Masterarbeit

Complexity of Parameterized Counting

Anselm Haak
Matr.-Nr.: 2790850

04.06.2015

Leibniz Universität Hannover
Institut für theoretische Informatik

Erstprüfer:	Heribert Vollmer
Zweitprüfer:	Arne Meier
Betreuer:	Heribert Vollmer

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Contents

1 Introduction

The field of parameterized complexity theory is a rather new one and has been studied quite intensively in recent years. Recently, Creignou and Vollmer classified parameterized satisfiability problems over different fragments of Post's lattice [CV]. Beside the parameterized weighted satisfiability problem and the enumeration of all of its solutions, they examined a parameterized version of #SAT, where the number of satisfying assignments of a certain weight are counted. Chauhan and Rao on the other hand introduced the class W[P]-PFPT [CR], which is a parameterized version of the class PP. They studied its properties and showed that many properties of PP also apply to W[P]-PFPT. One of the exceptions is closure under intersection.
In chapter 1 we want to classify a W[P]-PFPT-complete weighted satisfiability problem over the fragments from Post's lattice. Additionally, we do the same for the new class W[ThreshSAT], which we define in analogy to W[SAT].
In chapter 2, we transfer another result from classical complexity theory which was already known before closure of PP under intersection was shown: PP contains the Boolean algebra over NP. We will show the parameterized analogon, that is, W[P]-PFPT contains the Boolean algebra over W[P].
In chapter 3, we transfer two more counting classes to the parameterized world: $\oplus$P and $\mathrm{C}_=\mathrm{P}$. For both classes we show some basic results and build a foundation for further study.
In chapter 4 we generalize the class W[P]-PFPT by introducing the parameterized counting quantifier C[κ]. A similar quantifier is used in classical complexity theory in order to define the counting hierarchy, which is an extension of the polynomial time hierarchy. Since we focus on counting classes in this thesis, we leave out the $\exists$- and $\forall$-quantifiers and only examine the hierarchy of classes defined with only the C[κ]-quantifier.

2 Background

2.1 Preliminaries

Throughout this work, we use the common notations from complexity theory and parameterized complexity theory. We choose to define parameterized languages such that the parameter is always part of the input as we want to examine closure under set-operations. Although most of the notations should be clear, we briefly introduce the most important definitions.

Definition 2.1. Let Σ be an alphabet. A parameterized problem is a set of tuples (x,k), where $x \in \Sigma^*$ is the instance and $k \in \mathbb{N}$ is the parameter.

An fpt-algorithm is an algorithm that for any input (x,k) has a running time of at most $f(k) \cdot |x|^c$ steps, where f is a computable function and $c \in \mathbb{N}$. The class FPT is the class of all fpt-decidable parameterized problems.
A k-restricted nondeterministic Turing-machine (or k-restricted NTM) is a nondeterministic Turing-machine that has fpt-runtime and uses at most $f(k) \cdot \log|x|$ nondeterministic bits on input (x,k).
A Boolean circuit is a directed acyclic graph where the inner nodes are marked with Boolean functions or families of Boolean functions, each leaf is marked with a constant or one of the input variables and one node is marked as the output node.
Let φ be a Boolean circuit (or formula). Then $\mathrm{Var}(\varphi)$ is the set of all variables occurring in φ. $\varphi[a/b]$ is the circuit (or formula) φ with all occurrences of a replaced by b (a and b being variables or constants). The weight of an assignment to the variables of φ is the number of variables set to 1. $\#\mathrm{Sat}(\varphi)$ is the number of satisfying assignments of φ and $\#\mathrm{Sat}_k(\varphi)$ is the number of satisfying weight-k-assignments of φ.
Let A, B be two parameterized problems. Then

$$A \triangle B := \{(x,k) \mid ((x,k) \in A \text{ and } (x,k) \notin B) \text{ or } ((x,k) \notin A \text{ and } (x,k) \in B)\}$$

is the symmetric difference of A and B and

$$A \uplus B := \{(0x,k) \mid (x,k) \in A\} \cup \{(1x,k) \mid (x,k) \in B\}$$

is the marked union of A and B. The parameterized Cartesian product of A and B is

$$A \times_\kappa B = \{(x_1, x_2, k) \mid (x_1,k) \in A, (x_2,k) \in B\}.$$

For a Turing machine M, $L(M)$ is the language accepted by M and $M(x)$ denotes the computation of M on input x. If M is an oracle Turing machine and A is a language, $L(M,A)$ is the language accepted by M with oracle A and $M^A(x)$ denotes the computation of M with oracle A on input x. If M is a nondeterministic Turing machine, a subscript can be added to address specific computation paths: $M_y(x)$ denotes the computation path encoded by y of M on input x. Furthermore, $\#\mathrm{acc}_M(x)$ and $\#\mathrm{rej}_M(x)$ denote the number of accepting and rejecting computation paths of M on input x, respectively.
$\leq^{\mathrm{fpt}}$ denotes fpt-many-one-reducibility and $\leq^{\mathrm{fpt\text{-}T}}$ denotes fpt-Turing-reducibility. For a complexity class $\mathcal{K}$, $[\mathcal{K}]^{\mathrm{fpt}}$ denotes the closure of $\mathcal{K}$ under $\leq^{\mathrm{fpt}}$.

2.2 Function Classes

In the following, we want to define the parameterized function classes used in this paper, recall a few known and present some new results.

Definition 2.2. A parameterized function is a function $\Sigma^* \times \mathbb{N} \to \mathbb{N}$, where Σ is an alphabet.

Definition 2.3. FFPT is the class of all parameterized functions computable in fpt-runtime.

Definition 2.4. A p-bounded sum (product) of parameterized functions is a sum $g_1 + \cdots + g_{t(k)}$ (product $g_1 \cdot \cdots \cdot g_{t(k)}$) where t is a computable function and k is the parameter.

Definition 2.5. #W[P] is the class of all parameterized functions f for which there is a k-restricted NTM M such that

$$f(x,k) = \#\mathrm{acc}_M(x,k).$$

Definition 2.6. FFPT[κ] is the class of all FFPT-functions that are bounded by $|x|^{h(k)}$, where h is some computable function.

Lemma 2.7. FFPT[κ] $\subseteq$ #W[P].

Proof. Let $f \in \mathrm{FFPT}[\kappa]$. On input (x,k), a k-restricted NTM can guess $\lceil \log_2(f(x,k)) \rceil$ nondeterministic bits and then accept on the first $f(x,k)$ computation paths. Since f is bounded by $|x|^{h(k)}$ for some computable function h, $\lceil \log_2(f(x,k)) \rceil$ is bounded by $2 \cdot h(k) \cdot \log|x|$. □

Lemma 2.8. #W[P] *is closed under p-bounded sums and products.*

Proof. For summations, the k-restricted NTMs for the different functions can be simulated in parallel. For products, they can be simulated in succession. In both cases the running time and the nondeterminism stay within the needed bounds. □

Definition 2.9. Gap-FPT is the class of all parameterized functions f for which there is a k-restricted NTM M such that

$$f(x,k) = \#\mathrm{acc}_M(x,k) - \#\mathrm{rej}_M(x,k).$$

The following lemma has been shown in [CR].

