $(\varphi, k) \in \Sigma_2^{\mathrm{p}}[k*]$-WSat(circ). This means that there exists an assignment $\alpha : X \to \{0, 1\}$ of weight k such that $\forall Y.C[\alpha]$ evaluates to true. We show that $(\varphi', k) \in \Sigma_2^{\mathrm{p}}[k*]$-WSat(3DNF), by showing that $\forall Y \cup Z.\psi[\alpha]$ evaluates to true. Let $\beta : Y \cup Z \to \{0, 1\}$ be an arbitrary assignment to the variables $Y \cup Z$, and let β' be the restriction of β to the variables Y. We distinguish two cases: either (i) for each $r \in \mathrm{Nodes}(C)$ it holds that $\beta(z_r)$ coincides with the value that gate r gets in the circuit C given assignment $\alpha \cup \beta'$, or (ii) this is not the case. In case (i), by the fact that $\alpha \cup \beta'$ satisfies C, we know that $\beta(z_o) = 1$, and therefore $\alpha \cup \beta$ satisfies ψ. In case (ii), we know that for some gate $r \in \mathrm{Nodes}(C)$, the value of $\beta(z_r)$ does not coincide with the value assigned to r in C given the assignment $\alpha \cup \beta'$. We may assume without loss of generality that for all parent nodes r' of r it holds that $\beta(z_{r'})$ coincides with the value assigned to r' by $\alpha \cup \beta'$. In this case, there is some term of χ_r that is satisfied by $\alpha \cup \beta$. From this we can conclude that $(\varphi', k) \in \Sigma_2^{\mathrm{p}}[k*]$-WSat(3DNF).

($\Leftarrow$) Assume that $(\varphi', k) \in \Sigma_2^{\mathrm{p}}[k*]$-WSat(3DNF). This means that there exists some assignment $\alpha : X \to \{0, 1\}$ of weight k such that $\forall Y \cup Z.\psi[\alpha]$ evaluates to true. We now show that $\forall Y.C[\alpha]$ evaluates to true as well. Let $\beta' : Y \to \{0, 1\}$ be an arbitrary assignment to the variables Y. Define $\beta'' : Z \to \{0, 1\}$ as follows. For any $r \in \mathrm{Nodes}(C)$, we let $\beta''(z_r)$ be the value assigned to the node r in the circuit C by the assignment $\alpha \cup \beta'$. We then let $\beta = \beta' \cup \beta''$. Now, since $\forall Y \cup Z.\psi[\alpha]$ evaluates to true, we know that $\alpha \cup \beta$ satisfies ψ. By construction of β, we know that $\alpha \cup \beta$ does not satisfy the term χ_r for any $r \in \mathrm{Nodes}(C)$. Therefore, we know that $\beta(z_o) = 1$. By construction of β, this implies that $\alpha \cup \beta'$ satisfies C. Since β' was arbitrary, we can conclude that $\forall Y.C[\alpha]$ evaluates to true, and therefore that $(\varphi, k) \in \Sigma_2^{\mathrm{p}}[k*]$-WSat(circ). □

As mentioned above, in order to simplify notation, we will use $\Sigma_2^{\mathrm{p}}[k*]$ to denote the class $\Sigma_2^{\mathrm{p}}[k*, 1] = \ldots = \Sigma_2^{\mathrm{p}}[k*, \mathrm{P}]$. Also, for the sake of convenience, by a slight abuse of notation, we will often denote the problems $\Sigma_2^{\mathrm{p}}[k*]$-WSat(circ) and $\Sigma_2^{\mathrm{p}}[k*]$-WSat(form) by $\Sigma_2^{\mathrm{p}}[k*]$-WSat.

The result of Theorem 2 is useful for showing $\Sigma_2^{\mathrm{p}}[k*]$-hardness because it allows us to restrict our attention to instances where the matrix is in 3DNF when conceiving a reduction from $\Sigma_2^{\mathrm{p}}[k*]$-WSat. For example, the reductions used to show that the problem of reducing the size of a DNF formula by deleting k literals while preserving logical equivalence (that we discussed as an example in the beginning of Section 3) are based on this normalization result.

3.4 Normalization Results for $\Sigma_2^{\mathrm{p}}[*k, 1]$ and $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$

We now turn our attention to the classes $\Sigma_2^{\mathrm{p}}[*k, t]$. The proof technique that we used to show Theorem 2 cannot be used to show a collapse of the $*$-k hierarchy. Intuitively, the reason for this is that when dealing with the various problems $\Sigma_2^{\mathrm{p}}[*k]$-WSat($\mathcal{C}$), we cannot freely add auxiliary variables that can get any assignment

to the second quantifier block because the weight of assignments to the variables in the second quantifier block is restricted. It is possible that the classes of the $*$-k hierarchy are distinct.

In this section, we show normalization results for the classes $\Sigma_2^{\mathrm{p}}[*k, 1]$ and $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$. In particular, we show that the problem $\Sigma_2^{\mathrm{p}}[*k]$-WSat(2DNF) is already $\Sigma_2^{\mathrm{p}}[*k, 1]$-hard, and that the problem $\Sigma_2^{\mathrm{p}}[*k]$-WSat(circ) is already $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$-hard when restricted to circuits where the universally quantified variables occur only positively in the circuit. These normalization results are useful for showing hardness for the classes $\Sigma_2^{\mathrm{p}}[*k, 1]$ and $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$. In fact, all the $\Sigma_2^{\mathrm{p}}[*k, 1]$-completeness results and the $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$-completeness results that we mention in Section 6 are based on the normalization results in this section.

3.4.1 A Normalization Result for $\Sigma_2^{\mathrm{p}}[*k, 1]$

We begin with showing a normalization result for (the canonical problems of) the parameterized complexity class $\Sigma_2^{\mathrm{p}}[*k, 1]$. In particular, we show that the problem $\Sigma_2^{\mathrm{p}}[*k]$-WSat is already $\Sigma_2^{\mathrm{p}}[*k, 1]$-hard when the input circuits are restricted to formulas in r-DNF, for any constant $r \geq 2$. For the sake of convenience, we switch our perspective to the co-problem $\Pi_2^{\mathrm{p}}[*k]$-WSat when stating and proving the following results. Because we can make heavy use of the original normalization proof for the class W[1] by Downey and Fellows [9, 10, 11] to prove this normalization result, we provide only proof sketches. We begin with the following lemma, that shows that we can restrict our attention to CNF formulas (with higher bounds on the size of clauses).

Lemma 3. *For any $u \geq 1$, $\Pi_2^{\mathrm{p}}[*k]$-WSat($\mathrm{circ}_{1,u}$) $\leq_{\mathrm{fpt}}$ $\Pi_2^{\mathrm{p}}[*k]$-WSat(s-CNF), where $s = 2^u + 1$.*

Proof (sketch). The reduction is completely analogous to the reduction used in the proof of Downey and Fellows [9, Lemma 2.1], where the presence of universally quantified variables is handled in four steps. In Steps 1 and 2, in which only the form of the circuit is modified, no changes are needed. In Step 3, universally quantified variables can be handled exactly as existentially quantified variables. Step 4 can be performed with only a slight modification, the only difference being that universally quantified variables appearing in the input circuit will also appear in the resulting clauses that verify whether a given product-of-sums or sum-of-products is satisfied. It is straightforward to verify that this reduction with the mentioned modifications works for our purposes. □

We are now ready to prove our normalization result.

Theorem 4. $\Pi_2^{\mathrm{p}}[*k]$-WSat(2CNF) *is* $\Pi_2^{\mathrm{p}}[*k, 1]$*-complete.*

Proof (sketch). Clearly $\Pi_2^{\mathrm{p}}[*k]$-WSat(2CNF) is in $\Pi_2^{\mathrm{p}}[*k, 1]$, since a 2CNF formula can be considered as a constant-depth circuit of weft 1. To show that $\Pi_2^{\mathrm{p}}[*k]$-WSat(2CNF) is $\Pi_2^{\mathrm{p}}[*k, 1]$-hard, we give an fpt-reduction from $\Pi_2^{\mathrm{p}}[*k]$-WSat($\mathrm{circ}_{1,u}$) to $\Pi_2^{\mathrm{p}}[*k]$-WSat(2CNF), for arbitrary $u \geq 1$. By Lemma 3, we know that we can reduce $\Pi_2^{\mathrm{p}}[*k]$-WSat($\mathrm{circ}_{1,u}$) to $\Pi_2^{\mathrm{p}}[*k]$-WSat(s-CNF), for $s = 2^u + 1$. We continue the reduction in multiple steps. In each step, we let C denote the circuit resulting from the previous step, and we let Y denote the universally quantified and X the existentially quantified variables of C, and we let k denote the parameter value. We describe the last two steps only briefly, since these are completely analogous to constructions in the work of Downey and Fellows [10].

Step 1: contracting the universally quantified variables. This step transforms C into a CNF formula C' such that each clause contains at most one variable in Y and such that (C, k) is a yes-instance if and only if (C', k) is a yes-instance. We introduce new universally quantified variables Y' containing a variable y'_A for each set A of literals over Y of size at least 1 and at most s. Now, it is straightforward to construct a set D of polynomially many clauses of size 3 over Y and Y' such that the following property holds. An assignment α to $Y \cup Y'$ satisfies D if and only if for each subset $A = \{l_1, \ldots, l_b\}$ of literals over Y (of size at least 1 and at most s) it holds that $\alpha(y'_A) = 1$ if and only if $\alpha(l_j) = 1$ for some $j \in \{1, \ldots, b\}$. Note that we do not directly add the set D of clauses to the formula C'.

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

We introduce $|D|$ existentially quantified variables, including a variable x'_d for each clause $d \in D$. Let X' denote the new existentially quantified variables that we introduced, i.e., $X' = \{x^\star_1, \ldots, x^\star_{k-1}\} \cup \{\, x'_d : d \in D \,\}$. Then, for each $d \in D$, we add the following clauses to C'. Let $d = (l_1, l_2, l_3)$, where each l_i is a literal over $Y \cup Y'$. We add the clauses $(\neg x'_d \vee \neg l_1)$, $(\neg x'_d \vee \neg l_2)$ and $(\neg x'_d \vee \neg l_3)$, enforcing that the clause d cannot be satisfied if x'_d is set to true.

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

The idea of this reduction is the following. If $x^\star_1$ is set to true, then exactly one of the variables x'_d must be set to true—this is because the variables $x^\star_1, \ldots, x^\star_{k-1}$ must be set to true, all variables $x \in X$ must be set to false, and exactly k variables must be set to true. This can only result in an satisfying assignment if the clause $d \in D$ is not satisfied. Therefore, if an assignment α to the variables $Y \cup Y'$ does not satisfy D, there is a satisfying assignment of weight k that sets both $x^\star_1$ and x'_d to true, for some $d \in D$ that is not satisfied by α. Otherwise, we know that the value that α assigns to variables y'_A corresponds to the value that α assigns to $\bigvee_{a \in A} a$, for each set A of at most s literals over Y. Then any satisfying assignment of weight k for C is also a satisfying assignment of weight k for C'.

We formally prove that (C, k) is a yes-instance if and only if (C', k) is a yes-instance.

($\Rightarrow$) Suppose that for each $\alpha : Y \to \{0,1\}$ there is a truth assignment $\beta : X \to \{0,1\}$ of weight k such that $C[\alpha \cup \beta]$ is true. We show that (C', k) is a yes-instance. Take an arbitrary truth assignment $\alpha' : Y \cup Y' \to \{0,1\}$. We distinguish two cases: either (i) α' does not satisfy D or (ii) α' satisfies D. In case (i), we know that there is some clause $d \in D$ that is not satisfied by α'. Consider the truth assignment $\beta'_d : X \to \{0,1\}$ that sets the variables $x^\star_1, \ldots, x^\star_{k-1}$ and x'_d to 1, and all remaining variables to 0. It is straightforward to verify that β'_d has weight k and that $C[\alpha' \cup \beta'_d]$ is true. For case (ii), we consider the restriction α of α' to the variables Y. We know that there is some $\beta : X \to \{0,1\}$ of weight k such that $C[\alpha \cup \beta]$ is true. We construct the truth assignment $\beta' : X \cup X' \to \{0,1\}$ by letting $\beta(x) = \beta'(x)$ for all $x \in X$, and letting $\beta(x') = 0$ for all $x' \in X'$. Clearly, β has weight k. Moreover, by construction of C', since $C[\alpha \cup \beta]$ is true and because α satisfies D, we know that $C'[\alpha' \cup \beta']$ is true.

($\Leftarrow$) Conversely, suppose that for each $\alpha' : Y \cup Y' \to \{0,1\}$ there is a truth assignment $\beta' : X \cup X' \to \{0,1\}$ of weight k such that $C'[\alpha' \cup \beta']$ is true. We show that (C, k) is a yes-instance. Take an arbitrary truth assignment $\alpha : Y \to \{0,1\}$. Consider a truth assignment $\alpha' : Y \cup Y' \to \{0,1\}$ that extends α and that satisfies D. We know that there exists some truth assignment $\beta' : X \cup X' \to \{0,1\}$ of weight k such that $C'[\alpha' \cup \beta']$ is true. Moreover, by construction of C', since α' satisfies D, we know that $\beta'(x'_d) = 0$ for all $d \in D$. Then, we also know that $\beta'(x^\star_1) = 0$. If this were not the case, β' would be of weight $k-1$, which contradicts our assumption that β' has weight k. We then consider the restriction β of β' to the variables in X. We know that β has weight k. Moreover, by construction of C', since $C'[\alpha' \cup \beta']$ is true, and because α' satisfies D, we know that $C[\alpha \cup \beta]$ is true.

Step 2: making C antimonotone in X. This step transforms C into a circuit C' that has only negative occurrences of existentially quantified variables, and transforms k into k' depending only on k, such that (C, k) is a yes-instance if and only if (C', k') is a yes-instance. The reduction is completely analogous to the reduction in the proof of Downey and Fellows [10, Theorem 10.6].

Step 3: contracting the existentially quantified variables. This step transforms C into a circuit C' in CNF that contains only clauses with two variables in X and no variables in Y and clauses with one variable in X and one variable in Y, and transforms k into k' depending only on k, such that (C, k) is a yes-instance if and only if (C', k') is a yes-instance. The reduction is completely analogous to the reduction in the proof of Downey and Fellows [10, Theorem 10.7]. □

Corollary 5. *For any fixed integer $r \geq 2$, the problem $\Sigma^{\mathrm{p}}_2[*k]$-WSat($r$-DNF) is $\Sigma^{\mathrm{p}}_2[*k, 1]$-complete.*

3.4.2 A Normalization Result for $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$

Next, we provide a normalization result for $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$. In order to do so, we will need some definitions. Let C be a quantified Boolean circuit over two disjoint sets X and Y of variables that is in negation normal form. We say that C is *monotone in the variables in* Y if the only negation nodes that occur in the circuit C have variables in X as inputs, i.e., the variables in Y can appear only positively in the circuit. Then, the following restriction of $\Sigma_2^{\mathrm{p}}[*k]$-WSAT is already $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$-hard.

Theorem 6. *The problem $\Sigma_2^{\mathrm{p}}[*k]$-WSAT is $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$-hard, even when restricted to quantified circuits that are in negation normal form and that are monotone in the universal variables.*

Proof. We give an fpt-reduction from the problem $\Sigma_2^{\mathrm{p}}[*k]$-WSAT to the problem $\Sigma_2^{\mathrm{p}}[*k]$-WSAT restricted to circuits that are monotone in the universal variables. Let (C, k) be an instance of $\Sigma_2^{\mathrm{p}}[*k]$-WSAT, where C is a quantified Boolean circuit over the set X of existential variables and the set Y of universal variables, where $X = \{x_1, \ldots, x_n\}$ and where $Y = \{y_1, \ldots, y_m\}$. We construct an equivalent instance (C', k) of $\Sigma_2^{\mathrm{p}}[*k]$-WSAT where C' is a quantified Boolean circuit over the set X of existential variables and the set Y' of universal variables, and where the circuit C' is monotone in Y'. We may assume without loss of generality that C is in negation normal form. If this is not the case, we can simply transform C into an equivalent circuit that has this property using the De Morgan rule. The form of the circuit C is depicted in Figure 2.

This construction bears some resemblance to the construction used in a proof by Flum and Grohe [19, Theorem 3.14]. The plan is to replace the variables in Y by k copies, grouped in sets $Y^1, \ldots, Y^k$ of new variables. Each assignment of weight k to the new variables that sets a copy of a different variable to true in each set Y^i corresponds exactly to an assignment of weight k to the original variables in Y. Moreover, we will ensure that each assignment of weight k to the new variables that does not set a copy of a different variable to true in each set Y^i satisfies the newly constructed circuit. Using these new variables we can then construct internal nodes y_j and y'_j that, for each assignment to the new input nodes Y', evaluate to the truth value assigned to y_j and $\neg y_j$, respectively, by the corresponding truth assignment to the original input nodes Y.

We will describe this construction in more detail. The construction is also depicted in Figure 3. We let $Y' = \{\, y_j^i : 1 \leq i \leq k, 1 \leq j \leq m \,\}$. We introduce a number of new internal nodes. For each $1 \leq j \leq m$, we introduce an internal node y_j, that is the disjunction of the input nodes y_j^i, for $1 \leq i \leq k$. That is, the internal node y_j is true if and only if y_j^i is true for some $1 \leq i \leq k$. Intuitively, this node y_j corresponds to the input node y_j in the original circuit C. Moreover, we introduce an internal node $y'_{j,i}$ for each $1 \leq j \leq m$ and each $1 \leq i \leq k$, that is the disjunction of $y_{j'}^i$, for each $1 \leq j' \leq m$ such that $j \neq j'$. That is, the node $y'_{j,i}$ is true if and only if $y_{j'}^i$ is true for some j' that is different from j. Then, we introduce the node y'_j, for each $1 \leq j \leq m$, that is the conjunction of the nodes $y'_{j,i}$ for $1 \leq i \leq k$. That is, the node y'_j is true if and only if for each $1 \leq i \leq k$ there is some $j' \neq j$ for which the input node $y_{j'}^i$ is true. Intuitively, this node y'_j corresponds to the negated input node $\neg y_j$ in the original circuit C. Also, for each $1 \leq i \leq k$ and each $1 \leq j < j' \leq m$, we add an internal node $z_i^{j,j'}$ that is the conjunction of the input nodes y_j^i and $y_{j'}^i$. Then, for each $1 \leq i \leq k$ we add the internal node z_i that is the conjunction of all nodes $z_i^{j,j'}$, for $1 \leq j < j' \leq m$. Intuitively, z_i is true if and only if at least two input nodes in the set Y_i are set to true. In addition, we add a subcircuit B that acts on the nodes $y'_1, \ldots, y'_m$, and that is satisfied if and only if at least $m - k + 1$ of the nodes y'_j are set to true. It is straightforward to construct such a circuit B in polynomial time (see, e.g., [50, Figure 1]). Then, we add the subcircuit C with input nodes $x_1, \ldots, x_n$, negated input nodes $\neg x_1, \ldots, \neg x_n$, where the input nodes $y_1, \ldots, y_m$ are identified with the internal nodes $y_1, \ldots, y_m$ in the newly constructed circuit C', and where the negated input nodes $\neg y_1, \ldots, \neg y_m$ are identified with the internal nodes $y'_1, \ldots, y'_m$ in the newly constructed circuit C'. Finally, we let the output node be the disjunction of the nodes $z_1, \ldots, z_k$ and the output nodes of the subcircuits C and B. Since C is a circuit in negation normal form, the circuit C' is monotone in Y'. We claim that for each assignment $\alpha : X \to \{0, 1\}$ it holds that the circuit $C[\alpha]$ is satisfied by all assignments of weight k if and only if $C'[\alpha]$ is satisfied by all assignments of weight k.

($\Rightarrow$) Let $\alpha : X \to \{0, 1\}$ be an arbitrary truth assignment. Assume that $C[\alpha]$ is satisfied by all truth assignments $\beta : Y \to \{0, 1\}$ of weight k. We show that $C'[\alpha]$ is satisfied by all truth assignments $\beta' : Y' \to \{0, 1\}$ of weight k. Let $\beta' : Y' \to \{0, 1\}$ be an arbitrary truth assignment of weight k. We distinguish several

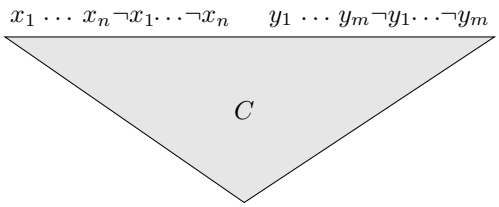


Figure 2: The original quantified Boolean circuit C in the proof of Theorem 6.