Lemma 2.10. Gap-FPT *is closed under p-bounded sums and products.*

Analogously to the corresponding results for GapP in classical complexity theory (see e.g. [Vol]), the following characterizations of Gap-FPT can be shown:

Lemma 2.11. Gap-FPT = #W[P] − #W[P] = FFPT[κ] − #W[P] = #W[P] − FFPT[κ]

Proof. <u>Gap-FPT ⊆ #W[P] − #W[P]</u>: Let $f \in$ Gap-FPT. Then there is a k-restricted NTM M such that $f(x,k) = \#\mathrm{acc}_M(x,k) - \#\mathrm{rej}_M(x,k)$ $\forall x,k$. Let M' be M with swapped accepting and rejecting states. Then we have $f(x,k) = \#\mathrm{acc}_M(x,k) - \#\mathrm{acc}_{M'}(x,k)$ and hence $f \in$ #W[P] − #W[P].

<u>#W[P] − #W[P] ⊆ FFPT[κ] − #W[P]</u>: Let $f \in$ #W[P] − #W[P]. There are k-restricted NTM M_1, M_2 such that $f(x,k) = \#\mathrm{acc}_{M_1}(x,k) - \#\mathrm{acc}_{M_2}(x,k)$ $\forall x,k$. W.l.o.g. we can assume that on

every input (x,k), M_1 uses exactly $g(k)\cdot\lfloor\log_2|x|\rfloor$ nondeterministic bits on all computation paths, where g is some computable function. Then we have

$$\begin{aligned}f(x,k) = {} & (\#\mathrm{acc}_{M_1}(x,k)+\#\mathrm{rej}_{M_1}(x,k))-(\#\mathrm{acc}_{M_2}(x,k)+\#\mathrm{rej}_{M_1}(x,k))\\ = {} & 2^{g(k)\cdot\lfloor\log_2|x|\rfloor}-(\#\mathrm{acc}_{M_2}(x,k)+\#\mathrm{rej}_{M_1}(x,k)).\end{aligned}$$

Note that $2^{g(k)\cdot\lfloor\log_2|x|\rfloor}$ can be computed in FPT-time and is bounded by $|x|^{g(k)}$. Let M_1' be M_1 with swapped accepting and rejecting states. Now we have

$$f(x,k)=2^{g(k)\cdot\lfloor\log_2|x|\rfloor}-\underbrace{(\#\mathrm{acc}_{M_2}(x,k)+\#\mathrm{acc}_{M_1'}(x,k))}_{\in\#\mathrm{W[P]}}$$

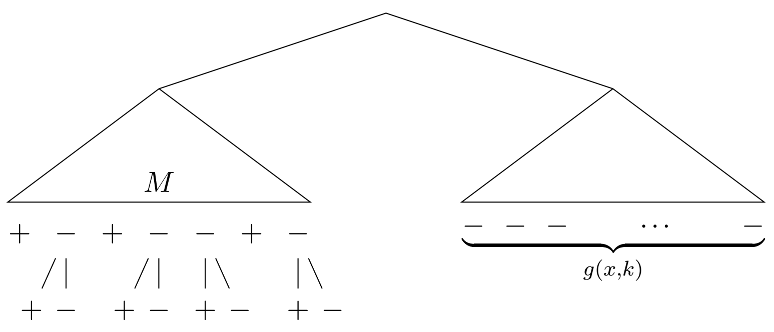
and hence $f(x,k)\in\mathrm{FFPT}[\kappa]-\#\mathrm{W[P]}$.

<u>$\mathrm{FFPT}[\kappa]-\#\mathrm{W[P]}\subseteq\#\mathrm{W[P]}-\mathrm{FFPT}[\kappa]$</u>: Let $f\in\mathrm{FFPT}[\kappa]-\#\mathrm{W[P]}$. There is $h\in\mathrm{FFPT}[\kappa]$ and a k-restricted NTM M such that $f(x,k)=h(x,k)-\#\mathrm{acc}_M(x,k)\ \forall x,k$. As before, w. l. o. g. we can assume that on every input (x,k) M uses exactly $g(k)\cdot\lfloor\log_2|x|\rfloor$ nondeterministic bits on all computation paths, where g is some computable function. Then we have

$$\begin{aligned}f(x,k) = {} & (h(x,k)+\#\mathrm{rej}_M(x,k))-(\#\mathrm{acc}_M(x,k)+\#\mathrm{rej}_M(x,k))\\ = {} & \underbrace{(\underbrace{h(x,k)}_{\in\mathrm{FFPT}[\kappa]\subseteq\#\mathrm{W[P]}}+\underbrace{\#\mathrm{rej}_M(x,k)}_{\in\#\mathrm{W[P]}})}_{\#\mathrm{W[P]}}-2^{g(k)\cdot\lfloor\log_2|x|\rfloor}.\end{aligned}$$

Hence, $f(x,k)\in\#\mathrm{W[P]}-\mathrm{FFPT}[\kappa]$.

<u>$\#\mathrm{W[P]}-\mathrm{FFPT}[\kappa]\subseteq\mathrm{Gap\text{-}FPT}$</u>: Let $f\in\#\mathrm{W[P]}-\mathrm{FFPT}[\kappa]$. There are $g(x,k)\in\mathrm{FFPT}[\kappa]$ and a k-restricted NTM M such that $f(x,k)=\#\mathrm{acc}_M(x,k)-g(x,k)$. Now let M' be the k-restricted NTM that guesses one of two possible computations: In the first, the whole computation of M is done, but for each rejecting path one rejecting and one accepting path is created (via an additional nondeterministic bit). In the second, a computation tree with g(x,k) leafs is built that rejects on every path.



Now we have

$$\begin{aligned}\#\mathrm{acc}_{M'}(x,k)-\#\mathrm{rej}_{M'}(x,k) = {} & (\#\mathrm{acc}_M(x,k)+\#\mathrm{rej}_M(x,k))-(\#\mathrm{rej}_M(x,k)+g(x,k))\\ = {} & \#\mathrm{acc}_M(x,k)-g(x,k)\\ = {} & f(x,k),\end{aligned}$$

which yields $f\in\mathrm{Gap\text{-}FPT}$. □

2.3 The class W[P]-PFPT

Beside the main classes in parameterized complexity theory like FPT and W[P], the class W[P]-PFPT was introduced quite recently by Chauhan and Rao [CR], who investigated some of its properties. The class can be seen as a parameterized version of the class PP. We want to define it and recall the results that will be needed later on.

Definition 2.12. W[P]-PFPT is the class of all parameterized problems L for which there is a k-restricted NTM M such that

$$x \in L \;\Leftrightarrow\; \text{The majority of computation paths of } M \text{ on input } x \text{ accepts.}$$

Remark 2.13. W[P]-PFPT *can be equivalently defined in terms of k-restricted probabilistic Turing-machines that accept iff the probability of reaching an accepting path is $> \frac{1}{2}$.*

Theorem 2.14. *Let L be a parameterized problem. The following statements are equivalent:*

1. $L \in$ W[P]-PFPT
2. *There is a k-restricted Turing machine M s. t.:* $(x,k) \in L \Leftrightarrow \#acc_M(x,k) - \#rej_M(x,k) > 0$.
3. *There is a function $f \in$ Gap-FPT s. t.:* $(x,k) \in L \Leftrightarrow f(x,k) > 0$
4. *There is a $B \in$ FPT and a computable function $f : \mathbb{N} \to \mathbb{N}$ s. t.:* $(x,k) \in L \Leftrightarrow |\{y \in \{0,1\}^{f(k)\cdot\log n} \mid (x,y,k) \in B\}| \geq 2^{f(k)\cdot\log n - 1} + 1$

Theorem 2.15. W[P]-PFPT *is closed under complementation.*

Lemma 2.16. W[P]-PFPT *is closed under symmetric difference.*

The following Lemma, although not shown by Chauhan and Rao, is quite obvious:

Lemma 2.17. W[P] $\subseteq$ W[P]-PFPT.