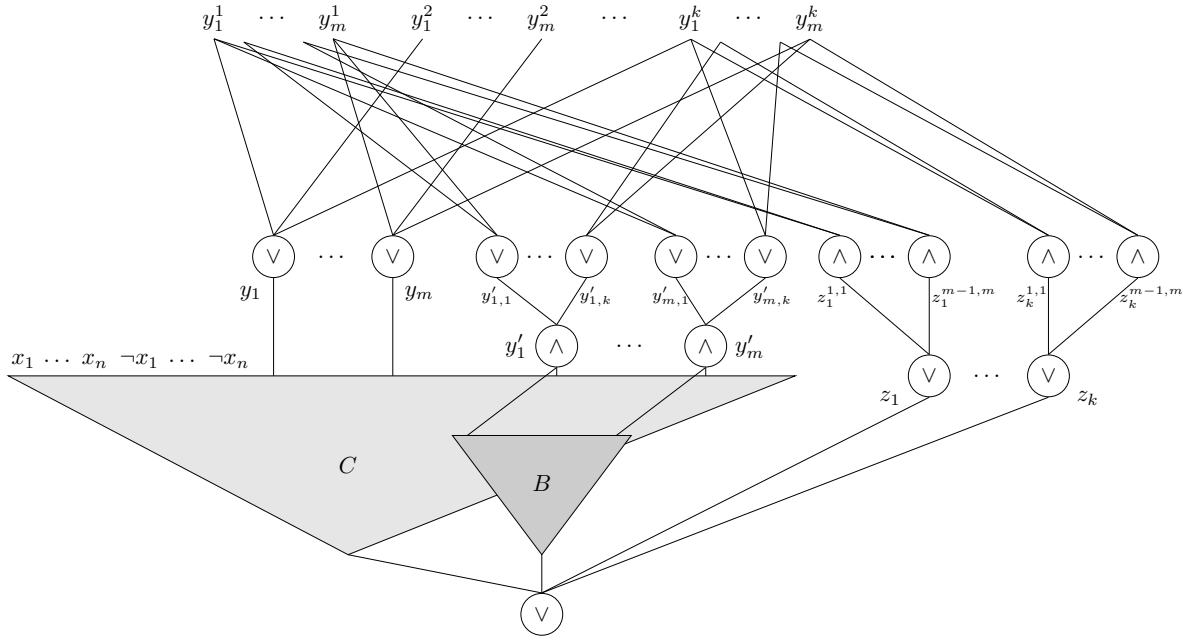


Figure 3: The constructed quantified Boolean circuit C' in the proof of Theorem 6.

cases: either (i) for some $1 \leq i \leq k$ there are some $1 \leq j < j' \leq m$ such that $\beta'(y_j^i) = \beta'(y_{j'}^i) = 1$, or (ii) for each $1 \leq i \leq k$ there is exactly one ℓ_i such that $\beta'(y_{\ell_i}^i) = 1$ and for some $1 \leq i < i' \leq k$ it holds that $\ell_i = \ell_{i'}$, or (iii) for each $1 \leq i \leq k$ there is exactly one ℓ_i such that $\beta'(y_{\ell_i}^i) = 1$ and for each $1 \leq i < i' \leq k$ it holds that $\ell_i \neq \ell_{i'}$. In case (i), we know that the assignment β' sets the node $z_i^{j,j'}$ to true. Therefore, β' sets the node z_i to true, and thus satisfies the circuit $C'[\alpha]$. In case (ii), we know that β' sets y'_j to true for at least $m - k + 1$ different values of j. Therefore, β' satisfies the subcircuit B, and thus satisfies $C'[\alpha]$. Finally, in case (iii), we know that β' sets exactly k different internal nodes y_j to true, and for each $1 \leq j \leq m$ sets the internal node y'_j to true if and only if it sets y_j to false. Then, since $C[\alpha]$ is satisfied by all truth assignments of weight k, we know that β' satisfies the subcircuit C, and thus satisfies $C'[\alpha]$. Since β' was arbitrary, we can conclude that $C'[\alpha]$ is satisfied by all truth assignments $\beta' : Y' \to \{0,1\}$ of weight k.

($\Leftarrow$) Let $\alpha : X \to \{0,1\}$ be an arbitrary truth assignment. Assume that $C'[\alpha]$ is satisfied by all truth assignments $\beta' : Y' \to \{0,1\}$ of weight k. We show that $C[\alpha]$ is satisfied by all truth assignments $\beta : Y \to \{0,1\}$ of weight k. Let $\beta : Y \to \{0,1\}$ be an arbitrary truth assignment of weight k. We now define the truth assignment $\beta' : Y' \to \{0,1\}$ as follows. Let $\{y_{\ell_1}, \ldots, y_{\ell_k}\} = \{\, y_j : 1 \leq j \leq m, \beta(y_j) = 1 \,\}$. For each $1 \leq i \leq k$ and each $1 \leq j \leq m$ we let $\beta'(y_j^i) = 1$ if and only if $j = \ell_i$. Clearly, β' has weight k. Moreover, the assignment β' sets the nodes $z_1, \ldots, z_k$ to false. Furthermore, it is the case that β' sets the internal node y_j

in C' to true for exactly those $1 \leq j \leq m$ for which $\beta(y_j) = 1$, and it sets the internal node y'_j in C' to true for exactly those $1 \leq j \leq m$ for which $\beta(y_j) = 0$. Thus, since β' sets exactly $m - k$ of the internal nodes y'_j to true, we know that β' sets (the output node of) the subcircuit B to false. Therefore, since β' satisfies the circuit $C'[\alpha]$, we can conclude that β' satisfies the subcircuit C, and thus that β satisfies $C[\alpha]$. Since β was arbitrary, we can conclude that $C[\alpha]$ is satisfied by all truth assignments $\beta : Y \to \{0, 1\}$ of weight k. □

4 Additional Characterizations

In this section, we discuss several different characterizations of $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$. In particular, we characterize $\Sigma_2^{\mathrm{p}}[k*]$ using (1) a parameterized model checking problem for first-order logic formulas, (2) a variant of the canonical problem $\Sigma_2^{\mathrm{p}}[k*]$-WSat where the truth assignments to the existential variables are restricted to weight *at most* k (rather than weight exactly k) and (3) using alternating Turing machines with appropriate bounds on the number of alternations and the number of nondeterministic steps. Moreover, for the class $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$, we briefly discuss a characterization in terms of alternating Turing machines that has been shown recently in the literature [28].

4.1 A First-order Model Checking Characterization for $\Sigma_2^{\mathrm{p}}[k*]$

We begin with giving an equivalent characterization of the class $\Sigma_2^{\mathrm{p}}[k*]$ in terms of model checking of first-order logic formulas. Consider the following parameterized model checking problem for first-order logic formulas (with an $\exists\forall$ quantifier prefix) with k existential variables.

$\Sigma_2^{\mathrm{p}}[k*]$-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 $\mathcal{A} \models \varphi$?

We show that this problem is complete for the class $\Sigma_2^{\mathrm{p}}[k*]$.

Theorem 7. $\Sigma_2^{\mathrm{p}}[k*]$-MC *is* $\Sigma_2^{\mathrm{p}}[k*]$*-complete.*

This completeness result follows Lemmas 8 and 9, that we prove below. We begin with showing membership in $\Sigma_2^{\mathrm{p}}[k*]$.

Lemma 8. $\Sigma_2^{\mathrm{p}}[k*]$-MC *is in* $\Sigma_2^{\mathrm{p}}[k*]$.

Proof. We show $\Sigma_2^{\mathrm{p}}[k*]$-membership of $\Sigma_2^{\mathrm{p}}[k*]$-MC by giving an fpt-reduction to $\Sigma_2^{\mathrm{p}}[k*]$-WSat. Let $(\varphi, \mathcal{A})$ be an instance of $\Sigma_2^{\mathrm{p}}[k*]$-MC, where $\varphi = \exists x_1, \ldots, x_k.\forall y_1, \ldots, y_n.\psi$ is a first-order logic sentence over vocabulary τ, and $\mathcal{A}$ is a τ-structure with domain A. We assume without loss of generality that ψ contains only connectives $\wedge$ and $\neg$.

We construct an instance (φ', k) of $\Sigma_2^{\mathrm{p}}[k*]$-WSat, where φ is of the form $\exists X'.\forall Y'.\psi'$. We define:

$$\begin{aligned} X' = &\ \{\, x'_{i,a} : 1 \leq i \leq k, a \in A \,\}, \text{ and} \\ Y' = &\ \{\, y'_{j,a} : 1 \leq j \leq n, a \in A \,\}. \end{aligned}$$

In order to define ψ', we will use the following auxiliary function μ on subformulas of ψ:

$$\mu(\chi) = \begin{cases} \mu(\chi_1) \wedge \mu(\chi_2) & \text{if } \chi = \chi_1 \wedge \chi_2, \\ \neg\mu(\chi_1) & \text{if } \chi = \neg\chi_1, \\ \bigvee\limits_{1 \leq i \leq u} \left(\psi_{z_1, a_1^i} \wedge \cdots \wedge \psi_{z_m, a_m^i} \right) & \text{if } \chi = R(z_1, \ldots, z_m) \text{ and} \\ & R^{\mathcal{A}} = \{(a_1^1, \ldots, a_m^1), \ldots, (a_1^u, \ldots, a_m^u)\}, \end{cases}$$

where for each $z \in X \cup Y$ and each $a \in A$ we define:

$$\psi_{z,a} = \begin{cases} x'_{i,a} & \text{if } z = x_i, \\ y'_{j,a} & \text{if } z = y_j. \end{cases}$$

Now, we define ψ' as follows:

$$\begin{aligned} \psi' = \quad & \psi'_{\text{unique-}X'} \wedge \left(\psi'_{\text{unique-}Y'} \to \mu(\psi)\right), \text{ where} \\ \psi'_{\text{unique-}X'} = \quad & \bigwedge_{1 \leq i \leq k} \left(\bigvee_{a \in A} x'_{i,a} \wedge \bigwedge_{\substack{a,a' \in A \\ a \neq a'}} (\neg x'_{i,a} \vee \neg x'_{i,a'}) \right), \text{ and} \\ \psi'_{\text{unique-}Y'} = \quad & \bigwedge_{1 \leq j \leq n} \left(\bigvee_{a \in A} y'_{j,a} \wedge \bigwedge_{\substack{a,a' \in A \\ a \neq a'}} (\neg y'_{j,a} \vee \neg y'_{j,a'}) \right). \end{aligned}$$

We show that $(\mathcal{A}, \varphi) \in \Sigma_2^{\mathrm{p}}[k*]$-MC if and only if $(\varphi', k) \in \Sigma_2^{\mathrm{p}}[k*]$-WSat.

($\Rightarrow$) Assume that there exists an assignment $\alpha : \{x_1, \ldots, x_k\} \to A$ such that $\mathcal{A}, \alpha \models \forall y_1, \ldots, y_n.\psi$. We define the assignment $\alpha' : X' \to \{0,1\}$ where $\alpha'(x'_{i,a}) = 1$ if and only if $\alpha(x_i) = a$. Clearly, α' has weight k. Also, note that α' satisfies $\psi'_{\text{unique-}X'}$. Now, let $\beta' : Y' \to \{0,1\}$ be an arbitrary assignment. We show that $\alpha' \cup \beta'$ satisfies ψ'. We distinguish two cases: either (i) for each $1 \leq j \leq n$, there is a unique $a_j \in A$ such that $\beta'(y'_{j,a_j}) = 1$, or (ii) this is not the case. In case (i), $\alpha' \cup \beta'$ satisfies $\psi'_{\text{unique-}Y'}$, so we have to show that $\alpha' \cup \beta'$ satisfies $\mu(\psi)$. Define the assignment $\beta : \{y_1, \ldots, y_n\} \to A$ by letting $\beta(y_j) = a_j$. We know that $\mathcal{A}, \alpha \cup \beta \models \psi$. It is now straightforward to show by induction on the structure of ψ that for each subformula χ of ψ holds that that $\alpha' \cup \beta'$ satisfies $\mu(\chi)$ if and only if $\mathcal{A}, \alpha \cup \beta \models \chi$. We then know in particular that $\alpha' \cup \beta'$ satisfies $\mu(\psi)$. In case (ii), we know that $\alpha' \cup \beta'$ does not satisfy $\psi'_{\text{unique-}Y'}$, and therefore $\alpha' \cup \beta'$ satisfies ψ'. This concludes our proof that $(\varphi', k) \in \Sigma_2^{\mathrm{p}}[k*]$-WSat.

($\Leftarrow$) Assume that there exists an assignment $\alpha : X' \to \{0,1\}$ of weight k such that $\forall Y'.\psi'[\alpha]$ is true. Since $\psi'_{\text{unique-}X'}$ contains only variables in X', we know that α satisfies $\psi'_{\text{unique-}X'}$. From this, we can conclude that for each $1 \leq i \leq k$, there is some unique $a_i \in A$ such that $\alpha(x'_{i,a_i}) = 1$. Now, define the assignment $\alpha' : \{x_1, \ldots, x_k\} \to A$ by letting $\alpha'(x_i) = a_i$.

We show that $\mathcal{A}, \alpha' \models \forall y_1, \ldots, y_n.\psi$. Let $\beta' : \{y_1, \ldots, y_n\} \to A$ be an arbitrary assignment. We define $\beta : Y' \to \{0,1\}$ by letting $\beta(y'_{i,a}) = 1$ if and only if $\beta(y_i) = a$. It is straightforward to verify that β satisfies $\psi'_{\text{unique-}Y'}$. We know that $\alpha \cup \beta$ satisfies ψ', so therefore $\alpha \cup \beta$ satisfies $\mu(\psi)$. It is now straightforward to show by induction on the structure of ψ that for each subformula χ of ψ holds that that $\alpha \cup \beta$ satisfies $\mu(\chi)$ if and only if $\mathcal{A}, \alpha' \cup \beta' \models \chi$. We then know in particular that $\mathcal{A}, \alpha' \cup \beta' \models \psi$. This concludes our proof that $(\mathcal{A}, \varphi) \in \Sigma_2^{\mathrm{p}}[k*]$-MC. □

Next, we turn our attention to showing $\Sigma_2^{\mathrm{p}}[k*]$-hardness.

Lemma 9. *$\Sigma_2^{\mathrm{p}}[k*]$-MC is $\Sigma_2^{\mathrm{p}}[k*]$-hard.*

Proof. We show $\Sigma_2^{\mathrm{p}}[k*]$-hardness by giving an fpt-reduction from $\Sigma_2^{\mathrm{p}}[k*]$-WSat(DNF). Let (φ, k) be an instance of $\Sigma_2^{\mathrm{p}}[k*]$-WSat, where $\varphi = \exists X.\forall Y.\psi$, $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$, $\psi = \delta_1 \vee \cdots \vee \delta_u$, and for each $1 \leq \ell \leq u$, $\delta_\ell = l_1^\ell \vee l_2^\ell \vee l_3^\ell$. We construct an instance $(\mathcal{A}, \varphi')$ of $\Sigma_2^{\mathrm{p}}[k*]$-MC. In order to do so, we first fix the following vocabulary τ (which does not depend on the instance (φ, k)): it contains unary relation symbols D, X and Y, and binary relation symbols C_1, C_2, C_3 and O. We construct the domain A of $\mathcal{A}$ as follows:

$$A = X \cup Y \cup \{\, \delta_\ell : 1 \leq \ell \leq u \,\} \cup \{\star\}.$$

Then, we define:

$$\begin{aligned} D^{\mathcal{A}} &= \{\, \delta_\ell : 1 \leq \ell \leq u \,\}; \\ X^{\mathcal{A}} &= X; \\ Y^{\mathcal{A}} &= Y \cup \{\star\}; \\ C_d^{\mathcal{A}} &= \{\, (\delta_\ell, z) : 1 \leq \ell \leq u, z \in X \cup Y, l_d^\ell \in \{x, \neg x\} \,\} \qquad \text{for } 1 \leq d \leq 3; \text{ and} \\ O^{\mathcal{A}} &= \{\, (\delta_\ell, \delta_{\ell'}) : 1 \leq \ell < \ell' \leq u \,\}. \end{aligned}$$

Intuitively, the relations D, X and Y serve to distinguish the various subsets of the domain A. The relations C_d, for $1 \leq d \leq 3$, encode (part of) the structure of the matrix ψ of the formula φ. The relation O encodes a linear ordering on the terms δ_ℓ.

We now define the formula φ' as follows:

$$\varphi' = \exists u_1, \ldots, u_k . \forall v_1, \ldots, v_m . \forall w_1, \ldots, w_u . \chi,$$

where we define χ to be of the following form:

$$\chi = \chi^U_{\text{proper}} \wedge ((\chi^V_{\text{proper}} \wedge \chi^W_{\text{exact}}) \rightarrow \chi_{\text{sat}}).$$

We will define the subformulas of χ below. Intuitively, the assignment of the variables u_i will correspond to an assignment $\alpha : X \rightarrow \{0,1\}$ of weight k that sets a variable $x \in X$ to true if and only if some u_i is assigned to x. Similarly, any assignment of the variables v_i will correspond to an assignment $\beta : Y \rightarrow \{0,1\}$ that sets $y \in Y$ to true if and only if some v_i is assigned to y. The variables w_ℓ will function to refer to the elements $\delta_\ell \in A$.

The formula χ^U_{proper} ensures that the variables $u_1, \ldots, u_k$ select exactly k different elements from X. We define:

$$\chi^U_{\text{proper}} = \bigwedge_{1 \leq i \leq k} X(u_i) \wedge \bigwedge_{1 \leq i < i' \leq k} (u_i \neq u_{i'}).$$

For the sake of clarity, we use the formula χ^V_{proper} to check whether each variable v_i is assigned to a value in $Y \cup \{\star\}$. We define:

$$\chi^V_{\text{proper}} = \bigwedge_{1 \leq i \leq m} Y(v_i).$$

Next, the formula χ^W_{exact} encodes whether the variables $w_1, \ldots, w_u$ get assigned exactly to the elements $\delta_1, \ldots, \delta_u$ (and also in that order, i.e., w_ℓ gets assigned δ_ℓ for each $1 \leq \ell \leq u$). We let:

$$\chi^W_{\text{exact}} = \bigwedge_{1 \leq \ell \leq u} D(w_\ell) \wedge \bigwedge_{1 \leq \ell < \ell' \leq u} O(w_\ell, w_{\ell'}).$$

Finally, we can turn to the formula χ_{sat}, which represents whether the assignments α and β represented by the assignment to the variables u_i and v_j satisfies ψ. We define:

$$\chi_{\text{sat}} = \bigvee_{1 \leq \ell \leq u} \chi^\ell_{\text{sat}},$$

where we let:

$$\chi^\ell_{\text{sat}} = \chi^{\ell,1}_{\text{sat}} \wedge \chi^{\ell,2}_{\text{sat}} \wedge \chi^{\ell,3}_{\text{sat}},$$

and for each $1 \leq d \leq 3$ we let:

$$\chi^{\ell,d}_{\text{sat}} = \begin{cases} \bigvee_{1 \leq j \leq k} C_d(w_\ell, u_j) & \text{if } l^\ell_d = x \in X, \\ \bigwedge_{1 \leq j \leq k} \neg C_d(w_\ell, u_j) & \text{if } l^\ell_d = \neg x \text{ for some } x \in X, \\ \bigvee_{1 \leq j \leq m} C_d(w_\ell, v_j) & \text{if } l^\ell_d = y \in Y, \\ \bigwedge_{1 \leq j \leq m} \neg C_d(w_\ell, v_j) & \text{if } l^\ell_d = \neg y \text{ for some } y \in X. \end{cases}$$

Intuitively, for each $1 \leq \ell \leq u$ and each $1 \leq d \leq 3$, the formula $\chi^{\ell,d}_{\text{sat}}$ will be satsified by the assignments to the variables u_i and v_i if and only if the corresponding assignments α to X and β to Y satisfy l^ℓ_d.

It is straightforward to verify that the instance $(\mathcal{A}, \varphi')$ can be constructed in polynomial time. We show that $(\varphi, k) \in \Sigma_2^{\mathrm{p}}[k*]$-WSat(3DNF) if and only if $(\mathcal{A}, \varphi') \in \Sigma_2^{\mathrm{p}}[k*]$-MC.

($\Rightarrow$) Assume that there exists an assignment $\alpha : X \to \{0,1\}$ of weight k such that $\forall Y.\psi[\alpha]$ is true. We show that $\mathcal{A} \models \varphi'$. Let $\{ x \in X : \alpha(x) = 1 \} = \{x_{i_1}, \ldots, x_{i_k}\}$. We define the assignment $\mu : \{u_1, \ldots, u_k\} \to A$ by letting $\mu(x_j) = x_{i_j}$ for all $1 \leq j \leq k$. It is straightforward to verify that $\mathcal{A}, \mu \models \psi^U_{\mathrm{proper}}$. Now, let $\nu : \{y_1, \ldots, y_m, w_1, \ldots, w_u\} \to A$ be an arbitrary assignment. We need to show that $\mathcal{A}, \mu \cup \nu \models \chi$, and we thus need to show that $\mathcal{A}, \mu \cup \nu \models (\chi^V_{\mathrm{proper}} \wedge \chi^W_{\mathrm{exact}}) \to \chi_{\mathrm{sat}}$. We distinguish several cases: either (i) $\nu(y_i) \notin Y \cup \{\star\}$ for some $1 \leq i \leq m$, or (ii) the above is not the case and $\nu(w_\ell) \neq \delta_\ell$ for some $1 \leq \ell \leq u$, or (iii) neither of the above is the case. In case (i), it is straightforward to verify that $\mathcal{A}, \mu \cup \nu \models \neg\chi^V_{\mathrm{proper}}$. In case (ii), it is straightforward to verify that $\mathcal{A}, \mu \cup \nu \models \neg\chi^W_{\mathrm{exact}}$. Consider case (iii). We construct the assignment $\beta : Y \to \{0,1\}$ by letting $\beta(y) = 1$ if and only if $\nu(v_i) = y$ for some $1 \leq i \leq m$. We know that $\alpha \cup \beta$ satisfies ψ, and thus in particular that $\alpha \cup \beta$ satisfies some term δ_ℓ. It is now straightforward to verify that $\mathcal{A}, \mu \cup \nu \models \chi^\ell_{\mathrm{sat}}$, and thus that $\mathcal{A}, \mu \cup \nu \models \chi_{\mathrm{sat}}$. This concludes our proof that $\mathcal{A} \models \varphi'$.

($\Leftarrow$) Assume that $\mathcal{A} \models \varphi'$. We show that $(\varphi, k) \in \Sigma_2^{\mathrm{p}}[k*]$-WSat(3DNF). We know that there exists an assignment $\mu : \{u_1, \ldots, u_k\} \to A$ such that $\mathcal{A}, \mu \models \forall u_1, \ldots, u_m.\forall w_1, \ldots, w_u.\chi$. Since $\mathcal{A}, \mu \models \chi^U_{\mathrm{proper}}$, we know that μ assigns the variables u_i to k different values $x \in X$. Define $\alpha : X \to \{0,1\}$ by letting $\alpha(x) = 1$ if and only if $\mu(u_i) = x$ for some $1 \leq i \leq k$. Clearly, α has weight k. Now, let $\beta : Y \to \{0,1\}$ be an arbitrary assignment. Construct the assignment $\nu : \{v_1, \ldots, v_m, w_1, \ldots, w_u\}$ as follows. For each $1 \leq i \leq m$, we let $\nu(v_i) = y_i$ if $\beta(y_i) = 1$, and we let $\nu(v_i) = \star$ otherwise. Also, for each $1 \leq \ell \leq u$, we let $\nu(w_\ell) = \delta_\ell$. It is straightforward to verify that $\mathcal{A}, \mu \cup \nu \models \chi^V_{\mathrm{proper}} \wedge \chi^W_{\mathrm{exact}}$. Therefore, we know that $\mathcal{A}, \mu \cup \nu \models \chi_{\mathrm{sat}}$, and thus that for some $1 \leq \ell \leq u$ it holds that $\mathcal{A}, \mu \cup \nu \models \chi^\ell_{\mathrm{sat}}$. It is now straightforward to verify that $\alpha \cup \beta$ satisfies δ_ℓ. Since β was arbitrary, this concludes our proof that $(\varphi, k) \in \Sigma_2^{\mathrm{p}}[k*]$-WSat. □

The problem $\Sigma_2^{\mathrm{p}}[k*]$-MC takes the relational vocabulary τ over which the structure $\mathcal{A}$ and the first-order logic sentence φ are defined as part of the input. However, the proof of Lemma 9 shows that the problem $\Sigma_2^{\mathrm{p}}[k*]$-MC is $\Sigma_2^{\mathrm{p}}[k*]$-hard already when the vocabulary τ is fixed and contains only unary and binary relation symbols.

Corollary 10. *The problem $\Sigma_2^{\mathrm{p}}[k*]$-MC is $\Sigma_2^{\mathrm{p}}[k*]$-hard even when the vocabulary τ is fixed and contains only unary and binary relation symbols.*

4.2 Another Weighted Satisfiability Characterization for $\Sigma_2^{\mathrm{p}}[k*]$

Next, we show that for the canonical problem $\Sigma_2^{\mathrm{p}}[k*]$-WSat, it does not matter whether we require the weight of truth assignments to the existential variables to be exactly k or at most k. That is, the variant of $\Sigma_2^{\mathrm{p}}[k*]$-WSat where truth assignments to the existentially quantified variables are restricted to weight at most k is also $\Sigma_2^{\mathrm{p}}[k*]$-complete. We will use this result as a technical lemma for the results in Section 4.3.

Formally, we consider the problem $\Sigma_2^{\mathrm{p}}[k*]$-WSat$^{\leq k}$, that is defined as follows.

$\Sigma_2^{\mathrm{p}}[k*]$-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 for all truth assignments β to Y the assignment $\alpha \cup \beta$ satisfies ψ?

We show that $\Sigma_2^{\mathrm{p}}[k*]$-WSat$^{\leq k}$ is $\Sigma_2^{\mathrm{p}}[k*]$-complete.

Proposition 11. *$\Sigma_2^{\mathrm{p}}[k*]$-WSat$^{\leq k}$ is $\Sigma_2^{\mathrm{p}}[k*]$-complete.*

Proof. Firstly, to show membership in $\Sigma_2^{\mathrm{p}}[k*]$, we give an fpt-reduction from $\Sigma_2^{\mathrm{p}}[k*]$-WSat$^{\leq k}$ to $\Sigma_2^{\mathrm{p}}[k*]$-WSat. Let (φ, k) be an instance of $\Sigma_2^{\mathrm{p}}[k*]$-WSat$^{\leq k}$, with $\varphi = \exists X.\forall Y.\psi$. We construct an instance (φ', k) of $\Sigma_2^{\mathrm{p}}[k*]$-WSat. Let $X' = \{x'_1, \ldots, x'_k\}$ be a set of fresh variables. Now define $\varphi' = \exists X \cup X'.\forall Y.\psi$. We show that $(\varphi, k) \in \Sigma_2^{\mathrm{p}}[k*]$-WSat$^{\leq k}$ if and only if $(\varphi', k) \in \Sigma_2^{\mathrm{p}}[k*]$-WSat.

($\Rightarrow$) Assume that $(\varphi, k) \in \Sigma_2^{\mathrm{p}}[k*]\text{-WSAT}^{\leq k}$. This means that there exists an assignment $\alpha : X \to \{0,1\}$ of weight $\ell \leq k$ such that $\forall Y.\psi[\alpha]$ evaluates to true. Define the assignment $\alpha' : X' \to \{0,1\}$ as follows. We let $\alpha'(x'_i) = 1$ if and only if $1 \leq i \leq k - \ell$. Then the assignment $\alpha \cup \alpha'$ has weight k, and $\forall Y.\psi[\alpha \cup \alpha']$ evaluates to true. Therefore, $(\varphi', k) \in \Sigma_2^{\mathrm{p}}[k*]\text{-WSAT}$.

($\Leftarrow$) Assume that $(\varphi', k) \in \Sigma_2^{\mathrm{p}}[k*]\text{-WSAT}$. This means that there exists an assignment $\alpha : X \cup X' \to \{0,1\}$ of weight k such that $\forall Y.\psi[\alpha]$ evaluates to true. Now let α' be the restriction of α to the set X of variables. Clearly, α' has weight at most k. Also, since ψ contains no variables in X', we know that $\forall Y.\psi[\alpha']$ evaluates to true. Therefore, $(\varphi, k) \in \Sigma_2^{\mathrm{p}}[k*]\text{-WSAT}^{\leq k}$.

Then, to show $\Sigma_2^{\mathrm{p}}[k*]$-hardness, we give an fpt-reduction from $\Sigma_2^{\mathrm{p}}[k*]\text{-WSAT}$ to $\Sigma_2^{\mathrm{p}}[k*]\text{-WSAT}^{\leq k}$. Let (φ, k) be an instance of $\Sigma_2^{\mathrm{p}}[k*]\text{-WSAT}$, where $\varphi = \exists X.\forall Y.\psi$, and $X = \{x_1, \ldots, x_n\}$. We construct an instance (φ', k') of $\Sigma_2^{\mathrm{p}}[k*]\text{-WSAT}^{\leq k}$. Let $C = \{\, c_{j,i} : 1 \leq i \leq k, 1 \leq j \leq n \,\}$ be a set of fresh propositional variables. Intuitively, we can think of the variables $c_{j,i}$ as being placed in a matrix with k columns and n rows: variable $c_{j,i}$ is positioned in the i-th column and in the j-th row. We ensure that in each column, exactly one variable is set to true (see ψ_{col} below), and that in each row, at most one variable is set to true (see ψ_{row} below). This way, any satisfying assignment must set exactly k variables in the matrix to true, in different rows. Next, we ensure that if any variable in the j-th row is set to true, that x_j is set to true (see ψ_{corr} below). This way, we know that exactly k variables x_j must be set to true in any satisfying assignment.

Formally, we define:

$$
\begin{aligned}
\varphi' &= \exists X \cup C.\forall Y.\psi';\\
k' &= 2k;\\
\psi' &= \psi_{\mathrm{col}} \wedge \psi_{\mathrm{row}} \wedge \psi_{\mathrm{corr}} \wedge \psi;\\
\psi_{\mathrm{col}} &= \bigwedge_{1 \leq j \leq n} \left(\bigvee_{1 \leq i \leq k} c_{j,i} \wedge \bigwedge_{1 \leq i < i' \leq k} (\neg c_{j,i} \vee \neg c_{j,i'}) \right);\\
\psi_{\mathrm{row}} &= \bigwedge_{1 \leq i \leq k} \bigwedge_{1 \leq j < j' \leq n} (\neg c_{j,i} \vee \neg c_{j',i}); \text{ and}\\
\psi_{\mathrm{corr}} &= \bigwedge_{1 \leq i \leq k} \bigwedge_{1 \leq j \leq n} (c_{j,i} \to x_j).
\end{aligned}
$$

Any assignment $\alpha : X \cup C \to \{0,1\}$ that satisfies $\psi_{\mathrm{col}} \wedge \psi_{\mathrm{row}}$ must set the variables $c_{j_1,1}, \ldots, c_{j_k,k}$ to true, for some $1 \leq j_1 < \cdots < j_k \leq n$. Furthermore, if α satisfies ψ_{corr}, it must also set $x_{j_1}, \ldots, x_{j_k}$ to true.

It is now easy to show that $(\varphi, k) \in \Sigma_2^{\mathrm{p}}[k*]\text{-WSAT}$ if and only if $(\varphi', k') \in \Sigma_2^{\mathrm{p}}[k*]\text{-WSAT}^{\leq k}$. Let $\alpha : X \to \{0,1\}$ be an assignment of weight k such that $\forall Y.\psi[\alpha]$ is true, where $\{\, x_i : 1 \leq i \leq n, \alpha(x_i) = 1 \,\} = \{x_{j_1}, \ldots, x_{j_k}\}$. Then consider the assignment $\gamma : C \to \{0,1\}$ where $\gamma(c_{j,i}) = 1$ if and only if $j = j_i$. Then the assignment $\alpha \cup \gamma$ has weight k', and has the property that $\forall Y.\psi'[\alpha \cup \gamma]$ is true.

Conversely, let $\gamma : X \cup C \to \{0,1\}$ be an assignment of weigth k' such that $\forall Y.\psi'[\gamma]$ is true. Then the restriction α of γ to the variables X has weight k, and has the property that $\forall Y.\psi[\alpha]$ is true. □

4.3 An Alternating Turing Machine Characterization for $\Sigma_2^{\mathrm{p}}[k*]$

In this section, we give a characterization of the class $\Sigma_2^{\mathrm{p}}[k*]$ by means of alternating Turing machines (ATMs). This characterization states that the class $\Sigma_2^{\mathrm{p}}[k*]$ is the class of parameterized problems that can be solved by an ATM with appropriate bounds on the type and number of nondeterministic steps used in the computation. A similar characterization has recently been developed for the class $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$ [28]. Additionally, we characterize the class $\Sigma_2^{\mathrm{p}}[k*]$ by showing that the halting problem for alternating Turing machines with appropriate bounds on the type and number of nondeterministic steps used is $\Sigma_2^{\mathrm{p}}[k*]$-complete.

We consider two particular types of ATMs. An $\exists\forall$-*Turing machine* (or simply $\exists\forall$-machine) is a 2-alternating ATM $(S_\exists, S_\forall, \Sigma, \Delta, s_0, F)$, where $s_0 \in S_\exists$.

Definition 12. *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:*

- *there is an accepting run of $\mathbb{M}$ with input ϵ,*

- *each computation path of $\mathbb{M}$ contains at most ℓ existential configurations and at most t universal configurations.*

Definition 13. *Let P be a parameterized problem. A $\Sigma_2^{\mathrm{p}}[k*]$-machine for P is an $\exists\forall$-machine $\mathbb{M}$ such that there exists a computable function f and a polynomial p such that*

- *$\mathbb{M}$ decides P in time $f(k) \cdot p(|x|)$; and*
- *and 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.*

Let $m \in \mathbb{N}$ be a positive integer. We consider the following parameterized problem.

$\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^m
Instance: An $\exists\forall$-machine $\mathbb{M}$ with m tapes, and positive integers $k, t \geq 1$.
Parameter: k.
Question: Does $\mathbb{M}$ halt on the empty string with existential cost k and universal cost t?

Moreover, we consider the following parameterized problem.

$\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT*
Instance: A positive integer m, an $\exists\forall$-machine $\mathbb{M}$ with m tapes, and positive integers $k, t \geq 1$.
Parameter: k.
Question: Does $\mathbb{M}$ halt on the empty string with existential cost k and universal cost t?

Note that for $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^m, the number m of tapes of the $\exists\forall$-machines in the input is a fixed constant, whereas for $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT*, the number of tapes is given as part of the input.

The parameterized complexity class $\Sigma_2^{\mathrm{p}}[k*]$ can then be characterized by alternating Turing machines as follows. These results can be seen as an analogue to the Cook-Levin Theorem for the complexity class $\Sigma_2^{\mathrm{p}}[k*]$.

Theorem 14. *The problem $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT* is $\Sigma_2^{\mathrm{p}}[k*]$-complete, and so is the problem $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^m for each $m \in \mathbb{N}$.*

Theorem 15. *$\Sigma_2^{\mathrm{p}}[k*]$ is exactly the class of parameterized decision problems P that are decidable by some $\Sigma_2^{\mathrm{p}}[k*]$-machine.*

Proof of Theorems 14 and 15. In order to show these results, we will use the following statements. We show how the results follow from these statements. We then present the statements (with a detailed proof) as Propositions 18 and 20–22.

(i) Let A and B be parameterized problems. If B is decidable by some $\Sigma_2^{\mathrm{p}}[k*]$-machine with m tapes, and if $A \leq_{\mathrm{fpt}} B$, then A is decidable by some $\Sigma_2^{\mathrm{p}}[k*]$-machine with m tapes (Proposition 18).

(ii) $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT$^* \leq_{\mathrm{fpt}} \Sigma_2^{\mathrm{p}}[k*]$-MC (Proposition 20).

(iii) For any parameterized problem P that is decidable by some $\Sigma_2^{\mathrm{p}}[k*]$-machine with m tapes, it holds that $P \leq_{\mathrm{fpt}} \Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^{m+1} (Proposition 21).

(iv) There is an $\Sigma_2^{\mathrm{p}}[k*]$-machine with a single tape that decides $\Sigma_2^{\mathrm{p}}[k*]$-WSAT$^{\leq k}$ (Proposition 22).

In addition to these statements, we will need one result known from the literature (Corollary 17, which follows from Proposition 16).

(v) $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT$^2 \leq_{\mathrm{fpt}} \Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^1 (Corollary 17).

To see that these statements imply the desired results, observe the following.

Together, (iii) and (iv) imply that $\Sigma_2^{\mathrm{p}}[k*]$-WSAT$^{\leq k} \leq_{\mathrm{fpt}} \Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^2. Clearly, for all $m \geq 2$, $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT$^2 \leq_{\mathrm{fpt}} \Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^m. This gives us $\Sigma_2^{\mathrm{p}}[k*]$-hardness of $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^m, for all $m \geq 2$. $\Sigma_2^{\mathrm{p}}[k*]$-hardness of $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^1 follows from Corollary 17, which implies that there is an fpt-reduction from $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^2 to $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^1. This also implies that $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT* is $\Sigma_2^{\mathrm{p}}[k*]$-hard. Then, by (ii), and since $\Sigma_2^{\mathrm{p}}[k*]$-MC is in $\Sigma_2^{\mathrm{p}}[k*]$ by Theorem 7, we obtain $\Sigma_2^{\mathrm{p}}[k*]$-completeness of $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT* and $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^m, for each $m \geq 1$. This proves Theorem 14. It remains to prove Theorem 15.

By (ii) and (iii), and by transitivity of fpt-reductions, we have that any parameterized problem P that is decided by an $\Sigma_2^{\mathrm{p}}[k*]$-machine is fpt-reducible to $\Sigma_2^{\mathrm{p}}[k*]$-WSAT, and thus is in $\Sigma_2^{\mathrm{p}}[k*]$. Conversely, let P be any parameterized problem in $\Sigma_2^{\mathrm{p}}[k*]$. Then, by $\Sigma_2^{\mathrm{p}}[k*]$-hardness of $\Sigma_2^{\mathrm{p}}[k*]$-WSAT$^{\leq k}$, we know that $P \leq_{\mathrm{fpt}} \Sigma_2^{\mathrm{p}}[k*]$-WSAT$^{\leq k}$. By (i) and (iv), we know that P is decidable by some $\Sigma_2^{\mathrm{p}}[k*]$-machine with a single tape. From this we conclude that $\Sigma_2^{\mathrm{p}}[k*]$ is exactly the class of parameterized problems P decided by some $\Sigma_2^{\mathrm{p}}[k*]$-machine. □

Firstly, we state the result known from the literature that we used in the proof of Theorems 14 and 15.

Proposition 16 ([36, Theorems 8.9 and 8.10])**.** *Let $m \geq 1$ be a (fixed) positive integer. For each ATM $\mathbb{M}$ with m tapes, there exists an ATM $\mathbb{M}'$ with 1 tape such that:*

- *$\mathbb{M}$ and $\mathbb{M}'$ are equivalent, i.e., they accept the same language;*
- *$\mathbb{M}'$ simulates n steps of $\mathbb{M}$ using $O(n^2)$ steps; and*
- *$\mathbb{M}'$ simulates existential steps of $\mathbb{M}$ using existential steps, and simulates universal steps of $\mathbb{M}$ using universal steps.*

Corollary 17. $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT$^2 \leq_{\mathrm{fpt}} \Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^1.

Next, we give detailed proofs of the statements (i)–(iv) that were used in the proof of Theorems 14 and 15 (Propositions 18 and 20–22). We begin with proving the first statement.

Proposition 18. *Let A and B be parameterized problems, and let $m \in \mathbb{N}$ be a positive integer. If B is decided by some $\Sigma_2^{\mathrm{p}}[k*]$-machine with m tapes and if $A \leq_{\mathrm{fpt}} B$, then A is decided by some $\Sigma_2^{\mathrm{p}}[k*]$-machine with m tapes.*

Proof. Let R be the fpt-reduction from A to B, and let M be an algorithm that decides B and that can be implemented by an $\Sigma_2^{\mathrm{p}}[k*]$-machine with m tapes. Clearly, the composition of R and M is an algorithm that decides A—by Proposition 16, we may assume that the fpt-reduction R is implemented by a deterministic Turing machine with 1 tape. It is straightforward to verify that the composition of R and M can be implemented by an $\Sigma_2^{\mathrm{p}}[k*]$-machine with m tapes. □

In order to prove the second statement that we used in the proof of Theorems 14 and 15, we prove the following technical lemma.

Lemma 19. *Let $\mathbb{M}$ be an $\exists\forall$-machine with m tapes and let $k, t \in \mathbb{N}$. We can construct an $\exists\forall$-machine $\mathbb{M}'$ with m tapes (in time polynomial in $|(\mathbb{M}, k, t)|$) such that the following are equivalent:*

- *there is an accepting run ρ of $\mathbb{M}'$ with input ϵ and each computation path in ρ contains* exactly *k existential configurations and* exactly *t universal configurations*
- *$\mathbb{M}$ halts on ϵ with existential cost k and universal cost t.*

Proof. Let $\mathbb{M} = (S_\exists, S_\forall, \Sigma, \Delta, s_0, F)$ be an $\exists\forall$-machine with m tapes. Now construct $\mathbb{M} = (S'_\exists, S'_\forall, \Sigma, \Delta', s_0, F')$ as follows:

$$
\begin{aligned}
S'_\exists &= \{\, s_i : s \in S_\exists, 1 \leq i \leq k+t \,\},\\
S'_\forall &= \{\, s_i : s \in S_\forall, 1 \leq i \leq k+t \,\},\\
\Delta' &= \{\, (s_i, \overline{a}, s'_{i+1}, \overline{a}', \overline{d}) : (s, \overline{a}, s', \overline{a}', \overline{d}) \in \Delta, 1 \leq i \leq k+t-1 \,\} \cup\\
&\qquad \{\, (s_i, \overline{a}, s_{i+1}, \overline{a}, \mathbf{S}^m) : s \in S_\exists \cup S_\forall, \overline{a} \in \Sigma^m \,\}, \text{ and}\\
F' &= \{\, f_{k+t} : f \in F \,\}.
\end{aligned}
$$

To see that $\mathbb{M}'$ satisfies the required properties, it suffices to see that for each (accepting) computation path $C_1 \to \ldots \to C_{k'+t'}$ of $\mathbb{M}$ with input ϵ that contains existential configurations $C_1, \ldots, C_{k'}$ and universal configurations $C_{k'+1}, \ldots, C_{k'+t'}$ for $1 \leq k' \leq k$ and $1 \leq t' \leq t$, it holds that

$$C_1^1 \to \ldots \to C_{k'}^{k'} \to C_{k'}^{k'+1} \to \ldots \to C_{k'}^{k} \to C_{k'+1}^{k+1} \to \ldots \to C_{k'+t'}^{k+t'} \to C_{k'+t'}^{k+t'+1} \to \ldots \to C_{k'+t'}^{k+t}$$

is an (accepting) computation path of $\mathbb{M}'$ with input ϵ, where for each $1 \leq i \leq k+t$ and each $1 \leq j \leq k'+t'$ we let C_j^i be the configuration $(s_i, x_1, p_1, \ldots, x_m, p_m)$, where $C_j = (s, x_1, p_1, \ldots, x_m, p_m)$. □

With this technical lemma in place, we can now prove the second statement that we used in the proof of Theorems 14 and 15.

Proposition 20. $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT* $\leq_{\mathrm{fpt}}$ $\Sigma_2^{\mathrm{p}}[k*]$-MC

Proof. Let $(\mathbb{M}, k, t)$ be an instance of $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT*, where $\mathbb{M} = (S_\exists, S_\forall, \Sigma, \Delta, s_0, F)$ is an $\exists\forall$-machine with m tapes, and k and t are positive integers. We constuct in fpt-time an instance $(\mathcal{A}, \varphi)$ of $\Sigma_2^{\mathrm{p}}[k*]$-MC, such that $(\mathbb{M}, k, t) \in \Sigma_2^{\mathrm{p}}[k*]$-TM-HALT* if and only if $(\mathcal{A}, \varphi) \in \Sigma_2^{\mathrm{p}}[k*]$-MC. By Lemma 19, it suffices to construct $(\mathcal{A}, \varphi)$ in such a way that $(\mathcal{A}, \varphi) \in \Sigma_2^{\mathrm{p}}[k*]$-MC if and only if there exists an accepting run ρ of $\mathbb{M}$ with input ϵ such that each computation path of ρ contains exactly k existential configurations and exactly t universal configurations.

We construct $\mathcal{A}$ to be a τ-structure with a domain A. We will define the vocabulary τ below. The domain A of $\mathcal{A}$ is defined as follows:

$$A = S_\exists \cup S_\forall \cup \Sigma \cup \{\$, \square\} \cup \{\mathbf{L}, \mathbf{R}, \mathbf{S}\} \cup \{0, \ldots, \max\{m, k+t-1\}\} \cup T,$$

where T is the set of tuples $(a_1, \ldots, a_m) \in (\Sigma \cup \{\$, \square\})^m$ and of tuples $(d_1, \ldots, d_m) \in \{\mathbf{L}, \mathbf{R}, \mathbf{S}\}^m$ occurring in transitions of Δ. Observe that $|A| = O(k + t + |\mathbb{M}|)$.