Proof. Let $L \in$ W[P] via the k-restricted NTM M. W. l. o. g. there is a computable function f such that on every input (x,k) M uses exactly $f(k) \cdot \lfloor \log_2 |x| \rfloor$ nondeterministic bits on all computation paths. Let M' be M with another nondeterministic branch added in the beginning of the computation adding exactly $2^{f(k)\cdot\lfloor \log_2 |x| \rfloor}$ accepting paths. Now at least one computation path of M accepts iff more than half of the computation paths of M' accept. Therefore M' accepts L in PP-fashion. □

For some results we will need classes defined using oracle Turing machines where the queries are bounded in a certain way:

Definition 2.18. Let $\mathcal{K}$ be a complexity class. Then FPT$^{\mathcal{K}[f(k),\kappa]}$ (resp. W[P]-PFPT$^{\mathcal{K}[f(k),\kappa]}$) is the class of parameterized problems that can be decided by an FPT-oracle Turing machine (resp. W[P]-PFPT-oracle Turing machine) M with the following restriction: There is a computable function f such that on every input (x,k) M uses at most $f(k)$ oracle queries (per computation path) and for every query (q,k') it holds that $k' = f(k)$.

While many properties of the class PP could be transferred to W[P]-PFPT, there are a few exceptions.
The widely known proof for closure of PP under intersection by Beigel, Reingold and Spielman uses closure of GapP under products where the number of factors is given by an FP-function. The corresponding property does not hold for Gap-FPT due to the logarithmic factor restricting the used nondeterminism of k-restricted NTMs:

Remark 2.19. *For input (x,k), the product of fpt-many factors $2^{f(k)\log|x|}$ where f is a computable function, cannot be the number of computation paths of any k-restricted NTM.*

Proof. For a computable function g and a constant a, let $g(k)\cdot\log|x|^a$ be the number of factors. Then the product has the form

$$2^{f(k)\cdot g(k)\cdot\log|x|\cdot|x|^a} \geq 2^{h(k)\cdot|x|^a},$$

where $h(k) = f(k)\cdot g(k)$. Therefore the number of needed nondeterministic bits is $\geq h(k)\cdot|x|^a$, which is not k-restricted. □

Thus, the proof of Beigel, Reingold and Spielman cannot be transferred to W[P]-PFPT. Similar considerations prevent the known proofs for the different amplification results that hold for the class PP from working for W[P]-PFPT.
Although it may be possible that some of the results where this problem occurs in the known proofs from classical complexity theory can be shown with different techniques, the problem is inherent in the definition of k-restriction and cannot be solved in general. For now, closure of W[P]-PFPT under intersection and whether some kinds of amplification apply for W[P]-PFPT remain open problems.

3 Parameterized Complexity of Threshold Satisfiability

In order to better understand the complexity of satisfiability problems, an often used technique is classifying said problems over the different clones given by Post's lattice. By doing so, we can identify where the difficult parts of the problems lie, such as the negation of implication for the satisfiability problem of propositional logic.
The same has been done for some parameterized satisfiability problems. Namely, the fragments of p-WCIRCSAT$_=$(BF), p-WSAT$_=$(BF) and their respective counting versions have been classified in [CV].
In the following we will classify the fragments of a threshold-satisfiability-problem that is complete for W[P]-PFPT. The reason we are not using some form of MajoritySAT is that showing hardness for fragments of a version of MajoritySAT is quite difficult.
Additionally we will introduce the class W[ThreshSAT] and classify fragments of its defining parameterized threshold-satisfiability-problem.

3.1 Preliminaries and Definitions

The following result was stated in [FG] and is needed to prove hardness of circuit satisfiability problems in parameterized complexity.

Fact 3.1. *Let $t : \mathbb{N}_0 \to \mathbb{N}_0$ be a function s. t. $t(n) \geq n \ \forall n \in \mathbb{N}_0$. If $L \subseteq \{0,1\}^*$ can be decided by a deterministic Turing machine in time $t(n)$, then there is a uniform family $(C_n)_{n\geq 0}$ of circuits s. t. $(C_n)_{n\geq 0}$ decides L and $||C_n|| \in \mathcal{O}(t(n)^2)$.*

Remark 3.2. *Here, uniformity means that a deterministic Turing machine can compute the circuit C_n in time polynomial in $||C_n||$ for all n. A circuit family is said to decide a language A if the Boolean function it computes is the characteristic function of A.*

We now want to introduce parameterized versions of ThresholdSAT for circuits and formulae.

Definition 3.3. The parameterized weighted threshold-satisfiability-problem over a class ofcircuits $\mathcal{C}$ is defined as follows: circuits $\mathcal{C}$ is defined as follows:

Problem:	p-WCIRCThreshSAT$_=(\mathcal{C})$
Input:	Circuit $C \in \mathcal{C}$, $t \in \mathbb{N}$, $k \in \mathbb{N}$
Question:	Does C have at least t satisfying assignments of weight exactly k?
Parameter:	k

Definition 3.4. The parameterized weighted threshold-satisfiability-problem over a class of formulae $\mathcal{C}$ is defined as follows:

Problem:	p-WThreshSAT$_=(\mathcal{C})$
Input:	Propositional formula $\varphi \in \mathcal{C}$, $t \in \mathbb{N}$, $k \in \mathbb{N}$
Question:	Does φ have at least t satisfying assignments of weight exactly k?
Parameter:	k

Now we can introduce the class W[ThreshSAT] in analogy to W[SAT].

Definition 3.5. The class W[ThreshSAT] is defined as

$$\mathrm{W[ThreshSAT]} := [\text{p-WThreshSAT}_=(\mathrm{BF})]^{\mathrm{fpt}}$$

3.2 Parameterized Complexity of ThreshSAT-problems

The parameterized weighted threshold-satisfiability-problem is complete for W[ThreshSAT] by definition, whereas W[P]-PFPT-completeness of p-WCIRCThreshSAT$_=$(BF) has to be shown, which we will do in the following.

3.2.1 p-WCIRCThreshSAT$_=$(BF)

First, we want to combine Fact 3.1 and Lemma 2.14 into a new lemma that better serves our purpose in proving hardness following the proof for W[P]-hardness of p-WSAT in [FG].