We now describe the relation symbols in τ and their interpretation in $\mathcal{A}$. The vocabulary τ contains the 5-ary relation symbol D (intended as "transition relation"), and the ternary relation symbol P (intended as "projection relation"), with the following interpretations:

$$\begin{aligned} D^{\mathcal{A}} &= \Delta, \text{ and} \\ P^{\mathcal{A}} &= \{\, (j, \bar{b}, b_j) : 1 \leq j \leq m, \bar{b} \in T, \bar{b} = (b_1, \ldots, b_m) \,\}. \end{aligned}$$

Moreover, τ contains the unary relation symbols R_{tape}, R_{cell}, R_{blank}, R_{end}, R_{symbol}, R_{init}, R_{acc}, R_{left}, R_{right}, R_{stay}, $R_\exists$, $R_\forall$, R_i for each $1 \leq i \leq k+t-1$, and R_a for each $a \in \Sigma$, which are interpreted in $\mathcal{A}$ as follows:

$$\begin{gathered} R_{\mathrm{tape}}^{\mathcal{A}} = \{1, \ldots, m\}, R_{\mathrm{cell}}^{\mathcal{A}} = \{1, \ldots, k+t\}, R_{\mathrm{blank}}^{\mathcal{A}} = \{\square\}, R_{\mathrm{end}}^{\mathcal{A}} = \{\$\}, R_{\mathrm{symbol}}^{\mathcal{A}} = \Sigma, \\ R_{\mathrm{init}}^{\mathcal{A}} = \{s_0\}, R_{\mathrm{acc}}^{\mathcal{A}} = F, R_{\mathrm{left}}^{\mathcal{A}} = \{\mathbf{L}\}, R_{\mathrm{right}}^{\mathcal{A}} = \{\mathbf{R}\}, R_{\mathrm{stay}}^{\mathcal{A}} = \{\mathbf{S}\}, R_\exists^{\mathcal{A}} = S_\exists, R_\forall^{\mathcal{A}} = S_\forall, \\ R_i^{\mathcal{A}} = \{i\} \text{ for each } 1 \leq i \leq k+t-1, \text{and } R_a^{\mathcal{A}} = \{a\} \text{ for each } a \in \Sigma. \end{gathered}$$

The formula φ that we will construct contains variables $z_\square, z_\$, z_{\mathrm{init}}, z_{\mathrm{left}}, z_{\mathrm{right}}, z_{\mathrm{stay}}, z_1, \ldots, z_{k+t}, z_{a_1}, \ldots, z_{a_{|\Sigma|}}$, where $\Sigma = \{a_1, \ldots, a_{|\Sigma|}\}$, that we will use to refer to elements of the singleton relations of $\mathcal{A}$. We define a formula $\psi_{\mathrm{constants}}$ that is intended to provide a fixed interpretation of some variables that we can use to refer to the elements of the singleton relations of $\mathcal{A}$:

$$\begin{aligned} \psi_{\mathrm{constants}} = {} & R_{\mathrm{end}}(z_\$) \wedge R_{\mathrm{blank}}(z_\square) \wedge R_{\mathrm{left}}(z_{\mathrm{left}}) \wedge R_{\mathrm{right}}(z_{\mathrm{right}}) \wedge \\ & R_{\mathrm{stay}}(z_{\mathrm{stay}}) \wedge \bigwedge_{0 \leq i \leq k+t-1} R_i(z_i) \wedge \bigwedge_{a \in \Sigma} R_a(z_a). \end{aligned}$$

The formula φ that we will construct aims to express that there exist k transitions (from existential states), such that for any sequence of $t-1$ transitions (from universal states), the entire sequence of transitions results in an accepting state. It will contain variables $s_i, t_i, s_i', t_i', d_i$, for $1 \leq i \leq k+t-1$.

The formula φ will also contain variables $p_{i,j}$ and $q_{i,j,\ell}$, for each $k+1 \leq i \leq k+t$, each $1 \leq j \leq m$ and each $1 \leq \ell \leq k+t$. The variables $p_{i,j}$ will encode the position of the tape head for tape j at the i-th

configuration in the computation path, and the variables $q_{i,j,\ell}$ will encode the symbol that is at cell ℓ of tape j at the i-th configuration in the computation path.

The position of the tape heads and the contents of the tapes for configurations 1 to k in the computation path, will not be encoded by means of variables, but by means of the formulas $\psi_{\text{symbol},i}$ and $\psi_{\text{position},i}$, which we define below. Intuitively, the reason for this is that the number of existentially quantified variables in the formula φ has to be bounded by a function of k, and the total amount of information that we need to encode is not bounded by any function of k. The size of subformulas of φ does not need to be bounded by a function of k, so we can encode this information using formulas $\psi_{\text{symbol},i}$ and $\psi_{\text{position},i}$. However, the size of the formulas $\psi_{\text{symbol},i}$ and $\psi_{\text{position},i}$ grows exponentially in i. Therefore, we can only use this encoding for configurations up to k. This is the reason why we use variables $p_{i,j}$ and $q_{i,j,\ell}$ to encode the configurations $k+1 \leq i \leq k+t$.

We define φ as follows:

$$\begin{aligned}
\varphi = {} & \exists s_1, t_1, s'_1, t'_1, d_1, \ldots, s_k, t_k, s'_k, t'_k, d_k. \\
& \forall z_0, z_\square, z_\$, z_{\text{init}}, z_{\text{left}}, z_{\text{right}}, z_{\text{stay}}, z_1, \ldots, z_{k+t}, z_{a_1}, \ldots, z_{a_{|\Sigma|}}. \\
& \forall s_{k+1}, t_{k+1}, s'_{k+1}, t'_{k+1}, d_{k+1}, \ldots, s_{k+t-1}, t_{k+t-1}, s'_{k+t-1}, t'_{k+t-1}, d_{k+t-1}. \\
& \forall p_{k+1,1}, \ldots, p_{k+t,m}. q_{k+1,1,1}, \ldots, q_{k+t,m,k+t}. \psi, \\
\psi = {} & \psi_{\text{constants}} \rightarrow (\psi_{\exists\text{-states}} \wedge \psi_{\exists\text{-tapes}} \wedge ((\psi_{\forall\text{-states}} \wedge \psi_{\forall\text{-tapes}}) \rightarrow \psi_{\text{accept}})), \\
\psi_{\exists\text{-states}} = {} & (s_1 = z_{\text{init}}) \wedge \bigwedge_{1 \leq i \leq k} D(s_i, t_i, s'_i, t'_i, d_i) \wedge \bigwedge_{1 \leq i \leq k-1} ((s_{i+1} = s'_i) \wedge R_\exists(s_{i+1})), \\
\psi_{\forall\text{-states}} = {} & \bigwedge_{k+1 \leq i \leq k+t-1} D(s_i, t_i, s'_i, t'_i, d_i) \wedge \bigwedge_{k \leq i \leq k+t-2} ((s_{i+1} = s'_i) \wedge R_\forall(s_{i+1})), \text{ and} \\
\psi_{\text{accept}} = {} & R_{\text{acc}}(s'_{k+t-1}),
\end{aligned}$$

where we define the formulas $\psi_{\exists\text{-tapes}}$ and $\psi_{\forall\text{-tapes}}$ below. In order to do so, for each $1 \leq i \leq k+1$ we define the quantifier-free formulas

$$\psi_{\text{symbol},i}(w, p, a, \overline{v}_i) \qquad \text{and} \qquad \psi_{\text{position},i}(w, p, \overline{v}_i),$$

with $\overline{v}_i = s_1, t_1, s'_1, t'_1, d_1, \ldots, s_{i-1}, t_{i-1}, s'_{i-1}, t'_{i-1}, d_{i-1}$. Intuitively:

- $\psi_{\text{symbol},i}(w, p, a, \overline{v}_i)$ represents whether the p-th cell of the w-th tape contains the symbol a, whenever the sequence of transitions in $\overline{v}_i$ has been carried out starting with empty tapes; and
- $\psi_{\text{position},i}(w, p, \overline{v}_i)$ represents whether the head of the w-th tape is at position p, whenever the sequence of transitions in $\overline{v}_i$ has been carried out starting with empty tapes.

In particular, for $i = 1$, we define formulas $\psi_{\text{symbol},1}(w, p, a)$ and $\psi_{\text{position},1}(w, p)$, because $\overline{v}_1$ is the empty sequence.

We define $\psi_{\text{symbol},i}(w, p, a, \overline{v}_i)$ and $\psi_{\text{position},i}(w, p, \overline{v}_i)$ simultaneously by induction on i as follows:

$$\begin{aligned}
\psi_{\text{symbol},1}(w, p, a) = {} & R_{\text{tape}}(w) \wedge R_{\text{cell}}(p) \wedge \\
& (p = z_0 \rightarrow a = z_\$) \wedge (p \neq z_0 \rightarrow a = z_\square), \\
\psi_{\text{position},1}(w, p) = {} & R_{\text{tape}}(w) \wedge (p = z_1), \\
\psi_{\text{symbol},i+1}(w, p, a, \overline{v}_{i+1}) = {} & R_{\text{tape}}(w) \wedge R_{\text{cell}}(p) \wedge \\
& ((\psi_{\text{position},i}(w, p, \overline{v}_i) \wedge P(w, t'_i, a)) \vee \\
& (\neg\psi_{\text{position},i}(w, p, \overline{v}_i) \wedge \psi_{\text{symbol},i}(w, p, a, \overline{v}_i))), \\
\psi_{\text{position},i+1}(w, p, \overline{v}_{i+1}) = {} & R_{\text{tape}}(w) \wedge (\psi_{\text{left},i+1}(w, p, \overline{v}_{i+1}) \vee \psi_{\text{right},i+1}(w, p, \overline{v}_{i+1}) \vee \\
& \psi_{\text{stay},i+1}(w, p, \overline{v}_{i+1})),
\end{aligned}$$

$$\psi_{\text{left},i+1}(w,p,\overline{v}_{i+1}) = P(w,d_i,z_{\text{left}}) \wedge \bigvee_{1\le j\le i+1} (\psi_{\text{position},i}(w,z_j,\overline{v}_i) \wedge (p = z_{j-1})),$$

$$\psi_{\text{right},i+1}(w,p,\overline{v}_{i+1}) = P(w,d_i,z_{\text{right}}) \wedge \bigvee_{1\le j\le i+1} (\psi_{\text{position},i}(w,z_j,\overline{v}_i) \wedge (p = z_{j+1})),$$

and

$$\psi_{\text{stay},i+1}(w,p,\overline{v}_{i+1}) = P(w,d_i,z_{\text{stay}}) \wedge \bigvee_{1\le j\le i+1} (\psi_{\text{position},i}(w,z_j,\overline{v}_i) \wedge (p = z_j)).$$

Note that for each $1 \le i \le k$, the size of the formulas $\psi_{\text{symbol},i}(w,p,a,\overline{v}_i)$ and $\psi_{\text{position},i}(w,p,\overline{v}_i)$ only depends on k. We can now define $\psi_{\exists\text{-tapes}}$:

$$\psi_{\exists\text{-tapes}} = \forall w.\forall p.\forall a. \bigwedge_{1\le i\le k} ((\psi_{\text{position},i}(w,p,\overline{v}_i) \wedge \psi_{\text{symbol},i}(w,p,a,\overline{v}_i)) \to P(w,t_i,a)).$$

Intuitively, the formulas $\psi_{\exists\text{-states}}$ and $\psi_{\exists\text{-tapes}}$ together represent whether the transitions specified by $s_i, t_i, s'_i, t'_i, d_i$, for $1 \le i \le k$, together constitute a valid (partial) computation path.

Next, we define the formula $\psi_{\forall\text{-tapes}}$:

$$\psi_{\forall\text{-tapes}} = \psi_{\forall\text{-tapes-1}} \wedge \psi_{\forall\text{-tapes-2}} \wedge \psi_{\forall\text{-tapes-3}} \wedge \psi_{\forall\text{-tapes-4}} \wedge \psi_{\forall\text{-tapes-5}},$$

$$\psi_{\forall\text{-tapes-1}} = \bigwedge_{\substack{k<i\le k+t\\ 1\le j\le m}} \left(R_{\text{cell}}(p_{i,j}) \wedge \bigwedge_{1\le \ell\le k+t} R_{\text{symbol}}(q_{i,j,\ell}) \right),$$

$$\psi_{\forall\text{-tapes-2}} = \bigwedge_{1\le j\le m} \left(\begin{array}{r} \bigwedge_{\substack{1\le \ell\le k+1\\ a\in\Sigma}} ((q_{k+1,j,\ell} = z_a) \leftrightarrow \psi_{\text{symbol},k+1}(z_j,z_{k+1},z_a,\overline{v}_{k+1})) \wedge \\ \bigwedge_{k+2\le \ell\le k+t} (q_{k+1,j,\ell} = z_\square) \end{array} \right),$$

$$\psi_{\forall\text{-tapes-3}} = \bigwedge_{\substack{1\le j\le m\\ 1\le i\le k+1}} ((p_{k+1,j} = z_i) \leftrightarrow \psi_{\text{position},k+1}(z_j,z_i,\overline{v}_{k+1})),$$

$$\psi_{\forall\text{-tapes-4}} = \bigwedge_{\substack{1\le j\le m\\ k<i<k+t\\ 1\le \ell\le k+t}} \left(\begin{array}{l} (P(z_j,d_i,z_{\text{left}}) \wedge (p_{i,j} = z_\ell)) \to (p_{i+1,j} = z_{\ell-1}) \wedge \\ (P(z_j,d_i,z_{\text{right}}) \wedge (p_{i,j} = z_\ell)) \to (p_{i+1,j} = z_{\ell+1}) \wedge \\ (P(z_j,d_i,z_{\text{stay}}) \wedge (p_{i,j} = z_\ell)) \to (p_{i+1,j} = z_\ell) \wedge \end{array} \right), \text{ and}$$

$$\psi_{\forall\text{-tapes-5}} = \bigwedge_{\substack{1\le j\le m\\ k<i<k+t\\ 1\le \ell\le k+t\\ a\in\Sigma}} \left(\begin{array}{l} ((p_{i,j} \ne z_\ell) \to (q_{i+1,j,\ell} = q_{i,j,\ell})) \wedge \\ ((p_{i,j} = z_\ell) \wedge P(z_j,t_i,z_a) \to (q_{i,j,\ell} = z_a)) \wedge \\ ((p_{i,j} = z_\ell) \wedge P(z_j,t'_i,z_a) \to (q_{i+1,j,\ell} = z_a)) \end{array} \right).$$

Intuitively, the formulas $\psi_{\forall\text{-states}}$ and $\psi_{\forall\text{-tapes}}$ together represent whether the transitions specified by $s_i, t_i, s'_i, t'_i, d_i$, for $k+1 \le i \le k+t-1$, together constitute a valid (partial) computation path, extending the computation path represented by the transitions $s_i, t_i, s'_i, t'_i, d_i$, for $1 \le i \le k$.

It is straightforward to verify that φ is (logically equivalent to a formula) of the right form, containing $k' = 5k$ existentially quantified variables. (Not all quantifiers in φ occur as outermost operators in the formula, but one can easily move them outwards.) Also, it is now straightforward to verify that $(\mathbb{M}, k, t) \in \Sigma_2^{\mathrm{p}}[k*]$-TM-HALT* if and only if $(\mathcal{A}, \varphi) \in \Sigma_2^{\mathrm{p}}[k*]$-MC. □

We now turn our attention to the third statement that we used in the proof of Theorems 14 and 15.

Proposition 21. *For any parameterized problem P that is decided by some $\Sigma_2^{\mathrm{p}}[k*]$-machine with m tapes, it holds that $P \leq_{\mathrm{fpt}} \Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^{m+1}.*

Proof. Let P be a parameterized problem, and let $\mathbb{M} = (S_\exists, S_\forall, \Sigma, \Delta, s_0, F)$ be an $\Sigma_2^{\mathrm{p}}[k*]$-machine with m tapes that decides it, i.e., there exists some computable function f and some polynomial p such that for any instance (x, k) of P we have that any computation path of $\mathbb{M}$ with input (x, k) has length at most $f(k) \cdot p(|x|)$ and contains at most $f(k) \cdot \log |x|$ nondeterministic existential configurations. Moreover, let $S = S_\exists \cup S_\forall$. We show how to construct in fpt-time for each instance (x, k) of P an $\exists\forall$-machine $\mathbb{M}^{(x,k)}$ with $m + 1$ tapes, and positive integers $k', t \in \mathbb{N}$ such that $\mathbb{M}^{(x,k)}$ accepts the empty string with existential cost k' and universal cost t if and only if $\mathbb{M}$ accepts (x, k).

The idea of this construction is the following. We add to Σ a fresh symbol $\sigma_{(C_1,\ldots,C_u)}$ for each $u \leq \lceil \log |x| \rceil$ and each sequence of possible "transitions" $T_1, \ldots, T_u$ of $\mathbb{M}$. The machine $\mathbb{M}^{(x,k)}$ starts with nondeterministically writing down $f(k)$ symbols $\sigma_{(T_1,\ldots,T_{\lceil \log |x| \rceil})}$ to tape $m + 1$ (stage 1). We will choose k' in such a way (see below) so that this can be done using k' nondeterministic existential steps. Then, using universal steps, it writes down the input (x, k) to its first tape (stage 2). It continues with simulating the existential steps in the execution of $\mathbb{M}$ with input (x, k) (stage 3): each deterministic existential step can simply be performed by a deterministic universal step, and each nondeterministic existential step can be simulated by "reading off" the next configuration from the symbols on tape $m+1$, and transitioning into this configuration (if this step is allowed by Δ). Finally, the machine $\mathbb{M}^{(x,k)}$ simply performs the universal steps in the execution of $\mathbb{M}$ with input (x, k) (stage 4).

Let (x, k) be an arbitrary instance of P. We construct $\mathbb{M}^{(x,k)} = (S'_\exists, S'_\forall, \Sigma', \Delta', s'_0, F')$. We split the construction of $\mathbb{M}^{(x,k)}$ into several steps that correspond to the various stages in the execution of $\mathbb{M}^{(x,k)}$ described above. We begin with defining Σ':

$$\Sigma' = \Sigma \cup \{\, \sigma_{(T_1,\ldots,T_u)} : 0 \leq u \leq \lceil \log |x| \rceil,\ 1 \leq i \leq u, \\ T_i \in S \times \Sigma^m \times \{\mathbf{L}, \mathbf{R}, \mathbf{S}\}^m \,\}.$$

Observe that for each $s \in S$ and each $\overline{a} \in \Sigma^m$, each $T_u = (s', \overline{a}', \overline{d})$ corresponds to a tuple $(s, \overline{a}, s', \overline{a}', \overline{d})$ that may or may not be contained in Δ, i.e., a "possible transition." Note that also $\sigma_() \in \Sigma'$, where () denotes the empty sequence. Moreover, it is straightforward to verify that $|\Sigma'| = |\Sigma| + O(|x| \cdot |S|)$, since m is a constant.

We now construct the formal machinery that executes the first stage of the execution of $\mathbb{M}^{(x,k)}$. We let:

$$\begin{aligned}
S_{1,\exists} &= \{s_{1,\mathrm{guess}}, s_{1,\mathrm{done}}\}, \text{ and} \\
\Delta'_1 &= \{\, (s_{1,\mathrm{guess}}, \overline{a}, s_{1,\mathrm{guess}}, \overline{a}', \overline{d}) : \overline{a} = \square^{m+1}, \overline{a}' = \square^m \sigma_{(T_1,\ldots,T_{\lceil \log |x| \rceil})}, \\
&\qquad 1 \leq i \leq \lceil \log |x| \rceil, T_i \in S \times \Sigma^m \times \{\mathbf{L}, \mathbf{R}, \mathbf{S}\}^m, \overline{d} = \mathbf{S}^m\mathbf{R} \,\} \cup \\
&\quad \{\, (s_{1,\mathrm{guess}}, \overline{a}, s_{1,\mathrm{done}}, \overline{a}, \overline{d}) : \overline{a} = \square^{m+1}, \overline{d} = \mathbf{S}^m\mathbf{L} \,\} \cup \\
&\quad \{\, (s_{1,\mathrm{done}}, \overline{a}, s_{1,\mathrm{done}}, \overline{a}, \overline{d}) : \overline{a} \in \{\square\}^m \times \Sigma', \overline{d} = \mathbf{S}^m\mathbf{L} \,\} \cup \\
&\quad \{\, (s_{1,\mathrm{done}}, \overline{a}, s_{2,0}, \overline{a}, \overline{d}) : \overline{a} = \square^m\$, \overline{d} = \mathbf{S}^m\mathbf{R} \,\},
\end{aligned}$$

where we will define $s_{2,0} \in S'_\forall$ below ($s_{2,0}$ will be the first state of the second stage of $\mathbb{M}^{(x,k)}$). Furthermore, we let:

$$s'_0 = s_{1,\mathrm{guess}}.$$

The intuition behind the above construction is that state $s_{1,\mathrm{guess}}$ can be used as many times as necessary to write a symbol $\sigma_{(T_1,\ldots,T_{\lceil \log |x| \rceil})}$ to the $(m + 1)$-th tape, for some sequence $T_1, \ldots, T_{\lceil \log |x| \rceil}$ of "guessed

transitions." Then, the state $s_{1,\text{done}}$ moves the tape head of tape $m+1$ back to the first position, in order to continue with the second stage of the execution of $\mathbb{M}^{(x,k)}$.

We continue with the definition of those parts of $\mathbb{M}^{(x,k)}$ that perform the second stage of the execution of $\mathbb{M}^{(x,k)}$, i.e., writing down the input (x,k) to the first tape. Let the sequence $(\sigma_1,\ldots,\sigma_n) \in \Sigma^n$ denote the representation of (x,k) using the alphabet Σ. We define:

$$\begin{aligned}
S_{2,\forall} &= \{\, s_{2,i} : 1 \le i \le n \,\} \cup \{s_{2,n+1} = s_{2,\text{done}}\}, \text{ and} \\
\Delta'_2 &= \{\, (s_{2,i}, \overline{a}, s_{2,i+1}, \overline{a}', \overline{d}) : 1 \le i \le n, \overline{a} \in \square^m\sigma, \sigma \in \Sigma', \overline{a} = \sigma_i\square^{m-1}\sigma, \overline{d} = \mathbf{RS}^m \,\} \cup \\
&\quad \{\, (s_{2,\text{done}}, \overline{a}, s_{2,\text{done}}, \overline{a}, \overline{d}) : \overline{a} \in \Sigma \times \{\square\}^{m-1} \times \Sigma', \overline{d} = \mathbf{LS}^m \,\} \cup \\
&\quad \{\, (s_{2,\text{done}}, \overline{a}, s_{3,0}, \overline{a}, \overline{d}) : \overline{a} = \{\$\} \times \{\square\}^{m-1} \times \Sigma', \overline{d} = \mathbf{RS}^m \,\},
\end{aligned}$$

where we will define $s_{3,0} \in S'_\forall$ below ($s_{3,0}$ will be the first state of the third stage of $\mathbb{M}^{(x,k)}$). Intuitively, each state $s_{2,i}$ writes the i-th symbol of the representation of (x,k) (that is, symbol σ_i) to the first tape, and state $s_{2,n+1} = s_{2,\text{done}}$ moves the tape head of the first tape back to the first position. Note that the states in $S_{2,\forall}$ are deterministic.