Lemma 3.6. *Let Σ be an alphabet and M a k-restricted NTM that works on inputs $(x,k) \in \Sigma^* \times \mathbb{N}$. Then there are a computable function f and an fpt-algorithm computing on input $(x,k) \in \Sigma^* \times \mathbb{N}$ a circuit $C_{(x,k)}$ with $f(k) \cdot \lfloor \log_2 |x| \rfloor$ input nodes s. t.*

$$\#\mathit{Sat}(C_{(x,k)}) = \#\mathit{acc}_M(x,k)$$

Proof. W. l. o. g. we can assume that for all inputs (x,k) all computation paths of M use exactly $f(k) \cdot \lfloor \log_2 |x| \rfloor$ nondeterministic bits and have a running time of at most $f(k) \cdot |x|^a$ steps for some constant a and computable function f. Moreover, we can assume that k is encoded unary (k is given as the string 1^k). Let B be the path language of M, that is,

$$B := \left\{ (x,y,k) \;\middle|\; \begin{array}{c} x \in \Sigma^*, y \in \{0,1\}^{f(k)\cdot\lfloor \log_2 |x| \rfloor}, k \in \mathbb{N} \text{ and } y \text{ encodes an} \\ \text{accepting computation path of } M \text{ on input } (x,k) \end{array} \right\}$$

Obviously, $B \in \mathrm{FPT}$. By Fact 3.1 we get a uniform family of circuits $(C_m)_{m\in\mathbb{N}}$ defined for $m = n + f(k) \cdot \lfloor \log_2 n \rfloor + k \ \ \forall n,k \in \mathbb{N}$ with:

(i) $\forall (x,y,k) \in \{0,1\}^n \times \{0,1\}^{f(k)\cdot\lfloor \log_2 n \rfloor} \times \{1^k\}$: $C_m((x,y,k)) = 1 \Leftrightarrow (x,y,k) \in B$

(ii) $||C_m|| \le (f(k) \cdot p(n))^2$, where p is some polynomial

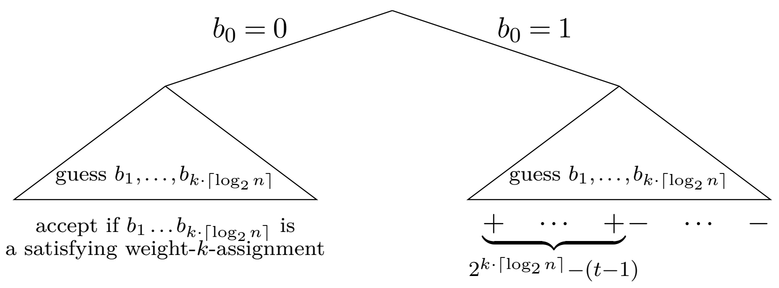
Let $C_{(x,k)}$ be the circuit $C_{|x|+f(k)\cdot\lfloor \log_2 |x| \rfloor + k}$ after setting the first $|x|$ input bits to the bits of x and the last k input bits to 1 (since k is encoded unary). Now we have $||C_{(x,k)}|| \le (f(k) \cdot p(|x|))^2$ for some polynomial p and the number of assignments satisfying $C_{(x,k)}$ is exactly the number of $y \in \{0,1\}^{f(k)\cdot\lfloor \log_2 |x| \rfloor}$ such that $(x,y,k) \in B$. Therefore we have

$$\#\mathrm{Sat}(C_{(x,k)}) = \#\mathrm{acc}_M(x,k).$$

Furthermore, the circuit $C_{(x,k)}$ has $f(k) \cdot \lfloor \log_2 |x| \rfloor$ input bits and since the family of circuits $(C_m)_{m\in\mathbb{N}}$ is uniform, the mapping $(x,k) \mapsto C_{(x,k)}$ can be computed by an fpt-algorithm. □

Theorem 3.7. p-WCIRCThreshSAT$_=$(*BF*) *is* W[P]-PFPT-*complete under* $\leq^{\mathrm{fpt}}$.

Proof. Membership: We will give an W[P]-PFPT-algorithm deciding p-WCIRCThreshSAT$_=$(BF). The idea is to split up the computation in two parts using a nondeterministic bit.
In the first part an assignment of weight k is guessed by nondeterministically choosing the indices of the variables set to 1. Then the machine accepts depending on whether the circuit is satisfied by that assignment.
In the second part equally many nondeterministic bits as in the first part are used and the needed number of paths is accepted such that C has $\geq t$ satisfying assignments iff more than half of the computation paths accept.



$b_0 = 0$
$b_0 = 1$
guess $b_1, \dots, b_{k\cdot\lceil\log_2 n\rceil}$
guess $b_1, \dots, b_{k\cdot\lceil\log_2 n\rceil}$
accept if $b_1 \dots b_{k\cdot\lceil\log_2 n\rceil}$ is a satisfying weight-k-assignment
$\underbrace{+ \cdots +}_{2^{k\cdot\lceil\log_2 n\rceil}-(t-1)} - \cdots -$

In the algorithm we guess all needed nondeterministic bits at the very beginning. Let $n := |\mathrm{Var}(C)|$ be the number of variables in the circuit C and $\{x_0, \dots, x_{n-1}\}$ be the variables. The algorithm works as follows:

```
Input: Circuit $C$, $t \in \mathbb{N}$, $k \in \mathbb{N}$
Nondeterministically guess $(b_i)_{i\in\{0,\dots,k\cdot\lceil\log_2 n\rceil\}}, b_i \in \{0,1\}$
if $b_0 = 0$ then
    $index_i \leftarrow b_{1+(i-1)\cdot\lceil\log_2 n\rceil} \dots b_{i\cdot\lceil\log_2 n\rceil}$ $\forall 1 \leq i \leq k$
    if ($index_i > n$ for some $i$) or ($index_i \leq index_j$ for some $i > j$) then
        reject
    end if
    $I \leftarrow \{index_i \mid 1 \leq i \leq k\}$
    for $i \in \{1, \dots, n\}$ do
        $x_i \leftarrow \begin{cases} 1, & \text{if } i \in I \\ 0, & \text{else} \end{cases}$
    end for
    if $C(x_1, \dots, x_n) = 1$ then
        accept
    else
        reject
    end if
else
    if $b_1 \dots b_{k\cdot\lceil\log_2 n\rceil} < 2^{k\cdot\lceil\log_2 n\rceil} - (t-1)$ then
        accept
    else
        reject
    end if
end if
```

We have to be careful in constructing this algorithm as not all sequences of $k \cdot \lceil \log_2 n \rceil$ bits correspond to assignments of weight k and the elements are not ordered. We deal with this by rejecting if the sequence does not correspond to an ordered sequence of k different variables before evaluating the circuit on the corresponding assignment in the first computation path ($b_0 = 0$). The number of accepting paths in the second computation path must be chosen accordingly: For correctness of our choice, note that we have a total of $k \cdot \lceil \log_2 n \rceil + 1$ nondeterministic bits resulting in $2^{k \cdot \lceil \log_2 n \rceil + 1}$ computation paths. Thus, the machine accepts iff more than $\frac{2^{k \cdot \lceil \log_2 n \rceil + 1}}{2} = 2^{k \cdot \lceil \log_2 n \rceil}$ paths accept. Since $2^{k \cdot \lceil \log_2 n \rceil} - (t-1)$ paths accept for $b_0 = 1$, this means that it accepts iff more than $2^{k \cdot \lceil \log_2 n \rceil} - (2^{k \cdot \lceil \log_2 n \rceil} - (t-1)) = t-1$ assignments of weight k are satisfying.