Next, we continue with the definition of those parts of $\mathbb{M}^{(x,k)}$ that perform the third stage of the execution of $\mathbb{M}^{(x,k)}$, i.e., simulating the existential steps in the execution of $\mathbb{M}$ with input (x,k). We define:

$$\begin{aligned}
S_{3,\forall} &= S_\exists, \text{ and} \\
\Delta'_3 &= \bigcup\{\, \Delta'_{3,s} : s \in S_\exists \,\},
\end{aligned}$$

where for each $s \in S_\exists$ we define the set $\Delta'_{3,s}$ as follows:

$$\Delta'_{3,s} = \bigcup\{\, \Delta'_{3,s,\overline{a}} : \overline{a} \in \Sigma^m \,\},$$

and where for each $s \in S_\exists$ and each $\overline{a} \in \Sigma^m$ we define:

$$\Delta_{(s,\overline{a})} = \{\, (s', \overline{a}', \overline{d}) : (s, \overline{a}, s', \overline{a}', \overline{d}) \in \Delta \,\},$$

$$\Delta'_{3,s,\overline{a}} = \begin{cases}
\{\, (s, \overline{a}\sigma', s', \overline{a}'\sigma', \overline{d}\mathbf{S}) : \sigma' \in \Sigma' \,\} & \text{if } \Delta_{(s,\overline{a})} = \{(s', \overline{a}', \overline{d})\}, \\
\{\, (s, \overline{a}\sigma_{(T_1,\ldots,T_u)}, s', \overline{a}'\sigma_{(T_2,\ldots,T_u)}, \overline{d}\mathbf{S}) : \\
\quad 1 \le u \le \lceil \log |x| \rceil, \\
\quad T_1 = (s', \overline{a}', \overline{d}), (s, \overline{a}, s', \overline{a}', \overline{d}) \in \Delta \,\} \cup & \text{if } |\Delta_{(s,\overline{a})}| > 1, \\
\{(s, \overline{a}\sigma_{()}, s, \overline{a}\square, \mathbf{S}^m\mathbf{R})\} & \\
\emptyset & \text{otherwise.}
\end{cases}$$

Observe that there exist transitions from states in $S_{3,\forall}$ to states in $S_\forall$; this will be unproblematic, since we will have that $S_\forall \subseteq S'_\forall$ (see below). Intuitively, each state in $S_\exists$ that is deterministic in $\mathbb{M}$ simply performs its behavior from $\mathbb{M}$ on the first m tapes, and ignores tape $m+1$. Each state in $S_\exists$ that would lead to nondeterministic behavior in $\mathbb{M}$, performs the transition T_1 that is written as first "possible transition" in the currently read symbol $\sigma_{(T_1,\ldots,T_u)}$ on tape $m+1$ (if this transition is allowed by Δ), and removes T_1 from tape $m+1$ (by replacing $\sigma_{(T_1,\ldots,T_u)}$ by $\sigma_{(T_2,\ldots,T_u)}$). Note that the states in $S_{3,\forall}$ are deterministic.

We continue with formally defining the part of $\mathbb{M}^{(x,k)}$ that performs stage 4, i.e., performing the (possibly nondeterministic) universal steps in the execution of $\mathbb{M}$ with input (x,k). We define:

$$\begin{aligned}
S_{4,\forall} &= S_\forall, \text{ and} \\
\Delta'_4 &= \{\, (s, \overline{a}\delta', s', \overline{a}'\delta', \overline{d}\mathbf{S}) : s \in S_\forall, \overline{a} \in \Sigma^m, (s, \overline{a}, s', \overline{a}', \overline{d}) \in \Delta \,\}.
\end{aligned}$$

Intuitively, each state in $S_\forall$ simply performs its behavior from $\mathbb{M}$ on the first m tapes, and ignores tape $m+1$. Note that the states in $S_{4,\forall}$ may be nondeterministic.

We conclude our definition of $\mathbb{M}^{(x,k)} = (S'_\exists, S'_\forall, \Sigma', \Delta', s'_0, F')$:

$$\begin{aligned}
S'_\exists &= S_{1,\exists}, \\
S'_\forall &= S_{2,\forall} \cup S_{3,\forall} \cup S_{4,\forall}, \\
\Delta' &= \Delta'_1 \cup \Delta'_2 \cup \Delta'_3 \cup \Delta'_4, \\
s'_0 &= s_{1,\text{guess}} \text{ (as mentioned above), and} \\
F' &= F.
\end{aligned}$$

Finally, we define k' and t:

$$k' = 2f(k) + 2 \qquad \text{and} \qquad t = 2|(x,k)| + f(k) \cdot (p(|x|) + 1) + 2.$$

Intuitively, $\mathbb{M}'$ needs $k' = 2f(k) + 2$ existential steps to write down $f(k)$ symbols $\sigma_{(T_1,\ldots,T_{\lceil \log |x| \rceil})}$ and return the tape head of tape $m+1$ to the first position. It needs $2|(x,k)| + 2$ steps to write the input (x,k) to the first tape and return the tape head of tape 1 to the first position. It needs at most $f(k) \cdot p(|x|) + f(k)$ steps to simulate the existential steps in the execution of $\mathbb{M}$ with input (x,k), and to perform the universal steps in the execution of $\mathbb{M}$ with input (x,k).

This concludes our construction of the instance $(\mathbb{M}^{(x,k)}, k', t)$ of $\Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^{m+1}. It is straightforward to verify that $(x,k) \in P$ if and only if $(\mathbb{M}^{(x,k)}, k', t) \in \Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^{m+1}, by showing that $\mathbb{M}$ accepts (x,k) if and only if $(\mathbb{M}^{(x,k)}, k', t) \in \Sigma_2^{\mathrm{p}}[k*]$-TM-HALT^{m+1}. □

Finally, we prove the fourth statement that we used in the proof of Theorems 14 and 15.

Proposition 22. *There is an $\Sigma_2^{\mathrm{p}}[k*]$-machine with a single tape that decides $\Sigma_2^{\mathrm{p}}[k*]$-WSAT$^{\leq k}$.*

Proof. We describe an $\Sigma_2^{\mathrm{p}}[k*]$-machine $\mathbb{M}$ with 1 tape for $\Sigma_2^{\mathrm{p}}[k*]$-WSAT$^{\leq k}$. We will not spell out the machine $\mathbb{M} = (S_\exists, S_\forall, \Sigma, \Delta, s_0, F)$ in full detail, but describe $\mathbb{M}$ in such detail that the working of $\mathbb{M}$ is clear and writing down the complete formal description of $\mathbb{M}$ can be done straightforwardly.

We assume that instances (φ, k) are encoded as strings $\sigma_1\sigma_2\ldots\sigma_n$ over an alphabet $\Sigma' \subseteq \Sigma$. We denote the representation of an instance (φ, k) using the alphabet Σ' by $\mathrm{Repr}(\varphi, k)$. Also, for any Boolean formula $\psi(Z)$ over variables Z and any (partial) assignment $\gamma : Z \to \{0,1\}$, we let $\mathrm{Repr}(\psi, \gamma)$ denote the representation (using alphabet Σ) of the formula ψ, where each variable z in the domain of γ is replaced by the constant value $\gamma(z)$.

Let (φ, k) be an instance of $\Sigma_2^{\mathrm{p}}[k*]$-WSAT$^{\leq k}$, where $\varphi = \exists X.\forall Y.\psi$, $X = \{x_1, \ldots, x_n\}$, and $Y = \{y_1, \ldots, y_m\}$. In the initial configuration of $\mathbb{M}$, the tape contains the word $\mathrm{Repr}(\varphi, k)$. We construct $\mathbb{M}$ in such a way that it proceeds in seven stages. Intuitively, in stage 1, $\mathbb{M}$ adds $\square\mathrm{Repr}(\psi, \emptyset)$ to the right of the tape contents—here $\emptyset$ denotes the empty assignment and ψ is the quantifier-free part of φ. We will refer to this word $\mathrm{Repr}(\psi, \emptyset)$ as the representation of ψ. In stage 2, it appends the word $(\square 1 \ldots 1)$, containing $\lceil \log n \rceil = u$ times the symbol 1, k times to the right of the tape contents. Next, in stage 3, $\mathbb{M}$ (nondeterministically) overwrites each such word $(\square 1 \ldots 1)$ by $(\square b_1, \ldots, b_u)$, for some bits $b_1, \ldots, b_u \in \{0,1\}$. Then, in stage 4, it repeatedly reads some word $(\square b_1, \ldots, b_u)$ written at the rightmost part of the tape, and in the representation of ψ, written as "second word" on the tape, instantiates variable x_i to the value 1, where $b_1 \ldots b_u$ is the binary representation of i. After stage 4, at most k variables x_i are instantiated to 1. Then, in stage 5, $\mathbb{M}$ instantiates the remaining variables x_i in the representation of ψ to the value 0. These first five stages are all implemented using states in $S_\exists$. The remaining two stages are implemented using states in $S_\forall$. In stage 6, $\mathbb{M}$ nondeterministically instantiates each variable y_j in the representation of ψ to some truth value 0 or 1. Finally, in stage 7, the machine verifies whether the fully instantiated formula ψ evaluates to true or not, and accepts if and only if the formula ψ evaluates to true.

We now give a more detailed description of the seven stages of $\mathbb{M}$, by describing what each stage does to the tape contents, and by giving bounds on the number of steps that each stage needs. In the initial configuration, the tape contents w_0 are as follows (we omit trailing blank symbols):

$$w_0 = \$\mathrm{Repr}(\varphi, k).$$

In stage 1, $\mathbb{M}$ transforms the tape contents w_0 to the following contents w_1:

$$w_1 = \$\mathrm{Repr}(\varphi, k)\square\mathrm{Repr}(\psi, \emptyset),$$

where $\emptyset$ denotes the empty assignment to the variables $X \cup Y$. This addition to the tape contents can be done by means of $O(|\mathrm{Repr}(\varphi, k)|)$ deterministic existential steps.

Next, in stage 2, $\mathbb{M}$ adds to the tape contents k words of the form $(\square 1 \dots 1)$, each containing $\lceil \log n \rceil$ times the symbol 1, resulting in the tape contents w_2 after stage 2:

$$w_2 = \$\mathrm{Repr}(\varphi, k)\square\mathrm{Repr}(\psi, \emptyset)\underbrace{\square\overbrace{1\dots1}^{\lceil \log n \rceil}}_{\text{word } 1}\underbrace{\square\overbrace{1\dots1}^{\lceil \log n \rceil}}_{\text{word } 2}\square\dots\underbrace{\square\overbrace{1\dots1}^{\lceil \log n \rceil}}_{\text{word } k}.$$

This addition to the tape contents can be done by means of $O(k \cdot |\mathrm{Repr}(\varphi, k)|^2)$ deterministic existential steps.

Then, in stage 3, $\mathbb{M}$ proceeds nondeterministically. It replaces each word of the form $(\square 1 \dots 1)$ that were written to the tape in stage 2 by a word of the form $(\square b_1, \dots, b_u)$, for some bits $b_1, \dots, b_u \in \{0, 1\} \subseteq \Sigma$. Here we let $u = \lceil \log n \rceil$. Resultingly, the tape contents w_3 after stage 3 are:

$$w_3 = \$\mathrm{Repr}(\varphi, k)\square\mathrm{Repr}(\psi, \emptyset)\square b_1^1 \dots b_u^1 \square b_1^2 \dots b_u^2 \square \dots \square b_1^k \dots b_u^k,$$

where for each $1 \leq i \leq k$ and each $1 \leq j \leq u$, $b_j^i \in \{0, 1\}$. This transformation of the tape contents can be done by means of $O(k \cdot \lceil \log n \rceil)$ nondeterministic existential steps.

In stage 4, $\mathbb{M}$ repeatedly performs the following transformation of the tape contents, until all words $\square b_1^i \dots b_u^i$ are removed. The tape contents w_3' before each such transformation are as follows:

$$w_3' = \$\mathrm{Repr}(\varphi, k)\square\mathrm{Repr}(\psi, \alpha)\square b_1^1 \dots b_u^1 \square b_1^2 \dots b_u^2 \square \dots \square b_1^\ell \dots b_u^\ell,$$

for some partial assignment $\alpha : X \to \{0, 1\}$, and some $1 \leq \ell \leq k$. Each such transformation functions in such a way that the tape contents w_3'' afterwards are:

$$w_3'' = \$\mathrm{Repr}(\varphi, k)\square\mathrm{Repr}(\psi, \alpha')\square b_1^1 \dots b_u^1 \square b_1^2 \dots b_u^2 \square \dots \square b_1^{\ell-1} \dots b_u^{\ell-1},$$

where the bit string $b_1^\ell \dots b_u^\ell$ is the binary representation of the integer $i \leq 2^u$, and where the assignment α' is defined for all $1 \leq j \leq n$, by:

$$\alpha'(x_j) = \begin{cases} \alpha(x_j) & \text{if } \alpha(x_j) \text{ is defined,} \\ 1 & \text{if } \alpha(x_j) \text{ is undefined and } j = i, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Each such transformation can be implemented by means of $O(|\mathrm{Repr}(\psi, \alpha)|^2 \cdot k\lceil \log n \rceil)$ deterministic existential steps. After all the k transformation of stage 4 are performed, the tape contents w_4 are thus as follows:

$$w_4 = \$\mathrm{Repr}(\varphi, k)\square\mathrm{Repr}(\psi, \alpha_{\mathrm{pos}}),$$

where $\alpha_{\mathrm{pos}} : X \to \{0, 1\}$ is the partial assignment such that $\mathrm{Dom}(\alpha_{\mathrm{pos}}) = \{i_1, \dots, i_k\}$, and $\alpha_{\mathrm{pos}}(x_{i_j}) = 1$ for each $1 \leq j \leq k$, where for each $1 \leq j \leq k$, the integer i_j is such that $b_1^j \dots b_u^j$ is the binary representation of i_j. The operations in stage 4 can be implemented by means of $O(|\mathrm{Repr}(\psi, \emptyset)|^2 \cdot k^2 \lceil \log n \rceil)$ deterministic existential steps.

Next, in stage 5, the machine $\mathbb{M}$ transforms the tape contents by modifying the word $\mathrm{Repr}(\psi, \alpha_{\mathrm{pos}})$, resulting in w_5:

$$w_5 = \$\mathrm{Repr}(\varphi, k)\square\mathrm{Repr}(\psi, \alpha),$$

where the complete assignment $\alpha' : X \to \{0, 1\}$ is defined as follows:

$$\alpha(x) = \begin{cases} \alpha_{\mathrm{pos}}(x) & \text{if } x \in \mathrm{Dom}(\alpha_{\mathrm{pos}}), \\ 0 & \text{otherwise.} \end{cases}$$

This can be done using $O(|\mathrm{Repr}(\psi, \alpha_{\mathrm{pos}})|)$ nondeterministic existential steps. Note that the assignment α has weight at most k.

Now, in stage 6, the machine $\mathbb{M}$ alternates to universal steps. It nondeterministically transforms the tape contents using $O(|\mathrm{Repr}(\psi, \alpha)|)$ nondeterministic universal steps, resulting in the tape contents w_6:

$$w_6 = \$\mathrm{Repr}(\varphi, k)\square\mathrm{Repr}(\psi, \alpha \cup \beta),$$

for some complete assignment $\beta : Y \to \{0, 1\}$.

Finally, in stage 7, $\mathbb{M}$ checks whether the assignment $\alpha \cup \beta$ satisfies the formula ψ. This check can be done by means of $O(|\mathrm{Repr}(\psi, \alpha \cup \beta)|)$ deterministic universal steps. The machine $\mathbb{M}$ accepts if and only if $\alpha \cup \beta$ satisfies ψ.

It is straightforward to verify that there exists a computable function f and a polynomial p such that each computation path of $\mathbb{M}$ with input (φ, k) has length at most $f(k) \cdot p(|\varphi|)$ and contains at most $f(k) \cdot \log |\varphi|$ nondeterministic existential configurations. Also, it is straightforward to verify that $\mathbb{M}$ accepts an input (φ, k) if and only if $(\varphi, k) \in \Sigma_2^{\mathrm{p}}[k*]\text{-WSAT}^{\leq k}$. This concludes our proof that the $\Sigma_2^{\mathrm{p}}[k*]$-machine $\mathbb{M}$ decides $\Sigma_2^{\mathrm{p}}[k*]\text{-WSAT}^{\leq k}$. □

This concludes our detailed treatment of the proof of Theorems 14 and 15.

5 Relation to Other Parameterized Complexity Classes

In order to be able to use hardness for the classes $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k, t]$ to show that parameterized problems are not fpt-reducible to SAT, we need to have confidence that these classes are indeed different from the classes para-NP and para-co-NP. We conjecture this to be the case. In this section, we provide several results that strengthen our confidence in this conjecture. In fact, we provide results that can be interpreted as evidence that the classes $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k, t]$ differ from the parameterized complexity classes para-NP, para-co-NP, para-Σ_2^{P} and para-Π_2^{P}.

5.1 Basic Separations for the Class $\Sigma_2^{\mathrm{p}}[k*]$

We begin with establishing several basic results about the relation of $\Sigma_2^{\mathrm{p}}[k*]$ to the parameterized complexity classes para-NP, para-co-NP, para-Σ_2^{P}, XNP, and Xco-NP. We will investigate the relation of $\Sigma_2^{\mathrm{p}}[k*]$ with para-Π_2^{P}, and more intricate results about the relation of $\Sigma_2^{\mathrm{p}}[k*]$ with para-co-NP, in Section 5.3.

First, we show that $\Sigma_2^{\mathrm{p}}[k*] \subseteq$ para-Σ_2^{P}. In polynomial time, any formula $\exists X.\forall Y.\psi$ can be transformed into a quantified Boolean formula with a $\exists\forall$ quantifier prefix that is true if and only if for some assignment α of weight k to the variables X the formula $\forall Y.\psi[\alpha]$ is true.

Proposition 23. $\Sigma_2^{\mathrm{p}}[k*] \subseteq$ para-Σ_2^{P}

Proof (sketch). We show that $\Sigma_2^{\mathrm{p}}[k*]\text{-WSAT} \in$ para-Σ_2^{P} by showing that each instance of $\Sigma_2^{\mathrm{p}}[k*]\text{-WSAT}$ can be transformed in polynomial time into an equivalent instance of QSAT_2. Let (φ, k) be an arbitrary instance of $\Sigma_2^{\mathrm{p}}[k*]\text{-WSAT}$, where $\varphi = \exists X.\forall Y.\psi$. We describe how to construct an instance $\varphi' = \exists X \cup Z.\forall Y.(\psi \wedge \psi')$ of QSAT_2 that is a yes-instance if and only if $(\varphi, k) \in \Sigma_2^{\mathrm{p}}[k*]\text{-WSAT}$.

Let $X = \{x_1, \ldots, x_n\}$. We let $Z = \{\, z_{i,j}, z_i : 1 \leq i \leq n, 1 \leq j \leq i \,\} \cup \{\, z_{i,0} : 1 \leq i \leq n \,\} \cup \{z_{n,n+1}\}$. Intuitively, the variables $z_{i,j}$ encodes whether among the variables $x_1, \ldots, x_i$ at least j variables are set to true, and the variables z_i encode whether among the variables $x_1, \ldots, x_n$ exactly i variables are set to true.

We let ψ' be a conjunction of several formulas. Firstly, we add the conjuncts $(z_{1,1} \leftrightarrow x_1)$ and $(z_{i,0} \leftrightarrow (x_1 \vee \neg x_1))$ for each $1 \leq i \leq n$. Moreover, for each $1 < i \leq n$, we add:

$$\left(x_i \leftrightarrow \bigwedge_{1 \leq j \leq i} (z_{i,j} \leftrightarrow z_{i-1,j-1})\right) \wedge \left(\neg x_i \leftrightarrow \bigwedge_{1 \leq j \leq i} (z_{i,j} \leftrightarrow z_{i-1,j})\right).$$

For each $1 \leq i \leq n$, we add $(z_i \leftrightarrow (z_{n,i} \wedge \neg z_{n,i+1}))$. We also add the conjunct $(z_{n,n+1} \leftrightarrow (x_1 \wedge \neg x_1))$. Then, to ensure that exactly k of the variables in X are set to true, we add z_k as conjunct.

It is straightforward to verify that $(\varphi, k) \in \Sigma_2^{\mathrm{p}}[k*]$-WSAT if and only if $\varphi' \in \mathrm{QSAT}_2$. □

Also, trivially, para-co-NP $\subseteq \Sigma_2^{\mathrm{p}}[k*]$, because the para-co-NP-complete parameterized problem UNSAT (where the parameter value is some fixed constant) can be seen as a special case of $\Sigma_2^{\mathrm{p}}[k*]$-WSAT. To summarize, we observe the following inclusions:

$$\text{para-co-NP} \subseteq \Sigma_2^{\mathrm{p}}[k*] \subseteq \text{para-}\Sigma_2^{\mathrm{P}} \qquad \text{and} \qquad \text{para-NP} \subseteq \Pi_2^{\mathrm{p}}[k*] \subseteq \text{para-}\Pi_2^{\mathrm{P}}.$$

This immediately leads to the following result.

Proposition 24. *Assuming that* NP $\neq$ co-NP*, it holds that* $\Sigma_2^{\mathrm{p}}[k*] \not\subseteq$ para-NP*.*

It is also not difficult to see that $\Sigma_2^{\mathrm{p}}[k*] \subseteq$ Xco-NP. This is witnessed by the straightforward brute-force algorithm to solve $\Sigma_2^{\mathrm{p}}[k*]$-WSAT that tries out all $\binom{n}{k} = O(n^k)$ assignments of weight k to the existentially quantified variables (and that uses nondeterminism to handle the assignment to the universally quantified variables).

Proposition 25. $\Sigma_2^{\mathrm{p}}[k*] \subseteq$ Xco-NP*.*

A natural question to ask is whether para-NP $\subseteq \Sigma_2^{\mathrm{p}}[k*]$. Since para-NP $\subseteq$ Xco-NP implies NP = co-NP [18, Proposition 8], this is unlikely.

Proposition 26. *Assuming that* NP $\neq$ co-NP*, it holds that* para-NP $\not\subseteq \Sigma_2^{\mathrm{p}}[k*]$*.*

This implies that $\Sigma_2^{\mathrm{p}}[k*]$ is likely to be a strict subset of para-Σ_2^{P}.

Corollary 27. *Assuming that* NP $\neq$ co-NP*, it holds that* $\Sigma_2^{\mathrm{p}}[k*] \subsetneq$ para-Σ_2^{P}*.*

Finally, we observe that it is unlikely that $\Sigma_2^{\mathrm{p}}[k*] \subseteq$ XNP. This is a direct consequence of the fact that para-co-NP $\subseteq$ XNP implies that NP = co-NP.