<u>Hardness</u>: In order to show hardness, we directly reduce an arbitrary language in W[P]-PFPT to p-WCIRCThreshSAT$_=$(BF) by using a slight modification of the proof for W[P]-hardness of p-WCIRCSAT$_=(BF)$ given in [FG]. Let $L \in$ W[P]-PFPT be some parameterized problem.
There is a k-restricted NTM M that w. l. o. g. on any input (x,k) uses exactly $f(k) \cdot \lfloor \log_2 |x| \rfloor$ nondeterministic bits on all computation paths. By Lemma 3.6, we can compute for any input (x,k) a circuit $C_{(x,k)}$ with $f(k) \cdot \lfloor \log_2 |x| \rfloor$ input bits in fpt-runtime s. t. $\#\mathrm{Sat}(C_{(x,k)}) = \#\mathrm{acc}_M(x,k)$, so we have

$$\begin{aligned} (x,k) \in L \Leftrightarrow &\quad \#\mathrm{acc}_M(x,k) \geq 2^{f(k) \cdot \lfloor \log_2 |x| \rfloor - 1} + 1 && (1) \\ \Leftrightarrow &\quad \#\mathrm{Sat}(C_{(x,k)}) \geq 2^{f(k) \cdot \lfloor \log_2 |x| \rfloor - 1} + 1. && (2) \end{aligned}$$

Now we change $C_{(x,k)}$ into a new circuit $D_{(x,k)}$ in a way that the number of satisfying assignments for $C_{(x,k)}$ is exactly the number of satisfying assignments of weight $f(k)$ for $D_{(x,k)}$. This is done with the so-called k-$\log n$-trick:
Let $n := |x|$. We split the input bits into blocks of $\lfloor \log_2 n \rfloor$ bits. The $\lfloor \log_2 n \rfloor$ bits of each block can be interpreted as the position of a single 1 in a block of $2^{\lfloor \log_2 n \rfloor}$ bits and vice versa. More precisely, we can use blocks of $2^{\lfloor \log_2 n \rfloor}$ bits as our input and use circuits to calculate the original $\lfloor \log_2 n \rfloor$ bit blocks from them. To make this work, we have to make sure that exactly one bit per block is set to 1 for every satisfying weight-$f(k)$-assignment of the new circuit. We now describe in detail how $D_{(x,k)}$ can be constructed.
Let the input nodes of $C_{(x,k)}$ be $\{u_{ij} \mid 1 \leq i \leq f(k), 1 \leq j \leq \lfloor \log_2 |x| \rfloor\}$ and the new input nodes of $D_{(x,k)}$ be $\{v_{ij} \mid 1 \leq i \leq f(k), 0 \leq j \leq 2^{\lfloor \log_2 |x| \rfloor} - 1\}$ (for both sets think of the first index as the block and the second index the position within that block).

To make sure that one bit per block is set to 1, for all $i \in \{1, \ldots, f(k)\}$ we add the circuit

$$S_i := \bigvee_{j=1}^{2^{\lfloor \log_2 |x| \rfloor}} v_{ij} \qquad (3)$$

and conjunct the nodes computing those disjunctions with the output node of $C_{(x,k)}$ in order to get the new output node of $D_{(x,k)}$. This suffices because now for all satisfying assignments at least one bit per block must be set to 1, but under this condition no weight-$f(k)$ assignment can have more than a single 1 in any block.
Next we need circuits computing the values for the input nodes of $C_{(x,k)}$ from the input nodes of $D_{(x,k)}$. The value of node u_{ij} can be computed as

$$U_{i,j} := \bigvee_{\substack{\ell \in \{0, \ldots, 2^{\lfloor \log_2 |x| \rfloor} - 1\}, \\ \mathrm{bit}(j,\ell) = 1}} v_{i\ell}, \qquad (4)$$

where $\mathrm{bit}(i,j)$ is the value of the i-th bit in the binary representation of j.
Altogether, we obtain $D_{(x,k)}$ by replacing the input nodes of $C_{(x,k)}$ with the gates computing the

respective formula from Equation 4 and conjuncting the whole circuit with the formulae in Equation 3 as depicted in the following figure:

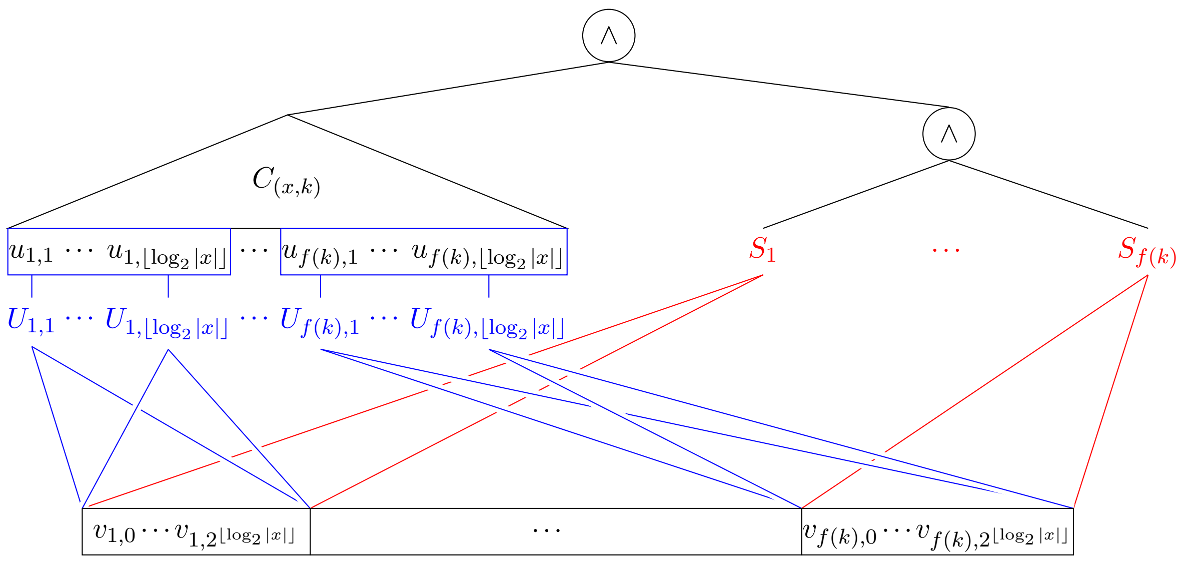

$\wedge$
$\wedge$
$C_{(x,k)}$
$u_{1,1} \cdots u_{1,\lfloor \log_2 |x| \rfloor}$
$\cdots$
$u_{f(k),1} \cdots u_{f(k),\lfloor \log_2 |x| \rfloor}$
S_1
$\cdots$
$S_{f(k)}$
$U_{1,1} \cdots U_{1,\lfloor \log_2 |x| \rfloor} \cdots U_{f(k),1} \cdots U_{f(k),\lfloor \log_2 |x| \rfloor}$
$v_{1,0} \cdots v_{1,2^{\lfloor \log_2 |x| \rfloor}}$
$\cdots$
$v_{f(k),0} \cdots v_{f(k),2^{\lfloor \log_2 |x| \rfloor}}$

Figure 1: Circuit $D_{(x,k)}$

This transformation can be done in fpt-runtime. Now we have

$$\begin{aligned}(x,k) \in L \Leftrightarrow \quad & \#\mathrm{Sat}(C_{(x,k)}) \geq 2^{f(k)\cdot\lfloor \log_2 |x| \rfloor - 1} + 1 && (5)\\ \Leftrightarrow \quad & \#\mathrm{Sat}_{f(k)}(D_{(x,k)}) \geq 2^{f(k)\cdot\lfloor \log_2 |x| \rfloor - 1} + 1. && (6)\end{aligned}$$

Hence, $(x,k) \mapsto (D_{(x,k)}, 2^{f(k)\cdot\lfloor \log_2 |x| \rfloor - 1} + 1, f(k))$ is an fpt-reduction from L to p-WCIRCThreshSAT$_=$(BF). □

Remark 3.8. *This proof could probably be changed to work for a parameterized version of a weighted MajoritySAT. However, after using the k-$\log n$-trick, the number of satisfying assignments is not equal to half of all assignments of the needed weight. The circuit would have to be adjusted to correct this.*

We want to add the completeness result for circuits over the base $\{\wedge, \vee, 0, 1\}$ at this point, because the proof is only a slight modification of the proof of Theorem 3.7.

Lemma 3.9. p-WCIRCThreshSAT$_=(\{\wedge, \vee, 0, 1\})$ *is* W[P]-PFPT-*complete.*

Proof. Membership follows directly from Theorem 3.7. For hardness, we modify the hardness proof of Theorem 3.7 as done by Flum and Grohe. We will only explain the part that needs to be changed: Since all circuits we add during the proof are already monotone, the only problem is that the circuit $C_{(x,k)}$ contains negations. To deal with this, we first bring $C_{(x,k)}$ in negation normal form (NNF), meaning that we now only have negations directly in front of variables. Then we add a new set of inputs u'_{ij} that serve the role of the negated inputs:
Wherever the negated input nodes occured before, we replace them with the corresponding u'_{ij}. This way we obtain a new circuit $C'_{(x,k)}$ with $2n$ input nodes (n being the number of inputs of $C_{(x,k)}$). If we make sure that those new input nodes are always set to the negated value of the corresponding old input node, this circuit behaves just as $C_{(x,k)}$.
We now compute $D'_{(x,k)}$ from $C'_{(x,k)}$ in the same way we computed $D_{(x,k)}$ from $C_{(x,k)}$ before, but

produce the values of u'_{ij} for $1 \leq i \leq f(k), 1 \leq j \leq \lfloor \log_2 |x| \rfloor$ from the new inputs as well, which is done for u'_{ij} by the monotone circuit

$$\bigvee_{\substack{\ell \in \{0,\ldots,2^{\lfloor \log_2 |x| \rfloor}-1\}, \\ \mathrm{bit}(j,\ell)=0}} v_{i\ell}.$$

Now we have

$$\begin{aligned}(x,k) \in L \Leftrightarrow \quad & \#\mathrm{Sat}(C_{(x,k)}) \geq 2^{f(k)\cdot\lfloor \log_2 |x| \rfloor -1} + 1 \\ \Leftrightarrow \quad & C'_{(x,k)} \text{ has } \geq 2^{f(k)\cdot\lfloor \log_2 |x| \rfloor -1} + 1 \text{ satisfying assignments with } u'_{ij} = \neg u_{ij} \ \forall i,j \\ \Leftrightarrow \quad & \#\mathrm{Sat}_{f(k)}(D'_{(x,k)}) \geq 2^{f(k)\cdot\lfloor \log_2 |x| \rfloor -1} + 1\end{aligned}$$

and $D'_{(x,k)}$ is a circuit over $\{\wedge, \vee, 0, 1\}$. □

3.2.2 Fragments of p-WCIRCThreshSAT$_=$(BF) and p-WThreshSAT$_=$(BF)

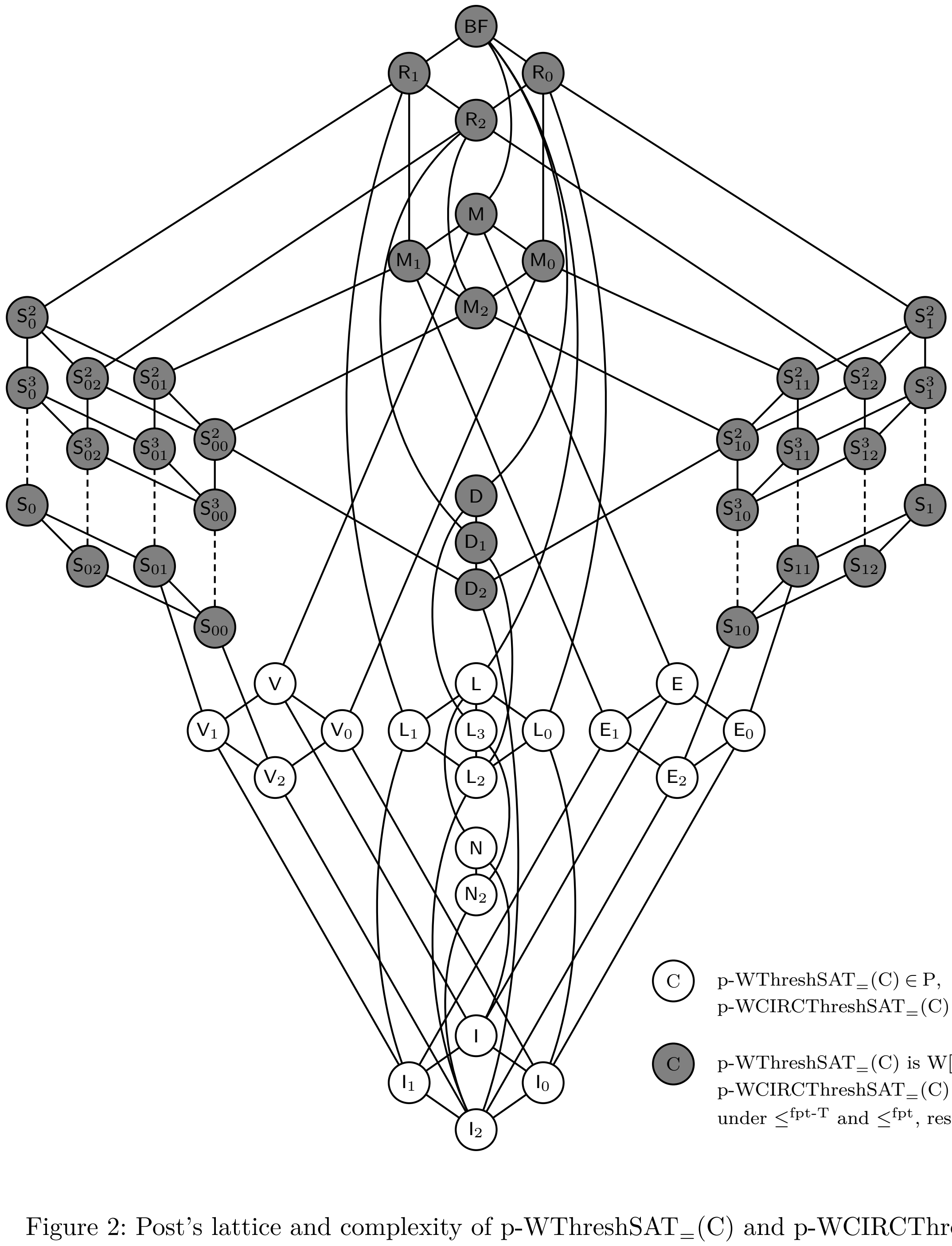
We have shown that the parameterized threshold-satisfiability problem over the class of all circuits and over the class of all monotone circuits is W[P]-PFPT-complete. The same problem over the class of all formulae is W[ThreshSAT]-complete by definition. Thus we can begin to classify both problems over fragments from Post's lattice (see Figure 2 and Table 1)—in Lemma 3.9 we already started with this. Since most of the reductions given for the counting-versions in [CV] are either parsimonious (the number of satisfying assignments is preserved) or Turing-1 reductions (Turing-reductions using only one oracle query), we can use them here. We can even use the Turing-1 reductions as common fpt-reductions because the threshold is part of the instance for our problem. There is one exception to this, namely completness of the clone M_2 for formulae, but that can be handled with a slight modification.