Proposition 28. *Assuming that* NP $\neq$ co-NP*, it holds that* $\Sigma_2^{\mathrm{p}}[k*] \not\subseteq$ XNP*.*

5.2 Basic Separations for the Classes $\Sigma_2^{\mathrm{p}}[*k, t]$

We continue with with establishing several basic results about the relation of the classes $\Sigma_2^{\mathrm{p}}[*k, t]$ to the parameterized complexity classes para-NP, para-co-NP, para-Π_2^{P}, XNP, and Xco-NP. We will investigate the relation of the classes $\Sigma_2^{\mathrm{p}}[*k, t]$ with para-Σ_2^{P}, and more intricate results about the relation of the classes $\Sigma_2^{\mathrm{p}}[*k, t]$ with para-NP, in Section 5.3.

Similarly to the case of k-$*$, we can observe the following inclusions:

$$\text{para-NP} \subseteq \Sigma_2^{\mathrm{p}}[*k, 1] \subseteq \cdots \subseteq \Sigma_2^{\mathrm{p}}[*k, \mathrm{P}] \subseteq \text{para-}\Sigma_2^{\mathrm{P}}$$

and

$$\text{para-co-NP} \subseteq \Pi_2^{\mathrm{p}}[*k, 1] \subseteq \cdots \subseteq \Pi_2^{\mathrm{p}}[*k, \mathrm{P}] \subseteq \text{para-}\Pi_2^{\mathrm{P}}.$$

This immediately leads to the following result.

Proposition 29. *Assuming that* NP $\neq$ co-NP*, it holds that* $\Sigma_2^{\mathrm{p}}[*k, 1] \not\subseteq$ para-co-NP*.*

It is also not so difficult to see that $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}] \subseteq$ XNP. This is witnessed by the straightforward brute-force algorithm to solve $\Sigma_2^{\mathrm{p}}[*k]$-WSAT that tries out all $\binom{n}{k} = O(n^k)$ assignments of weight k to the universally quantified variables (and that uses nondeterminism to handle the assignment to the existentially quantified variables).

Proposition 30. $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}] \subseteq$ XNP*.*

A natural question to ask is whether para-co-NP is contained in any of the classes $\Sigma_2^{\mathrm{p}}[*k,t]$. Since para-co-NP $\subseteq$ XNP implies NP = co-NP [18, Proposition 8], this is unlikely.

Proposition 31. *Assuming that* NP $\neq$ co-NP*, it holds that* para-co-NP $\not\subseteq \Sigma_2^{\mathrm{p}}[*k,\mathrm{P}]$.

This immediately gives us the following separation.

Corollary 32. *Assuming that* NP $\neq$ co-NP*, it holds that* $\Sigma_2^{\mathrm{p}}[*k,\mathrm{P}] \subsetneq$ para-Σ_2^{P}.

Finally, we observe that it is unlikely that $\Sigma_2^{\mathrm{p}}[*k,1] \subseteq$ Xco-NP. This is a direct consequence of the fact that para-NP $\subseteq$ Xco-NP implies that NP = co-NP.

Proposition 33. *Assuming that* NP $\neq$ co-NP*, it holds that* $\Sigma_2^{\mathrm{p}}[*k,1] \not\subseteq$ Xco-NP.

5.3 More Separation Results

In Sections 5.1 and 5.2, we provided several results relating the classes $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k,t]$ to other parameterized complexity classes known from the literature. However, several important questions were left open. Concretely, we did not establish whether $\Sigma_2^{\mathrm{p}}[k*] \subseteq$ para-co-NP or whether $\Sigma_2^{\mathrm{p}}[*k,t] \subseteq$ para-NP, for any of the classes $\Sigma_2^{\mathrm{p}}[*k,t]$. In this section, we provide results that address these questions.

In particular, we show that under various complexity-theoretic assumptions, it holds that $\Sigma_2^{\mathrm{p}}[k*] \not\subseteq$ para-co-NP and $\Sigma_2^{\mathrm{p}}[*k,t] \not\subseteq$ para-NP. The complexity-theoretic assumptions that underlie our results are stronger than the most commonly used assumptions (such as P $\neq$ NP). The assumptions that we use in this section are related to the non-existence of subexponential-time reductions from canonical problems of the second level of the PH to canonical problems of the first level—e.g., the non-existence of a subexponential-time reduction from QSAT_2(DNF) to SAT.

Due to the fact that the underlying complexity-theoretic assumptions for the results in this section are stronger than the most commonly used assumptions, we cannot claim that they are as widely believed to be true. Correspondingly, we urge the reader to be skeptical about the absolute truth of the separation results that we give in this section. It might well turn out to be the case that the strong assumptions that we use are not true. Nevertheless, we believe that the results in this section are a useful step towards a better understanding of the parameterized complexity classes $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k,t]$.

We begin in Section 5.3.1 with showing that inclusion of A[2] in para-NP or para-co-NP implies the existence of a subexponential-time reduction from QSAT_2(3DNF) to SAT or UNSAT, respectively. In fact, for the case of A[2], we can show a slightly stronger statement, namely that such subexponential-time reductions must exist even in the case where there exists an $f(k)m^{o(k^{1/3})}$ time reduction from the A[2]-complete problem A[2]-MC to SAT or UNSAT—in other words, we can rule out the existence of a larger class of reductions under the same complexity-assumption.

Since the classes $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k,t]$ contain A[2], the result that we establish in Section 5.3.1 already establishes a separation between the classes $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k,t]$ on the one hand, and the classes para-NP and para-co-NP, on the other hand. For the classes $\Sigma_2^{\mathrm{p}}[k*]$, $\Sigma_2^{\mathrm{p}}[*k,2]$ and $\Sigma_2^{\mathrm{p}}[*k,\mathrm{P}]$, we can establish separation results from para-NP and para-co-NP using weaker complexity-theoretic assumptions—namely, that there exists no subexponential-time reduction from QSAT_2(DNF) or QSAT_2 to SAT or UNSAT. We develop these results in Section 5.3.2.

Neither of the results in Sections 5.3.1 and 5.3.2 implies the other. The result in Section 5.3.1 rules out a larger class of reductions than fpt-reductions, and the result in Section 5.3.2 is based on a weaker complexity-theoretic assumption.

Throughout the remainder of the paper, we use the "little-o" notation to denote $o^{\mathrm{eff}}(\cdot)$ as used by Flum and Grohe [19], i.e., for any function $f, g : \mathbb{N} \to \mathbb{N}$, we say that $f(n)$ is $o(g(n))$ if there is a computable function $\lambda : \mathbb{N} \to \mathbb{N}$ that is unbounded and some $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ it holds that $f(n) \leq g(n)/\lambda(n)$. Moreover, we may assume without loss of generality that λ is nondecreasing [19, Lemma 3.23].

5.3.1 Relation of A[2] with para-NP and para-co-NP

We begin by showing that if A[2] $\subseteq$ para-NP, then there exists a subexponential-time reduction from QSAT_2(3DNF) to SAT. In other words, there exists no fpt-reduction from A[2]-MC to SAT, unless there exists a subexponential-time reduction from QSAT_2(3DNF) to SAT. The following result even rules out reductions with a running time of $f(k)m^{o(k^{1/3})}$ under the same complexity-theoretic assumption.

Theorem 34. *If there exists an $f(k)m^{o(k^{1/3})}$ time reduction from* A[2]-MC *to* SAT*, where k denotes the parameter value (i.e., the size of the the first-order formula), m denotes the instance size, and f is some computable function, then there exists a subexponential-time reduction from* QSAT_2(3DNF) *to* SAT*, i.e., a reduction that runs in time $2^{o(n)}$, where n denotes the number of variables.*

Proof. Suppose that there is a reduction R from A[2]-MC to SAT that runs in time $f(k)m^{o(k^{1/3})}$, that is, for sufficiently large values of k it runs in time $f(k)m^{k^{1/3}/\lambda(k)}$ for some computable function f and some computable, unbounded and nondecreasing function λ. We construct a reduction from QSAT_2(3DNF) to SAT that, for sufficiently large values of n, runs in time $2^{o(n)}$, where n is the number of variables. Let (φ, k) be an instance of QSAT_2, where $\varphi = \exists X_1.\forall X_2.\psi$ and where ψ is in 3DNF. We will construct an equivalent instance $(\mathcal{A}, \varphi')$ of A[2]-MC from the instance (φ, k) of QSAT_2(3DNF), and we will then use the reduction from A[2]-MC to SAT to transform $(\mathcal{A}, \varphi')$ to an equivalent instance of SAT in subexponential time.

We may assume without loss of generality that $|X_1| = |X_2|$; we can add dummy variables to X_1 and X_2 to ensure this. Then, let $n = |X_1| = |X_2|$. We denote the size of ψ by m. Let $X = X_1 \cup X_2$. We may assume without loss of generality that $f(\ell) \geq 2^\ell$ for all $\ell \geq 1$, and that f is nondecreasing and unbounded.

In order to construct $(\mathcal{A}, \varphi')$, we will use several auxiliary definitions. We define the function g as follows:

$$g(\ell) = f(c \cdot \ell^3);$$

we will define the value of the constant c below. Then, define the function g^{rev} as follows:

$$g^{\text{rev}}(h) = \max\{\, q : g(2q+1) \leq h \,\}.$$

Since the function f is nondecreasing and unbounded, the functions g and g^{rev} are also nondecreasing and unbounded. Moreover, we get that $g(2g^{\text{rev}}(h)+1) \leq h$. Also, since $f(\ell) \geq 2^\ell$ for all $\ell \geq 1$, we get that $g(\ell) \geq 2^\ell$ for all $\ell \geq 1$, and therefore also that $g^{\text{rev}}(h) \leq \log h$, for all $h \geq 1$.

We then choose the integers r and k as follows.

$$r = \lfloor n / g^{\text{rev}}(n) \rfloor \qquad \text{and} \qquad k = \lceil n/r \rceil.$$

Due to this choice for k and r, we get the following inequalities:

$$r \leq \frac{n}{g^{\text{rev}}(n)}, \quad k \geq g^{\text{rev}}(n), \quad r \geq \frac{n}{2g^{\text{rev}}(n)}, \quad \text{and} \quad k \leq 2g^{\text{rev}}(n) + 1.$$

We now construct an instance $(\mathcal{A}, \varphi')$ of A[2]-MC such that $\mathcal{A} \models \varphi'$ if and only if φ is a yes-instance of QSAT_2. In order to do so, for each $i \in \{1,2\}$, we split X_i into k disjoint sets $X_{i,1}, \ldots, X_{i,k}$. We do this in such a way that each set $X_{i,j}$ has at most n/k elements, i.e., $|X_{i,j}| \leq 2r$ for each $i \in \{1,2\}$ and each $1 \leq j \leq k$. To construct φ', we will introduce a first-order variable $y_{i,j}$ for each $i \in \{1,2\}$ and each $1 \leq j \leq k$. Intuitively, these variables will be used to quantify over truth assignments to the variables in the sets $X_{i,j}$. For each $i \in \{1,2\}$, we let $Y_i = \{\, y_{i,j} : 1 \leq j \leq k \,\}$. Moreover, for the sake of convenience, we introduce alternative names for these variables. Let $Z = Y_1 \cup Y_2 = \{z_1, \ldots, z_{2k}\}$. Also, for each $i \in \{1,2\}$ and each $1 \leq j \leq k$, we introduce a binary predicate symbol $S_{i,j}$. In addition, we introduce a ternary predicate symbol R. We then define the first-order formula φ' as follows:

$$\begin{aligned}
\varphi' &= \exists Y_1.\forall Y_2.(\psi'_1 \wedge (\psi'_2 \rightarrow \psi'_3)); \\
\psi'_i &= \bigwedge_{1 \leq j \leq k} S_{i,j}(y_{i,j}) && \text{for each } i \in \{1,2\}; \text{ and} \\
\psi'_3 &= \bigvee_{1 \leq j_1 < j_2 < j_3 \leq 2k} R(z_{j_1}, z_{j_2}, z_{j_3}).
\end{aligned}$$

We define the relational structure $\mathcal{A}$ as follows. The universe A of $\mathcal{A}$ is defined as follows:

$$A = \{\, \alpha : i \in \{1,2\}, 1 \leq j \leq k, \alpha \text{ is a truth assignment to } X_{i,j} \,\}.$$

Then, for each $i \in \{1,2\}$ and each $1 \leq j \leq k$, the interpretation of the relation $S_{i,j}$ is defined as follows:

$$S_{i,j}^{\mathcal{A}} = \{\, \alpha : \alpha \text{ is a truth assignment to } X_{i,j} \,\}.$$

Finally, the interpretation of the relation R is defined as follows:

$$R^{\mathcal{A}} = \{\, (\alpha_1, \alpha_2, \alpha_3) : \text{the assignment } \alpha_1 \cup \alpha_2 \cup \alpha_3 \text{ satisfies some term in } \psi \,\}.$$

Let $m' = |\mathcal{A}| + |\varphi'|$, and $k' = |\varphi'|$. Observe that by the construction of $\mathcal{A}$, we know that $|\mathcal{A}| \leq 2k2^{2r} + 8k^3 2^{6r}$. Moreover, we have that $k' = |\varphi'| = O(k^3)$. In fact, we can straightforwardly construct φ' in such a way that $k' = c \cdot k^3$, for some constant c. We let this constant c be the constant used for the definition of the function g above.

We verify that $\varphi \in \text{QSAT}_2$ if and only if $\mathcal{A} \models \varphi'$.

($\Rightarrow$) Suppose that $\varphi \in \text{QSAT}_2$. Then there is a truth assignment $\beta_1 : X_1 \to \{0,1\}$ such that for all truth assignment $\beta_2 : X_2 \to \{0,1\}$ it holds that $\psi[\beta_1 \cup \beta_2]$ is true. We show that $\mathcal{A} \models \varphi'$. We define the assignment $\gamma_1 : Y_1 \to A$ as follows. For each $1 \leq j \leq k$, we let $\gamma_1(y_{1,j}) = \alpha_{1,j}$, where $\alpha_{1,j}$ is the restriction of β_1 to the variables in $X_{1,j}$. We show that for each assignment $\gamma_2 : Y_2 \to A$ it holds that $\mathcal{A}, \gamma_1 \cup \gamma_2 \models \psi'$. Take an arbitrary assignment $\gamma_2 : Y_2 \to A$. Let $\gamma = \gamma_1 \cup \gamma_2$. Clearly, γ satisfies ψ'_1. Suppose that γ satisfies ψ'_2. We need to show that then γ also satisfies ψ'_3. For each $i \in \{1,2\}$ and each $1 \leq j \leq k$, we have that $\gamma(y_{i,j})$ is a truth assignment to the propositional variables $X_{i,j}$. Now consider the truth assignment $\beta : X_1 \cup X_2 \to \{0,1\}$ that is defined as follows. For each $x \in X_{i,j}$, we let $\beta(x) = \alpha_{i,j}(x)$, where $\alpha_{i,j} = \gamma(y_{i,j})$. By construction of γ_1 and β, we know that β agrees with β_1 on the variables in X_1. Therefore, we know that β must satisfy ψ, that is, β must satisfy some term in ψ. Since each term contains at most three literals, we know that there are some $1 \leq j_1 < j_2 < j_3 \leq 2k$ such that $R(z_{j_1}, z_{j_2}, z_{j_3})$ is satisfied by γ. Therefore, γ satisfies ψ'_3. Since γ_2 was arbitrary, we can conclude that $\mathcal{A} \models \varphi'$.

($\Leftarrow$) Conversely, suppose that $\mathcal{A} \models \varphi'$. That is, there is some assignment $\gamma_1 : Y_1 \to A$ such that for all assignments $\gamma_2 : Y_2 \to A$ it holds that $\mathcal{A}, \gamma_1 \cup \gamma_2 \models \psi'_1 \wedge (\psi'_2 \to \psi'_3)$. We show that $\varphi \in \text{QSAT}_2$. Since ψ'_1 contains only variables in Y_1, we know that γ_1 satisfies ψ'_1. Consider the truth assignment $\beta_1 : X_1 \to \{0,1\}$ that is defined as follows. For each $x \in X_{1,j}$, we let $\beta_1(x) = \alpha_{1,j}(x)$, where $\alpha_{1,j} = \gamma_1(y_{1,j})$. We know that β_1 is well defined, because γ_1 satisfies ψ'_1. We show that for all truth assignments $\beta_2 : X_2 \to \{0,1\}$ it holds that $\beta_1 \cup \beta_2$ satisfies ψ. Take an arbitrary truth assignment $\beta_2 : X_2 \to \{0,1\}$. Then, we define $\gamma_2 : Y_2 \to A$ as follows. For each $1 \leq j \leq k$, we let $\gamma_2(y_{2,j}) = \alpha_{2,j}$, where $\alpha_{2,j}$ is the restriction of β_2 to the variables in $X_{2,j}$. Let $\gamma = \gamma_1 \cup \gamma_2$. Clearly, γ satisfies ψ'_2. Therefore, we know that γ also satisfies ψ'_3. By construction of ψ'_3 and $\mathcal{A}$, there must be some $1 \leq j_1 < j_2 < j_3 \leq 2k$ such that $R(z_{j_1}, z_{j_2}, z_{j_3})$ is satisfied by γ. From this, we can conclude that $\beta_1 \cup \beta_2$ satisfies some term in ψ. Since β_2 was arbitrary, we can conclude that $\varphi \in \text{QSAT}_2$.

Since $(\mathcal{A}, \varphi')$ is an instance of A[2]-MC, we can apply the reduction R to obtain an equivalent instance φ'' of SAT. By first constructing $(\mathcal{A}, \varphi')$ from φ, and then constructing φ'' from $(\mathcal{A}, \varphi')$, we get a reduction R' from QSAT_2 to SAT. We analyze the running time of this reduction R' in terms of the values n.

Firstly, constructing $(\mathcal{A}, \varphi')$ can be done in time:

$$O(2k2^{2r} \cdot 8k^3 2^{6r} \cdot |\psi|) = 2^{o(n)}.$$

Then, applying the reduction R to obtain φ'' from $(\mathcal{A}, \varphi')$ takes time $f(k')(m')^{(k')^{1/3}/\lambda(k')}$. We analyze the different factors of this expression in terms of n. Firstly:

$$f(k') = f(ck^3) = g(k) \leq g(2g^{\text{rev}}(n) + 1) \leq n.$$

In our analysis of the running time of R, we will use an auxiliary function λ'. In particular, it will turn out that in this analysis we need the following inequality to hold:

$$\lambda'(n) \leq \frac{\lambda(ck^3) g^{\text{rev}}(n)}{6(ck^3)^{1/3}}.$$

We will ensure this by defining λ' as follows:

$$\lambda'(h) = \frac{\lambda(cg^{\mathrm{rev}}(h)^3)g^{\mathrm{rev}}(h)}{6c^{1/3}(2g^{\mathrm{rev}}(h)+1)}.$$

We will also need λ' to be unbounded. In order to see that λ' is unbounded, we observe the following inequality:

$$\lambda'(h) \geq \frac{\lambda(cg^{\mathrm{rev}}(h)^3)g^{\mathrm{rev}}(h)}{6c^{1/3}3g^{\mathrm{rev}}(h)} = \frac{\lambda(cg^{\mathrm{rev}}(h)^3)}{18c^{1/3}}.$$

Then, in order to analyze the second factor $(m')^{(k')^{1/3}/\lambda(k')}$ in terms of n, we firstly consider the following inequality (here we use the function λ'):

$$(2^{6r})^{(k')^{1/3}/\lambda(k')} \leq 2^{6n(k')^{1/3}/(\lambda(k')g^{\mathrm{rev}}(n))} = 2^{n6(ck^3)^{1/3}/(\lambda(ck^3)g^{\mathrm{rev}}(n))} \leq 2^{n/\lambda'(n)} = 2^{o(n)}.$$

We then get:

$$(m')^{(k')^{1/3}/\lambda(k')} \leq (k' + 2k2^{2r} + 8k^3 2^{6r})^{(k')^{1/3}/\lambda(k')} \leq (k')^{k'}(2k)^{k'}2^{2r(k')^{1/3}/\lambda(k')}(8k^3)^{k'}2^{6r(k')^{1/3}/\lambda(k')}.$$

Then, since $(k')^{k'} \leq O(\log^3 n)^{O(\log^3 n)} = 2^{o(n)}$, we know that the factors $(k')^{k'}$ and $(2k)^{k'}$ are $2^{o(n)}$. By a similar argument, we know that the factor $(8k^3)^{k'}$ is $2^{o(n)}$. Moreover, because we know that $(2^{6r})^{(k')^{1/3}/\lambda(k')} \leq 2^{o(n)}$ we know that the factors $2^{2r(k')^{1/3}/\lambda(k')}$ and $2^{6r(k')^{1/3}/\lambda(k')}$ are also $2^{o(n)}$. Therefore, we know that $(m')^{(k')^{1/3}/\lambda(k')}$ is $2^{o(n)}$. Concluding, the reduction R' from QSAT_2(3DNF) to SAT runs in time $2^{o(n)}$. □

Theorem 34 directly gives us the following corollary, separating A[2] from para-NP.

Corollary 35. *Assuming that there exists no subexponential-time reduction from* QSAT_2(3DNF) *to* SAT*, it holds that* A[2] $\not\subseteq$ para-NP*.*

In particular, since A[2] $\subseteq \Sigma_2^{\mathrm{p}}[*k, 1]$, this allows us to separate $\Sigma_2^{\mathrm{p}}[*k, 1]$ from para-NP.

Corollary 36. *Assuming that there exists no subexponential-time reduction from* QSAT_2(3DNF) *to* SAT*, it holds that* $\Sigma_2^{\mathrm{p}}[*k, 1] \not\subseteq$ para-NP*.*

The proof of Theorem 34 can straightforwardly be modified to separate A[2] also from para-co-NP.

Corollary 37. *Assuming that there exists no subexponential-time reduction from* QSAT_2(3DNF) *to* UNSAT*, it holds that* A[2] $\not\subseteq$ para-co-NP*.*

In fact, the result of Theorem 34 generalizes to higher levels of the A-hierarchy and higher levels of the PH.