In the following, let B be a base of Boolean functions.
We begin by showing membership in W[ThreshSAT] and W[P]-PFPT, respectively, for the parameterized weighted threshold-satisfiability problem over arbitrary bases of Boolean functions given as formulae and circuits, respectively.

Lemma 3.10.

p-WCIRCThreshSAT$_=(B) \in$ W[P]-PFPT *and* p-WThreshSAT$_=(B) \in$ W[ThreshSAT].

Proof. p-WCIRCThreshSAT$_=(B) \in$ W[P]-PFPT can be shown completely analogously to membership in the proof of Theorem 3.7.
p-WThreshSAT$_=(B) \leq^{\mathrm{fpt}}$ p-WThreshSAT$_=$(BF) via the identy as the reduction function shows p-WThreshSAT$_=(B) \in$ W[ThreshSAT]. □



○ C: p-WThreshSAT$_=$(C) $\in$ P, p-WCIRCThreshSAT$_=$(C) $\in$ P

● C: p-WThreshSAT$_=$(C) is W[ThreshSAT]-complete, p-WCIRCThreshSAT$_=$(C) is W[P]-complete under $\leq^{\text{fpt-T}}$ and $\leq^{\text{fpt}}$, respectively

Figure 2: Post's lattice and complexity of p-WThreshSAT$_=$(C) and p-WCIRCThreshSAT$_=$(C)

Class	Definition	Base	
BF	All Boolean functions	$\{x \wedge y, \neg x\}$	
R_0	$\{f \mid f$ is $\bot$-reproducing$\}$	$\{x \wedge y, x \oplus y\}$	
R_1	$\{f \mid f$ is $\top$-reproducing$\}$	$\{x \vee y, x \leftrightarrow y\}$	
R_2	$\mathsf{R}_0 \cap \mathsf{R}_1$	$\{x \vee y, x \wedge (y \leftrightarrow z)\}$	
M	$\{f \mid f$ is monotone$\}$	$\{x \vee y, x \wedge y, \bot, \top\}$	
M_0	$\mathsf{M} \cap \mathsf{R}_0$	$\{x \vee y, x \wedge y, \bot\}$	
M_1	$\mathsf{M} \cap \mathsf{R}_1$	$\{x \vee y, x \wedge y, \top\}$	
M_2	$\mathsf{M} \cap \mathsf{R}_2$	$\{x \vee y, x \wedge y\}$	
S_0	$\{f \mid f$ is $\bot$-separating$\}$	$\{x \rightarrow y\}$	
S_1	$\{f \mid f$ is $\top$-separating$\}$	$\{x \nrightarrow y\}$	
S_0^n	$\{f \mid f$ is $\bot$-separating of degree $n\}$	$\{x \rightarrow y, T_2^{n+1}\}$	
S_1^n	$\{f \mid f$ is $\top$-separating of degree $n\}$	$\{x \nrightarrow y, T_n^{n+1}\}$	
S_{00}	$\mathsf{S}_0 \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \vee (y \wedge z)\}$	
S_{00}^n	$\mathsf{S}_0^n \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \vee (y \wedge z), \mathrm{T}_2^3\}$	if $n = 2$,
		$\{\mathrm{T}_2^{n+1}\}$	if $n \geq 3$
S_{01}	$\mathsf{S}_0 \cap \mathsf{M}$	$\{x \vee (y \wedge z), \top\}$	
S_{01}^n	$\mathsf{S}_0^n \cap \mathsf{M}$	$\{T_2^{n+1}, \top\}$	
S_{02}	$\mathsf{S}_0 \cap \mathsf{R}_2$	$\{x \vee (y \nrightarrow z)\}$	
S_{02}^n	$\mathsf{S}_0^n \cap \mathsf{R}_2$	$\{x \vee (y \nrightarrow z), T_2^{n+1}\}$	
S_{10}	$\mathsf{S}_1 \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \wedge (y \vee z)\}$	
S_{10}^n	$\mathsf{S}_1^n \cap \mathsf{R}_2 \cap \mathsf{M}$	$\{x \wedge (y \vee z), \mathrm{T}_2^3\}$	if $n = 2$,
		$\{\mathrm{T}_n^{n+1}\}$	if $n \geq 3$
S_{11}	$\mathsf{S}_1 \cap \mathsf{M}$	$\{x \wedge (y \vee z), \bot\}$	
S_{11}^n	$\mathsf{S}_1^n \cap \mathsf{M}$	$\{T_n^{n+1}, \bot\}$	
S_{12}	$\mathsf{S}_1 \cap \mathsf{R}_2$	$\{x \wedge (y \rightarrow z)\}$	
S_{12}^n	$\mathsf{S}_1^n \cap \mathsf{R}_2$	$\{x \wedge (y \rightarrow z), T_n^{n+1}\}$	
D	$\{f \mid f$ is self-dual$\}$	$\{\mathrm{maj}\{x, \overline{y}, \overline{z}\}\}$	
D_1	$\mathsf{D} \cap \mathsf{R}_2$	$\{\mathrm{maj}\{x, y, \overline{z}\}\}$	
D_2	$\mathsf{D} \cap \mathsf{M}$	$\{\mathrm{maj}\{x, y, z\}\}$	
L	$\{f \mid f$ is linear$\}$	$\{x \oplus y, \top\}$	
L_0	$\mathsf{L} \cap \mathsf{R}_0$	$\{x \oplus y\}$	
L_1	$\mathsf{L} \cap \mathsf{R}_1$	$\{x \leftrightarrow y\}$	
L_2	$\mathsf{L} \cap \mathsf{R}_2$	$\{x \oplus y \oplus z\}$	
L_3	$\mathsf{L} \cap \mathsf{D}$	$\{x \oplus y \oplus z \oplus \top\}$	
V	$\{f \mid f$ is a disjunction or constant$\}$	$\{x \vee y, \bot, \top\}$	
V_0	$\mathsf{M}_0 \cap \mathsf{V}$	$\{x \vee y, \bot\}$	
V_1	$\mathsf{M}_1 \cap \mathsf{V}$	$\{x \vee y, \top\}$	
V_2	$\mathsf{M}_2 \cap \mathsf{V}$	$\{x \vee y\}$	
E	$\{f \mid f$ is a conjunction or constant$\}$	$\{x \wedge y, \bot, \top\}$	
E_0	$\mathsf{M}_0 \cap \mathsf{E}$	$\{x \wedge y, \bot\}$	
E_1	$\mathsf{M}_1 \cap \mathsf{E}$	$\{x \wedge y, \top\}$	
E_2	$\mathsf{M}_2 \cap \mathsf{E}$	$\{x \wedge y\}$	
N	$\{f \mid f$ depends on at most one variable$\}$	$\{\neg x, \bot, \top\}$	
N_2	$\mathsf{L}_3 \cap \mathsf{N}$	$\{\neg x\}$	
I	$\{f \mid f$ is a projection or a constant$\}$	$\{\mathrm{id}, \bot, \top\}$	
I_0	$\mathsf{R}_0 \cap \mathsf{I}$	$\{\mathrm{id}, \bot\}$	
I_1	$\mathsf{R}_1 \cap \mathsf{I}$	$\{\mathrm{id}, \top\}$	
I_2	$\mathsf{R}_2 \cap \mathsf{I}$	$\{\mathrm{id}\}$	

Table 1: List of all Boolean clones with their bases

We continue by proving two helpful lemmata that allow us to get rid of constants from a base of Boolean functions in certain cases for both formulae and circuits. This will be needed for the hardness proofs.