Proposition 38. *Let* $t \geq 2$ *and* $i \geq 1$*. If* A[t] $\subseteq$ para-Σ_i^{P}*, then there exists a subexponential-time reduction from* QSAT_t(3DNF) *to* QSAT_i *for even* t*, and from* QSAT_t(3CNF) *to* QSAT_i *for odd* t*. If* A[t] $\subseteq$ para-Π_i^{P}*, then there exists a subexponential-time reduction from* QSAT_t(3DNF) *to* co-QSAT_i *for even* t*, and from* QSAT_t(3CNF) *to* co-QSAT_i *for odd* t*.*

Proof (sketch). The proof of Theorem 34 straightforwardly generalizes to arbitrary $t \geq 2$ and arbitrary $i \geq 1$, both for para-Σ_i^{P} and para-Π_i^{P}. □

5.3.2 Results for $\Sigma_2^{\mathrm{p}}[k*]$, $\Sigma_2^{\mathrm{p}}[*k, 2]$ and $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$

Next, we show that separation results that we established in Section 5.3.1 can be strengthened for the cases of $\Sigma_2^{\mathrm{p}}[k*]$, $\Sigma_2^{\mathrm{p}}[*k, 2]$ and $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$—namely, we can establish separation results using a weaker complexity-theoretic assumption. We begin with the case for $\Sigma_2^{\mathrm{p}}[*k, 2]$, and show the following slightly stronger result, ruling out fpt-reductions with a running time of $f(k)n^{o(k)}m^{O(1)}$.

Theorem 39. *If there exists an $f(k)n^{o(k)}m^{O(1)}$ time reduction from $\Sigma_2^{\mathrm{p}}[kk]$-WSAT$(\Gamma_{2,4})$ to* SAT*, where k denotes the parameter value (i.e., the integer k given as part of the input), n denotes the number of variables and m denotes the instance size, then there exists a subexponential-time reduction from* QSAT$_2$(DNF) *to* SAT*, i.e., a reduction that runs in time $2^{o(n)}m^{O(1)}$, where n denotes the number of variables and m denotes the instance size.*

Proof. Assume that there exists a reduction R from $\Sigma_2^{\mathrm{p}}[kk]$-WSAT$(\Gamma_{2,4})$ to SAT that runs in time $f(k)n^{k/\lambda(k)}m^{O(1)}$ for sufficiently large values of k, where f is some computable function and λ is some nondecreasing and unbounded computable function.

We now construct a reduction from QSAT$_2$(DNF) to SAT that runs in time $2^{o(n)}m^{O(1)}$, where n is the number of variables, and m is the instance size. Let $\varphi = \exists X_1.\forall X_2.\psi$ be an instance of QSAT$_2$(DNF), where ψ is in DNF. We will construct an equivalent instance (φ', k) of $\Sigma_2^{\mathrm{p}}[kk]$-WSAT$(\Gamma_{2,4})$ from the instance (φ, k) of QSAT$_2$(DNF), and we will then use the reduction from $\Sigma_2^{\mathrm{p}}[kk]$-WSAT$(\Gamma_{2,4})$ to SAT to transform (φ', k) to an equivalent instance of SAT in subexponential time.

We may assume without loss of generality that $|X_1| = |X_2| = n$; we can add dummy variables to X_1 and X_2 to ensure this. We denote the size of ψ by m. Let $X = X_1 \cup X_2$. We may assume without loss of generality that $f(k) \geq 2^k$ and that f is nondecreasing and unbounded.

In order to construct (φ', k), we will use several auxiliary definitions. We define the function f^{rev} as follows:

$$f^{\mathrm{rev}}(h) = \max\{\, q : f(2q+1) \leq h \,\}.$$

Since the function f is nondecreasing and unbounded, the function f^{rev} is also nondecreasing and unbounded. Also, we know that $f(2f^{\mathrm{rev}}(h)+1) \leq h$, and since $f(k) \geq 2^k$, we know that $f^{\mathrm{rev}}(h) \leq \log h$. We then choose the integers r and k as follows.

$$r = \lfloor n/f^{\mathrm{rev}}(n) \rfloor \qquad \text{and} \qquad k = \lceil n/r \rceil.$$

Due to this choice for k and r, we get the following inequalities:

$$r \leq \frac{n}{f^{\mathrm{rev}}(n)}, \quad k \geq f^{\mathrm{rev}}(n), \quad r \geq \frac{n}{2f^{\mathrm{rev}}(n)}, \quad \text{and} \quad k \leq 2f^{\mathrm{rev}}(n)+1.$$

We now construct an instance (φ', k) of $\Sigma_2^{\mathrm{p}}[kk]$-WSAT$(\Gamma_{2,4})$ that is a yes-instance if and only if φ is a yes-instance of QSAT$_2$(DNF). We will describe φ' as a quantified Boolean formula whose matrix corresponds to a circuit of depth 4 and weft 2. In order to do so, for each $i \in \{1,2\}$, we split X_i into k disjoint sets $X_{i,1}, \ldots, X_{i,k}$. We do this in such a way that each set $X_{i,j}$ has at most n/k elements, i.e., $|X_{i,j}| \leq 2r$ for all $i \in \{1,2\}$ and all $1 \leq j \leq k$. Now, for each truth assignment $\alpha : X_{i,j} \to \{0,1\}$ we introduce a new variable $y_{i,j}^{\alpha}$. Formally, we define a set of variables $Y_{i,j}$ for each $i \in \{1,2\}$ and each $1 \leq j \leq k$:

$$Y_{i,j} = \{\, y_{i,j}^{\alpha} : \alpha \text{ is a truth assignment to } X_{i,j} \,\}.$$

We have that $|Y_{i,j}| \leq 2^{2r}$, for each $i \in \{1,2\}$ and each $1 \leq j \leq k$. We let $Y_i = \bigcup_{1 \leq j \leq k} Y_{i,j}$, and we let $Y = Y_1 \cup Y_2$.

We continue the construction of the formula φ'. For each $i \in \{1,2\}$, we define the formula ψ_{Y_i} as follows:

$$\psi_{Y_i} = \bigwedge_{1 \leq j \leq k} \bigwedge_{\substack{\alpha,\alpha' : X_{i,j} \to \{0,1\} \\ \alpha \neq \alpha'}} \left(\neg y_{i,j}^{\alpha} \vee \neg y_{i,j}^{\alpha'} \right).$$

Then we define the auxiliary functions $\sigma_0, \sigma_1 : X \to 2^Y$, that map variables in X to sets of variables in Y. For each $x \in X_{i,j}$, we let:

$$\sigma_0(x) = \{\, y_{i,j}^{\alpha} : \alpha \text{ is a truth assignment to } X_{i,j}, \alpha(x) = 0 \,\}, \text{ and}$$
$$\sigma_1(x) = \{\, y_{i,j}^{\alpha} : \alpha \text{ is a truth assignment to } X_{i,j}, \alpha(x) = 1 \,\}.$$

Intuitively, for $b \in \{0,1\}$, $\sigma_b(x)$ corresponds to those variables $y^{\alpha}_{i,j}$ where α is an assignment that sets x to b.

Now, we construct a formula ψ'', by transforming the formula ψ in the following way. We replace each occurrence of a positive literal $x \in X_{i,j}$ in ψ by the formula χ_x, that is defined as follows:

$$\chi_x = \bigwedge_{y^{\alpha}_{i,j} \in \sigma_0(x)} \neg y^{\alpha}_{i,j}.$$

Moreover, we replace each occurrence of a negative literal $\neg x$ in ψ (for $x \in X_{i,j}$) by the formula $\chi_{\neg x}$, that is defined as follows:

$$\chi_{\neg x} = \bigwedge_{y^{\alpha}_{i,j} \in \sigma_1(x)} \neg y^{\alpha}_{i,j}.$$

We can now define the quantified Boolean formula φ'. We let $\varphi' = \exists Y_1.\forall Y_2.\psi'$, where ψ' is defined as follows:

$$\psi' = \psi_{Y_1} \wedge (\psi_{Y_2} \rightarrow \psi'').$$

The formula ψ' can be seen as a circuit of depth 4 and weft 2. In the remainder, we will refer to ψ' as a circuit.

We verify that $\varphi \in \mathrm{QSAT}_2(\mathrm{DNF})$ if and only if $(\varphi', k) \in \Sigma^{\mathrm{p}}_2[kk]\text{-WSAT}(\Gamma_{2,4})$.

($\Rightarrow$) Assume that $\varphi \in \mathrm{QSAT}_2(\mathrm{DNF})$, i.e., that there exists a truth assignment $\beta_1 : X_1 \rightarrow \{0,1\}$ such that for all truth assignments $\beta_2 : X_2 \rightarrow \{0,1\}$ it holds that $\psi[\beta_1 \cup \beta_2]$ is true. We show that $(\varphi', k) \in \Sigma^{\mathrm{p}}_2[kk]\text{-WSAT}$. We define the truth assignment $\gamma_1 : Y_1 \rightarrow \{0,1\}$ by letting $\gamma_1(y^{\alpha}_{1,j}) = 1$ if and only if β_1 coincides with α on the variables $X_{1,j}$, for each $1 \leq j \leq k$ and each $\alpha : X_{1,j} \rightarrow \{0,1\}$. Clearly, γ_1 has weight k. Moreover, γ_1 satisfies ψ_{Y_1}. We show that for each truth assignment $\gamma_2 : Y_2 \rightarrow \{0,1\}$ of weight k it holds that $\psi'[\gamma_1 \cup \gamma_2]$ is true. Let γ_2 be an arbitrary truth assignment of weight k. We distinguish two cases: either (i) γ_2 does not satisfy ψ_{Y_2}, or (ii) γ_2 does satisfy ψ_{Y_2}. In case (i), clearly, $\psi'[\gamma_1 \cup \gamma_2]$ is true. In case (ii), we know that for each $1 \leq j \leq k$, there is exactly one $\alpha_j : X_{2,j} \rightarrow \{0,1\}$ such that $\gamma_2(y^{\alpha_j}_{2,j}) = 1$. Now let $\beta_2 : X_2 \rightarrow \{0,1\}$ be the assignment that coincides with α_j on the variables $Y_{2,j}$, for each $1 \leq j \leq k$. We know that $\psi[\beta_1 \cup \beta_2]$ is true. Then, by definition of ψ'', it follows that $\psi''[\gamma_1 \cup \gamma_2]$ is true as well. Since γ_2 was arbitrary, we can conclude that $(\varphi', k) \in \Sigma^{\mathrm{p}}_2[kk]\text{-WSAT}$.

($\Leftarrow$) Conversely, assume that $(\varphi', k) \in \Sigma^{\mathrm{p}}_2[kk]\text{-WSAT}$, i.e., that there exists a truth assignment $\gamma_1 : Y_1 \rightarrow \{0,1\}$ of weight k such that for all truth assignments $\gamma_2 : Y_2 \rightarrow \{0,1\}$ of weight k it holds that $\psi'[\gamma_1 \cup \gamma_2]$ is true. We show that $\varphi \in \mathrm{QSAT}_2$. Since ψ_{Y_1} contains only variables in Y_1, we know that γ_1 satisfies ψ_{Y_1}, i.e., that for each $1 \leq j \leq k$ there is a unique $\alpha_j : X_{1,j} \rightarrow \{0,1\}$ such that $\gamma_1(y^{\alpha_j}_{1,j}) = 1$. We define the truth assignment $\beta_1 : X_1 \rightarrow \{0,1\}$ to be the unique truth assignment that coincides with α_j for each $1 \leq j \leq k$. We show that for all truth assignments $\beta_2 : X_2 \rightarrow \{0,1\}$ it holds that $\psi[\beta_1 \cup \beta_2]$ is true. Let β_2 be an arbitrary truth assignment. We construct the truth assignment $\gamma_2 : Y_2 \rightarrow \{0,1\}$ by letting $\gamma_2(y^{\alpha}_{2,j}) = 1$ if and only if β_2 coincides with α on the variables in $Y_{2,j}$, for each $1 \leq j \leq k$ and each $\alpha : X_{2,j} \rightarrow \{0,1\}$. Clearly, γ_2 has weight k. Moreover, γ_2 satisfies ψ_{Y_2}. Therefore, since we know that $\psi'[\gamma_1 \cup \gamma_2]$ is true, we know that $\psi''[\gamma_1 \cup \gamma_2]$ is true. Then, by definition of ψ'', it follows that $\psi[\beta_1 \cup \beta_2]$ is true as well. Since β_2 was arbitrary, we can conclude that $\varphi \in \mathrm{QSAT}_2(\mathrm{DNF})$.

We observe some properties of the quantified Boolean formula $\varphi' = \exists Y_1.\forall Y_2.\psi'$. Each Y_i, for $i \in \{1,2\}$, contains at most $n' = k2^{2r}$ variables. Furthermore, the circuit ψ' has size $m' \leq O(k2^{4r} + 2^{2r}m) \leq O(k2^{4r}m)$, since the size of the subcircuits ψ_{Y_1} and ψ_{Y_2} is bounded by $O(k2^{4r})$ and the size of the subcircuit ψ'' is bounded by $O(2^{2r}m)$—this is due to the fact that we do not need to make copies of internal nodes representing literals. Finally, it is straightforward to verify that the circuit ψ' can be constructed in time $O((m')^2)$.

Since (φ', k) is an instance of $\Sigma^{\mathrm{p}}_2[kk]\text{-WSAT}(\Gamma_{2,4})$, we can apply the reduction R to obtain an equivalent instance (φ'', k'') of SAT. This reduction runs in time $f(k)(n')^{k/\lambda(k)}(m')^{O(1)}$. By first constructing (φ', k) from φ, and then constructing φ'' from (φ', k), we get a reduction R' from $\mathrm{QSAT}_2(\mathrm{DNF})$ to SAT, that runs in time $f(k)(n')^{k/\lambda(k)}(m')^{O(1)} + O((m')^2)$. We analyze the running time of this reduction R' in terms of the values n and m. Firstly:

$$f(k) \leq f(2f^{\mathrm{rev}}(n) + 1) \leq n.$$

In our analysis of the running time of R', we will use an auxiliary function λ'. In particular, it will turn out that in this analysis we need the following inequality to hold:

$$\lambda'(n) \leq \frac{\lambda(k)}{6}.$$

We do so by defining λ' as follows:

$$\lambda'(h) = \frac{\lambda(f^{\text{rev}}(h))}{6}.$$

We will also need λ' to be unbounded. Since both λ and f^{rev} are nondecreasing and unbounded, λ' is a nondecreasing and unbounded function.

We have,

$$\begin{aligned}(n')^{k/\lambda(k)} = &\quad (k2^{2r})^{k/\lambda(k)} \leq k^k 2^{2kr/\lambda(k)} \leq k^k 2^{2kn/(\lambda(k)f^{\text{rev}}(n))} \leq k^k 2^{6n/\lambda(k)} \\ \leq &\quad k^k 2^{n/\lambda'(n)} = O(\log n)^{O(\log n)} 2^{n/\lambda'(n)} = 2^{o(n)}.\end{aligned}$$

Finally, consider the factor m'. Since f^{rev} is nondecreasing and unbounded,

$$m' \leq O(k2^{4r}m) = O(\log n 2^{4n/f^{\text{rev}}(n)}m) = 2^{o(n)}m.$$

Therefore, both terms $(m')^{O(1)}$ and $O((m')^2)$ in the running time of R' are bounded by $2^{o(n)}m^{O(1)}$. Combining all these, we conclude that the running time $f(k)(n')^{k/\lambda(k)}(m')^{O(1)} + O((m')^2)$ of R' is bounded by $2^{o(n)}m^{O(1)}$. Therefore, R' is a subexponential-time reduction from QSAT_2(DNF) to SAT. This completes our proof. □

Theorem 39 gives us the following corollary, separating $\Sigma_2^{\text{p}}[*k, 2]$ from para-NP.

Corollary 40. *Assuming that there exists no subexponential-time reduction from* QSAT_2(DNF) *to* SAT*, it holds that* $\Sigma_2^{\text{p}}[*k, 2] \not\subseteq$ para-NP*.*

Proof. The parameterized problem $\Sigma_2^{\text{p}}[kk]$-$\text{WSAT}(\Gamma_{2,4})$ is in $\Sigma_2^{\text{p}}[*k, 2]$. Therefore, the result for $\Sigma_2^{\text{p}}[*k, 2]$ follows directly from Theorem 39. □

Finally, the proof of Theorem 39 extends to even stronger results for the cases of $\Sigma_2^{\text{p}}[*k, \text{SAT}]$ and $\Sigma_2^{\text{p}}[*k, \text{P}]$.

Corollary 41. *Assuming that there exists no subexponential-time reduction from* QSAT_2 *to* SAT*, it holds that* $\Sigma_2^{\text{p}}[*k, \text{SAT}] \not\subseteq$ para-NP*. Moreover, assuming that there exists no subexponential-time reduction from the extension of* QSAT_2 *to quantified Boolean circuits to* SAT*, it holds that* $\Sigma_2^{\text{p}}[*k, \text{P}] \not\subseteq$ para-NP*.*

Proof (sketch). The proof of Theorem 39 can straightforwardly be modified to show this result. □

A similar result as for the case of $\Sigma_2^{\text{p}}[*k, \text{P}]$ holds for the case of $\Sigma_2^{\text{p}}[k*]$.

Corollary 42. *Assuming that there exists no subexponential-time reduction from the extension of* QSAT_2 *to quantified Boolean circuits to* UNSAT*, it holds that* $\Sigma_2^{\text{p}}[k*] \not\subseteq$ para-co-NP*.*

Proof (sketch). The proof of Theorem 39 can straightforwardly be modified to show this result. □

5.3.3 Relation of $\Sigma_2^{\text{p}}[k*]$ and $\Sigma_2^{\text{p}}[*k, t]$ with XNP and Xco-NP

The results that we showed above also allow us to separate the classes $\Sigma_2^{\text{p}}[k*]$ and $\Sigma_2^{\text{p}}[*k, t]$ from the parameterized complexity classes XNP and Xco-NP.

Corollary 43. *If*

(i) $\Sigma_2^{\text{p}}[k*] = \text{XNP}$*,*

(ii) $\Sigma_2^{\text{p}}[k*] = \text{Xco-NP}$*,*

(iii) $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}] = \mathrm{XNP}$*, or*

(iv) $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}] = \mathrm{Xco\text{-}NP}$*,*

then for each $t \geq 3$ *there is a subexponential-time reduction from* $\mathrm{QSAT}_t(3\mathrm{CNF})$ *to* QSAT_2 *and from* $\mathrm{QSAT}_t(3\mathrm{DNF})$ *to* QSAT_2*.*

Proof. Suppose that one of the statements (i)–(iv) holds. Then for each $t \geq 3$ it holds that $\mathrm{A}[t] \subseteq \mathrm{XP} \subseteq \mathrm{XNP} \cap \mathrm{Xco\text{-}NP} \subseteq \text{para-}\Sigma_2^{\mathrm{P}}$. Therefore, the required consequence follows from Proposition 38. □

5.4 Relation Between the Classes $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k, t]$

Finally, in this section, we argue that the classes $\Sigma_2^{\mathrm{p}}[k*]$, $\Pi_2^{\mathrm{p}}[k*]$, $\Sigma_2^{\mathrm{p}}[*k, t]$ and $\Pi_2^{\mathrm{p}}[*k, t]$ are different from each other. For this, we will use some of the results that we developed in Section 5.3.

We begin with some results that are based on the assumption that NP $\neq$ co-NP.

Proposition 44. *Assuming that* NP $\neq$ co-NP*, the following statements hold:*

(i) $\Sigma_2^{\mathrm{p}}[k*] \not\subseteq \Pi_2^{\mathrm{p}}[k*]$*;*

(ii) $\Sigma_2^{\mathrm{p}}[*k, 1] \not\subseteq \Pi_2^{\mathrm{p}}[*k, \mathrm{P}]$*;*

(iii) $\Sigma_2^{\mathrm{p}}[k*] \not\subseteq \Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$*; and*

(iv) $\Sigma_2^{\mathrm{p}}[*k, 1] \not\subseteq \Sigma_2^{\mathrm{p}}[k*]$.

Proof. These results all follow from the facts that para-co-NP $\subseteq$ XNP implies NP = co-NP, and that para-NP $\subseteq$ Xco-NP implies NP = co-NP. Take for instance result (i). Suppose that $\Sigma_2^{\mathrm{p}}[k*] \subseteq \Pi_2^{\mathrm{p}}[k*]$. Since para-co-NP $\subseteq \Sigma_2^{\mathrm{p}}[k*]$ and $\Pi_2^{\mathrm{p}}[k*] \subseteq$ XNP, we get that para-co-NP $\subseteq$ XNP, and thus that NP = co-NP. The other results can be proven analogously. □

All that remains to show now is that $\Sigma_2^{\mathrm{p}}[k*] \not\subseteq \Pi_2^{\mathrm{p}}[*k, \mathrm{P}]$ and $\Sigma_2^{\mathrm{p}}[*k, 1] \not\subseteq \Pi_2^{\mathrm{p}}[k*]$. Proposition 38 allows us to show these results under the assumption that there is no subexponential-time reduction from $\mathrm{QSAT}_2(3\mathrm{DNF})$ to co-QSAT_2.

Proposition 45. *Assuming that there is no subexponential-time reduction from* $\mathrm{QSAT}_2(3\mathrm{DNF})$ *to* co-QSAT_2*, the following statements hold:*

(v) $\Sigma_2^{\mathrm{p}}[k*] \not\subseteq \Pi_2^{\mathrm{p}}[*k, \mathrm{P}]$*; and*

(vi) $\Sigma_2^{\mathrm{p}}[*k, 1] \not\subseteq \Pi_2^{\mathrm{p}}[k*]$.

Proof. We give a proof for result (v). The other result can be proven analogously.

Suppose that $\Sigma_2^{\mathrm{p}}[k*] \subseteq \Pi_2^{\mathrm{p}}[*k, \mathrm{P}]$. Then, since $\mathrm{A}[2] \subseteq \Sigma_2^{\mathrm{p}}[k*]$ and $\Pi_2^{\mathrm{p}}[*k, \mathrm{P}] \subseteq \text{para-}\Pi_2^{\mathrm{P}}$, we get that $\mathrm{A}[2] \subseteq \text{para-}\Pi_2^{\mathrm{P}}$. Then, by Proposition 38, we get that there exists a subexponential-time reduction from $\mathrm{QSAT}_2(3\mathrm{DNF})$ to co-QSAT_2. □

6 Application for the Analysis of Problems

Finally, we underline how useful the newly developed parameterized complexity classes $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k, t]$ are for the analysis of parameterized variants of problems at higher levels of the PH. We do so in two ways.

Firstly, in Section 6.1, we conduct a case study. This case study discusses the scenario where we are faced with a Π_2^{P}-complete problem, and we have several parameterizations in mind. We would like to find out for which of the parameterized variants of the problem we can find an fpt-reduction to SAT, and for which variants this is not possible. We show that the newly developed parameterized complexity classes are invaluable in such concrete complexity analyses of problems at higher levels of the PH. The Π_2^{P}-complete

problem that we consider in our case study is the problem of deciding whether any 3-coloring of the edges of a given graph can be extended to a proper 3-coloring of the entire graph.

Secondly, in the remainder of this section (that is, in Sections 6.2–6.4), we provide a compendium of parameterized variants of problems at higher levels of the PH whose complexity has been analyzed recently in the literature, and whose complexity is characterized using the classes $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k,t]$ (and their co-classes). All but one of the parameterized problems that we consider are complete for one of the parameterized complexity classes that we developed in this paper. Moreover, the problems that we consider originate in a variety of domains in computer science and artificial intelligence.

For a more extensive compendium of parameterized variants of problems at higher levels of the PH and their complexity, we refer to a technical report [30]. This report also includes parameterized variants of the problems that we consider in this section that are complete for the classes para-NP, para-co-NP, para-Σ_2^{P}, and para-Π_2^{P}—for the sake of succinctness, we omitted these variants here.

6.1 Case Study: Extending Graph Colorings

In order to illustrate the significance of the theoretical results that we developed and investigated in Sections 3–5, we demonstrate how the newly developed parameterized complexity classes are indispensable when analyzing the parameterized complexity of problems at higher levels of the PH. We do so by considering a case study, where we look at several parameterized variants of a Π_2^{P}-complete problem. In particular, we would like to investigate which of these parameterized variants admit an fpt-reduction to SAT, and for which of these parameterized variants fpt-reductions to SAT are impossible.

The problem that we consider in this case study related to extending colorings to the leaves of a graph to a coloring on the entire graph. Let $G = (V, E)$ be a graph. We will denote those vertices $v \in V$ that have degree 1 by *leaves*. We call a (partial) function $c : V \to \{1, 2, 3\}$ a 3-*coloring (of G)*. Moreover, we say that a 3-coloring c is *proper* if c assigns a color to every vertex $v \in V$, and if for each edge $e = \{v_1, v_2\} \in E$ holds that $c(v_1) \neq c(v_2)$. We consider the problem 3-Coloring-Extension. The input for this problem consists of a graph $G = (V, E)$ with n leaves, and an integer m. The question is to decide 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. This problem, in its unparameterized form, is Π_2^{P}-complete [2].

We consider several parameterized variants of this problem.

- 3-Coloring-Extension(degree), where the parameter k is the degree of G, i.e., k is the maximum number of neighbors of any vertex $v \in V$;
- 3-Coloring-Extension(#leaves), where the parameter k is the number of leaves, i.e., $k = n$;
- 3-Coloring-Extension(#col.leaves), where the parameter k is the number of pre-colored leaves, i.e., $k = m$; and
- 3-Coloring-Extension(#uncol.leaves), where the parameter k is the number of leaves that are not pre-colored, i.e., $k = n - m$.

These parameterized problems fall into several categories. The first category is that of those problems for which we manage to come up with an fpt-reduction to SAT. The parameterized problem 3-Coloring-Extension(#leaves) belongs to this category. We can iterate over all possible 3-colorings to m of the leaves of G in time $2^m \cdot \binom{n}{m}$. Since $m \leq n$ and since in this case $k = n$, iterating over all possible 3-colorings to m of the leaves of G can be done in fixed-parameter tractable time. Then, for each such coloring, checking whether it can be extended to a proper 3-coloring to the entire graph can be done with a nondeterministic algorithm. Therefore, the problem can be solved with a nondeterministic fpt-algorithm, and thus the problem is in para-NP [19, Section 2.2]. In other words, the problem 3-Coloring-Extension(#leaves) admits an fpt-reduction to SAT.

On the other hand, several of the other parameterized variants of 3-Coloring-Extension fall into a different category. This is the category of parameterized problems that are already hard for the second level of the PH, even for constant values of the parameter. This is the case for the problems 3-Coloring-Extension

(degree) and 3-Coloring-Extension(#uncol.leaves). Namely, the problem 3-Coloring-Extension is already Π_2^{P}-hard when restricted to instances where G has degree 4 and where $n = m$ [2]. Therefore, both 3-Coloring-Extension(degree) and 3-Coloring-Extension(#uncol.leaves) are para-Π_2^{P}-hard, and thus do not admit fpt-reductions to SAT unless NP = co-NP.

At this point, we would like to point out that it is entirely reasonable for negative results in this context—that is, for results that show that no fpt-reduction to SAT is possible—to be based on complexity-theoretic assumptions. For example, para-Π_2^{P}-hardness results indicate the impossibility of fpt-reductions to SAT, assuming that NP $\neq$ co-NP. This is reasonable for the following reason. If it were the case that P = NP, then also P = Π_2^{P}. In other words, in that case, the problem 3-Coloring-Extension would be solvable in polynomial time, and would thus admit an fpt-reduction to SAT for any choice of parameter. Stated conversely, any unconditional result that 3-Coloring-Extension does not admit an fpt-reduction to SAT for some parameter would imply that P $\neq$ NP.

We are left with one of the parameterized variants of 3-Coloring-Extension, namely, with 3-Coloring-Extension(#col.leaves). This parameterized problem belongs to the third category. Parameterized problems in this category seemingly do not admit an fpt-reduction to SAT, unlike problems in the first category. Additionally, for these problems it is unreasonable to hope that we can rule out fpt-reductions to SAT by showing hardness for para-Π_2^{P} or para-Σ_2^{P}. This is due to the following reason. If we fix a constant value of the parameter for 3-Coloring-Extension(#col.leaves), the problem is in NP. For example, fix a value k_0 for the parameter $k = m$. Then using a similar algorithm as the one we sketched above for the case of 3-Coloring-Extension(#leaves), we can easily show that the problem is in NP. However, since problems that are hard for para-Π_2^{P} are Π_2^{P}-hard for a finite set of parameter values [18], it immediately follows that para-Π_2^{P}-hardness for this problem would imply that NP = co-NP.

In other words, for this third category of parameterized problems the previously known parameterized complexity toolbox fails to provide an accurate picture of the computational complexity of problems. Namely, there is a significant gap between the lower and upper bounds provided by the parameterized complexity tools. In the concrete example of 3-Coloring-Extension(#col.leaves), this gap is between membership in para-Π_2^{P} and hardness for para-NP. Moreover, when considering the question whether a parameterized problem admits an fpt-reduction to SAT, such a gap indicates the lack of an answer.

In order to provide an adequate analysis of the complexity of parameterized problems in this third category, the newly developed parameterized complexity classes $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k, t]$ (and their co-classes $\Pi_2^{\mathrm{p}}[k*]$ and $\Pi_2^{\mathrm{p}}[*k, t]$) are crucial. This is illustrated by the remaining parameterized problem in our case study. The problem 3-Coloring-Extension(#col.leaves) is $\Pi_2^{\mathrm{p}}[k*]$-complete [34]. This means that we can rule out fpt-reductions to SAT for this problem, under the assumption that there exists no subexponential-time reduction to SAT from the extension of QSat_2 to quantified Boolean circuits (Corollary 42).

Overview of Parameterized Complexity Results for 3-Coloring-Extension For the sake of completeness, we conclude the treatment of our case study with a brief overview of the different parameterized problems that we discussed, together with their parameterized complexity.

3-Coloring-Extension(degree) *Instance:* a graph $G = (V, E)$ with n 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 [33, 34].

3-Coloring-Extension(#leaves) *Instance:* a graph $G = (V, E)$ with n 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 [33, 34].

3-Coloring-Extension(#col.leaves) *Instance:* a graph $G = (V, E)$ with n 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: $\Pi_2^{\mathrm{p}}[k*]$-complete [33, 34].

3-Coloring-Extension(#uncol.leaves) *Instance:* a graph $G = (V, E)$ with n 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 [33, 34].

6.2 Problems in Knowledge Representation and Reasoning

Next, we continue with our compendium of parameterized variants of problems at higher levels of the PH whose complexity has been analyzed recently in the literature, and whose complexity is characterized using the classes $\Sigma_2^{\mathrm{p}}[k*]$ and $\Sigma_2^{\mathrm{p}}[*k, t]$ (and their co-classes). We begin with a number of problems from the area of Knowledge Representation and Reasoning. Concretely, we consider several parameterized variants of the consistency problem for disjunctive answer set programs, a robust variant of the constraint satisfaction problem, and two parameterized problems related to abductive reasoning.

6.2.1 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. [7, 40]). A *disjunctive logic program* P is a finite set of rules of the form $r = (a_1 \vee \cdots \vee a_k \leftarrow b_1, \ldots, b_m, \mathit{not}\ c_1, \ldots, \mathit{not}\ c_n)$, for $k, m, n \geq 0$, where all a_i, b_j and c_l are atoms. A rule is called *disjunctive* if $k > 1$, and it is called *normal* if $k \leq 1$ (note that we only call rules with strictly more than one disjunct in the head disjunctive). A rule is called *dual-Horn* if $m \leq 1$. A program is called normal if all its rules are normal, it is called negation-free if all its rules are negation-free, and it is called dual-Horn if all its rules are dual-Horn. We let $\mathrm{At}(P)$ denote the set of all atoms occurring in P. By *literals* we mean atoms a or their negations *not* a. The *reduct* of a program P with respect to a set M of atoms, denoted P^M, is the program obtained from P by: (i) removing rules with *not* a in the body for each $a \in M$, and (ii) removing literals *not* a from all other rules [21]. 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 several parameterized variants of this problem.

One of the parameterized variants of this problem that we consider is 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.

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 [32, 33].

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 [33].

ASP-CONSISTENCY(#dual-Horn.rules) *Instance:* A disjunctive logic program P. *Parameter:* The number of rules of P that are dual-Horn. *Question:* Does P have an answer set?
Complexity: $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$-complete [33].

6.2.2 Robust Constraint Satisfaction

The following problem is based on the class of robust constraint satisfaction problems introduced by Gottlob [25] and Abramsky, Gottlob and Kolaitis [1]. These problems are concerned with the question of whether every partial assignment of a particular size can be extended to a full solution, in the setting of constraint satisfaction problems.

A *CSP instance* N is a triple (X, D, C), where X is a finite set of *variables*, the *domain* D is a finite set of *values*, and C is a finite set of *constraints*. Each constraint $c \in C$ is a pair (S, R), where $S = \mathrm{Var}(c)$, 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 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: $\Pi_2^{\mathrm{p}}[k*]$-complete [32, 33].

6.2.3 Abductive Reasoning

We conclude this section with several parameterized variants related to 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 [12].

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: $\Sigma_2^{\mathrm{p}}[*k, 1]$-complete [29].

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: $\Sigma_2^{\mathrm{p}}[*k, \mathrm{P}]$-complete [29].

6.3 Minimization Problems for Propositional Logic

We continue our compendium in this section with a number of parameterized problems that are related to minimizing DNF formulas and minimizing implicants of 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 [54].

SHORTEST-IMPLICANT-CORE(core size) *Instance:* A DNF formula φ, an implicant C of φ, and an integer k. *Parameter:* k. *Question:* Does there exists an implicant $C' \subseteq C$ of φ of size k?
Complexity: $\Sigma_2^{\mathrm{p}}[k*]$-complete [31, 33].

SHORTEST-IMPLICANT-CORE(reduction size) *Instance:* A DNF formula φ, an implicant C of φ of size n, and an integer k. *Parameter:* k. *Question:* Does there exists an implicant $C' \subseteq C$ of φ of size $n - k$?
Complexity: $\Sigma_2^{\mathrm{p}}[k*]$-complete [31, 33].

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 [31, 33].

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, solvable in fpt-time using $O(\log k)$ queries to an NP oracle, and in $\Sigma_2^{\mathrm{p}}[k*]$ [31, 33].

6.4 Other Problems

We conclude our compendium by considering a parameterized problem about the question whether every clique of a subgraph of a graph can be extended to a large enough clique of the entire graph, and a parameterized variant of a problem that arises in the area of computational social choice.

6.4.1 Clique Extensions

Let $G = (V, E)$ be a graph. A clique $C \subseteq V$ of G is a subset of vertices that induces a complete subgraph of G, i.e. $\{v, v'\} \in E$ for all $v, v' \in C$ such that $v \neq v'$. The W[1]-complete problem of determining whether a graph has a clique of size k is an important problem in the W-hierarchy, and is used in many W[1]-hardness proofs. The following related problem is complete for $\Pi_2^{\mathrm{p}}[*k, 1]$.

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 $(\lvert C\rvert + k)$-clique?
Complexity: $\Pi_2^{\mathrm{p}}[*k, 1]$-complete [33].

6.4.2 Agenda Safety in Judgment Aggregation

Finally, we consider a parameterized problem that is 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 a truth assignment to the variables occurring in Φ that makes all formulas in J true. Let $\mathcal{J}(\Phi)$ denote the set of all complete and consistent subsets of Φ. We call a sequence $\boldsymbol{J} \in \mathcal{J}(\Phi)^n$ of complete and consistent subsets a *profile*. A (resolute) *judgment aggregation procedure* for the agenda Φ and n individuals is a function $F : \mathcal{J}(\Phi)^n \to \mathcal{P}(\Phi \backslash \emptyset) \backslash \emptyset$ that returns for each profile $\boldsymbol{J}$ a non-empty set $F(\boldsymbol{J})$ of non-empty judgment sets. An example is the *majority rule* F^{maj}, where $F^{\mathrm{maj}}(\boldsymbol{J}) = \{J^*\}$ and where $\varphi \in J^*$ if and only if φ occurs in the majority of judgment sets in $\boldsymbol{J}$, for each $\varphi \in \Phi$. We call F *complete* and *consistent*, if each $J^* \in F(\boldsymbol{J})$ is complete and consistent, respectively, for every $\boldsymbol{J} \in \mathcal{J}(\Phi)^n$. For instance, the majority rule F^{maj} is complete, whenever the number n of individuals is odd. An agenda Φ is *safe* with respect to an aggregation procedure F, if F is consistent when applied to profiles of judgment sets over Φ. We say that an agenda Φ satisfies the *median property (MP)* if every inconsistent subset of Φ has itself an inconsistent subset of size at most 2. Safety for the majority rule can be characterized in terms of the median property as follows: an agenda Φ is safe for the majority rule if and only

if Φ satisfies the MP [13, 43]. The problem of deciding whether an agenda satisfies the MP is Π_2^{P}-complete [13].

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: $\Pi_2^{\mathrm{P}}[k*]$-hard [14, 15, 16].

7 Conclusion

We developed a general theoretical framework that supports the classification of parameterized problems on whether they admit an fpt-reduction to SAT or not. Our theory is based on two new hierarchies of complexity classes, the k-$*$ and $*$-k hierarchies. We developed basic structural results for these parameterized complexity classes, that make it possible to conveniently use them in the analysis of concrete parameterized problems. Additionally, we underpinned the robustness of our theory by providing a characterization of the new complexity classes in terms of weighted QBF satisfiability, alternating Turing machines, and first-order model checking. Also, we provide evidence that the newly developed classes are different from parameterized complexity classes that are known from the literature. Finally, to indicate how well the parameterized complexity classes that we developed can be used to analyze the parameterized complexity of problems at higher levels of the PH, we considered a case study and we provided a compendium of natural parameterized problems from various domains that are complete for the new classes.

In this paper, we extended and combined ideas known from the theory of parameterized complexity as it is usually applied to problems in NP, and applied these ideas to the setting of parameterized variants of problems at the second level of the PH and higher. Even though the proofs in this paper are mostly based on ideas and proof techniques that were known from the literature, the results in this paper provide useful insights about the inherent computational complexity of parameterized variants of problems at the second level of the PH. For instance, for one of the parameterized analogues of the W-hierarchy that we considered (namely, the classes $\Sigma_2^{\mathrm{P}}[*k, t]$) the structure of the W-hierarchy is reflected, whereas for the other parameterized analogue of the W-hierarchy that we considered (the class $\Sigma_2^{\mathrm{P}}[k*]$), this structure vanishes. As a consequence, we were able to show that problems in the latter class do not admit fpt-reductions to SAT using weaker complexity-theoretic assumptions than we needed for problems in the former classes. Discoveries such as these are nontrivial outcomes of our adaptation of known parameterized complexity results and techniques to the setting of parameterized variants of problems at the second level of the PH.

Directions for Future Research The results in this paper provide a relatively clear understanding of the parameterized complexity class originally defined as the k-$*$ hierarchy. The $*$-k hierarchy seems to be more intricate and we have focused only on two of its levels—namely, on the classes $\Sigma_2^{\mathrm{P}}[*k, 1]$ and $\Sigma_2^{\mathrm{P}}[*k, \mathrm{P}]$. A more comprehensive study of the $*$-k hierarchy including alternative characterizations of its classes using first-order model checking and machine models comprises an interesting topic for future research.

Another interesting topic for future investigations is to get a deeper understanding of the separation results that we developed Section 5.3. For instance, we showed that $\Sigma_2^{\mathrm{P}}[k*] \not\subseteq$ para-co-NP unless there exists a subexponential-time reduction from QSAT_2 to UNSAT. It would greatly improve our understanding of the class $\Sigma_2^{\mathrm{P}}[k*]$ if we could establish a similar result that is based on more widely believed complexity-theoretic assumptions. Alternatively, a more in-depth investigation into the (im)possibility of constructing subexponential-time reductions from (variants of) QSAT_2 to SAT or UNSAT would allow us to better appreciate the results from Section 5.3.

In the development of theoretical parameterized complexity tools in this paper, we focused our attention on the range between the first and the second level of the PH. This is partially due to the fact that many

natural problems lie there [48], and this is partially because SAT solvers perform significantly better in practice than solvers for problems at higher levels of the PH or for PSPACE-complete problems. However, it would also be interesting to identify fpt-reductions from problems at higher levels of the PH to problems at a lower level of the PH. For instance, for a Σ_3^{P}-complete problem, an fpt-reduction to a problem in Σ_2^{P} could be used as a starting point for algorithmic methods that could work better in practice than existing solving methods for the problem. With this more general setting in mind, it would be helpful to develop tools that can be used to provide evidence that fpt-reductions to higher levels of the PH are not possible. The classes of the k-$*$ and $*$-k hierarchies can straightforwardly be generalized to analogous variants at higher levels of the PH. For example, for the third level of the PH, one could define the classes $\Sigma_3^{\mathrm{p}}[**k,t]$, $\Sigma_3^{\mathrm{p}}[*k*]$, $\Sigma_3^{\mathrm{p}}[*kk,t]$, $\Sigma_3^{\mathrm{p}}[k**]$, $\Sigma_3^{\mathrm{p}}[k*k,t]$, $\Sigma_3^{\mathrm{p}}[kk*]$, and $\Sigma_3^{\mathrm{p}}[kkk,t]$. Several results from this paper also carry over straightforwardly to this more general setting—for instance, the collapse result of Theorem 2 extends to any hierarchy $\Sigma_i^{\mathrm{p}}[\omega*,t]$ for $\omega \in \{*,k\}^{i-1}$. We hope that this paper provides a starting point for further developments.

Acknowledgments

This research was supported by the European Research Council (ERC), project 239962 (COMPLEX REASON), and the Austrian Science Fund (FWF), project P26200 (Parameterized Compilation). We gratefully acknowledge the many helpful suggestions made by the anonymous reviewers.

References

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

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

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

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

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

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

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

[8] Marek Cygan, Fedor V Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

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

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

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

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

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

[14] Ulle Endriss, Ronald de Haan, and Stefan Szeider. Parameterized complexity results for agenda safety in judgment aggregation. In *Proceedings of the 5th International Workshop on Computational Social Choice (COMSOC 2014)*. Carnegie Mellon University, June 2014. Revised version available as [16].

[15] Ulle Endriss, Ronald de Haan, and Stefan Szeider. Parameterized complexity results for agenda safety in judgment aggregation. In Gerhard Weiss, Pinar Yolum, Rafael H. Bordini, and Edith Elkind, editors, *Proceedings of AAMAS 2015, the 14th International Conference on Autonomous Agents and Multiagent Systems*, pages 127–136. IFAAMAS/ACM, 2015.

[16] Ulle Endriss, Ronald de Haan, and Stefan Szeider. Parameterized complexity results for agenda safety in judgment aggregation. Technical Report AC-TR-16-005, Algorithms and Complexity Group, TU Wien, 2016. Revised version of [14].

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

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

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

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

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

[22] Oded Goldreich. *Computational complexity: a conceptual perspective.* Cambridge University Press, Cambridge, 2008.

[23] Oded Goldreich. *P, NP, and NP-Completeness: The Basics of Complexity Theory.* Cambridge University Press, Cambridge, 2010.

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

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

[26] Georg Gottlob, Francesco Scarcello, and Martha Sideri. Fixed-parameter complexity in AI and non-monotonic reasoning. *Artificial Intelligence*, 138(1-2):55–86, 2002.

[27] Ronald de Haan. An overview of non-uniform parameterized complexity. Technical Report TR15-130, *Electronic Colloquium on Computational Complexity* (ECCC), 2015.

[28] Ronald de Haan, Martin Kronegger, and Andreas Pfandler. Fixed-parameter tractable reductions to sat for planning. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015), July 25–31, 2015, Buenos Aires, Argentina*, pages 2897–2903, 2015.

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

[30] Ronald de Haan and Stefan Szeider. Compendium of parameterized problems at higher levels of the Polynomial Hierarchy. Technical Report TR14–143, *Electronic Colloquium on Computational Complexity* (ECCC), 2014.

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

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

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

[34] Ronald de Haan and Stefan Szeider. Machine characterizations for parameterized complexity classes beyond para-NP. In *Proceedings of the 41st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2015)*, volume 8939 of *Lecture Notes in Computer Science*, pages 217–229. Springer Verlag, January 2015.

[35] Ronald de Haan and Stefan Szeider. 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, April 25-29, 2016*, pages 453–462. AAAI Press, 2016.

[36] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Series in Computer Science. Addison-Wesley-Longman, second edition, 2001.

[37] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. of Computer and System Sciences*, 63(4):512–530, 2001.

[38] Daniel Kroening, Joël Ouaknine, Ofer Strichman, Thomas Wahl, and James Worrell. Linear completeness thresholds for bounded model checking. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *23rd International Conference on Computer Aided Verification (CAV 2011)*, volume 6806 of *LNCS*, pages 557–572. Springer Verlag, 2011.

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

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

[41] João P. Marques-Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability*, pages 131–153. IOS Press, 2009.

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

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

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

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

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

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

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

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

[50] Carsten Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer Verlag, 2005.

[51] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012.

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

[53] G. S. Tseitin. Complexity of a derivation in the propositional calculus. *Zap. Nauchn. Sem. Leningrad Otd. Mat. Inst. Akad. Nauk SSSR*, 8:23–41, 1968. English translation reprinted in [49].

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

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