Lemma 3.11. *If $\wedge \in [B]$, then*

$$\text{p-WCIRCThreshSAT}_=(B \cup \{1\}) \leq^{\text{fpt}} \text{p-WCIRCThreshSAT}_=(B) \textit{ and}$$
$$\text{p-WThreshSAT}_=(B \cup \{1\}) \leq^{\text{fpt}} \text{p-WThreshSAT}_=(B).$$

Proof. We give the proof for formulae; the result for circuits can be shown analogously.
Let φ be an arbitrary formula over $B \cup \{1\}$. Then let $\varphi' := \varphi[1/y] \wedge y$, where y is a new variable. The number of satisfying weight-$k+1$-assignments of φ' is exactly the number of satisfying weight-k-assignments of φ. Furthermore, φ' can be expressed by a formula over B because $\wedge \in [B]$. Thus, $(\varphi, t, k) \mapsto (\varphi, t, k+1)$ is an fpt-reduction from p-WThreshSAT$_=(B \cup \{1\})$ to p-WThreshSAT$_=(B)$. □

Lemma 3.12. *If $\vee \in [B]$, then*

$$\text{p-WCIRCThreshSAT}_=(B \cup \{0\}) \leq^{\text{fpt}} \text{p-WCIRCThreshSAT}_=(B) \textit{ and}$$
$$\text{p-WThreshSAT}_=(B \cup \{0\}) \leq^{\text{fpt}} \text{p-WThreshSAT}_=(B).$$

Proof. We give the proof for formulae, the result for circuits can be shown analogously.
Let φ be an arbitrary formula over $B \cup \{0\}$ and $n := |\text{Var}(\varphi)|$. Then let $\varphi' := \varphi[0/y] \vee y$, where y is a new variable. The number of satisfying weight-k-assignments of φ' is exactly the number of satisfying weight-k-assignments of φ (for the case $y = 0$) plus the number of possibilities to set $k-1$ out of n variables to 1 (for the case $y = 1$), so we have $\#\text{Sat}_k(\varphi') = \#\text{Sat}_k(\varphi) + \binom{n}{k-1}$. Furthermore, φ' can be expressed by a formula over B because $\vee \in [B]$. Thus, $(\varphi, t, k) \mapsto (\varphi', t + \binom{n}{k-1}, k)$ is an fpt-reduction from p-WThreshSAT$_=(B \cup \{0\})$ to p-WThreshSAT$_=(B)$. □

We are now able to show hardness for the different clones from Post's lattice.

Lemma 3.13. *If $M_2 \subseteq [B]$, then* p-WThreshSAT$_=(B)$ *is* W[ThreshSAT]*-complete under* $\leq^{\text{fpt-T}}$ *and* p-WCIRCThreshSAT$_=(B)$ *is* W[P]-PFPT*-complete under* $\leq^{\text{fpt}}$*.*

Proof. <u>p-WThreshSAT$_=(B)$</u>: Let B' be any base of Boolean functions with $[B'] = \text{BF}$. Then p-WThreshSAT$_=(B') \leq^{\text{fpt-T}}$ p-WThreshSAT$_=(\{\wedge, \vee, \neg\})$ (this follows from Corollary 4.8 in [Tho]). For hardness, we show p-WThreshSAT$_=(\{\wedge, \vee, \neg\}) \leq^{\text{fpt-T}}$ p-WThreshSAT$_=(B)$.
We start by proving p-WThreshSAT$_=(\{\wedge, \vee, \neg\}) \leq^{\text{fpt-T}}$ p-WThreshSAT$_=(\{\wedge, \vee\})$. The idea for the proof is the same as for the weighted #SAT-problem examined in [CV]: We bring the given formula φ into the form

$$\varphi \equiv \alpha \wedge \neg\beta,$$

where α and β are monotone formulae (an explanation of how this can be done follows). Then it holds that $\#\text{Sat}_k(\varphi) = \#\text{Sat}_k(\alpha) - \#\text{Sat}_k(\alpha \wedge \beta)$ as long as both α and β use all variables. The problem is that we cannot directly get the answer whether $\#\text{Sat}_k(\varphi) \geq t$ for some threshold t from only two oracle-calls for α and β as it is done for the weighted #SAT-problem in [CV]. Instead, we will use binary search with oracle-calls to compute $\#\text{Sat}_k(\alpha \wedge \beta)$ and then use another oracle-call for α with a modified threshold to get the answer (modifying the threshold spares us another binary search).

We can bring φ in the desired form as follows: Let $n := |\mathrm{Var}(\varphi)|$ and $x_1, \ldots, x_n$ be the variables in φ. First we bring φ in negation normal form (NNF) and call the new formula φ'. Then we create n new variables $x'_1, \ldots x'_n$ and replace every occurance of $\neg x_i$ in φ' with x'_i for all $1 \leq i \leq n$. We call the new formula φ''. Now we only have to make sure that $x_i = \neg x'_i$ for all $1 \leq i \leq n$. This is done by conjuncting the formula with

$$\bigwedge_{i=1}^{n} (x_i \vee x'_i) \wedge \neg \bigvee_{i=1}^{n} (x_i \wedge x'_i).$$

Therefore, in the notation from above we have:

$$\alpha := \varphi'' \wedge \bigwedge_{i=1}^{n} (x_i \vee x'_i) \quad \text{and}$$

$$\beta := \bigvee_{i=1}^{n} (x_i \wedge x'_i).$$

We now give the fpt-Turing reduction as an algorithm using oracle-calls to p-WThreshSAT$_=(\{\wedge, \vee\})$.

```
Input: Formula $\varphi(x_1, \ldots, x_n)$ over $\{\wedge, \vee, \neg\}$, $t \in \mathbb{N}$, $k \in \mathbb{N}$
Let $\varphi'$ be $\varphi$ transformed to NNF
Let $\varphi''$ be $\varphi'$, where $\neg x_i$ is replaced by a new variable $x'_i$ $\forall 1 \leq i \leq n$
$\alpha \leftarrow \varphi'' \wedge \bigwedge_{i=1}^{n} (x_i \vee x'_i)$
$\beta \leftarrow \bigvee_{i=1}^{n} (x_i \wedge x'_i)$
$l \leftarrow 0$
$h \leftarrow \binom{n}{k}$
while $l < h$ do
    $m \leftarrow \lceil \frac{l+h}{2} \rceil$
    if $(\alpha \wedge \beta, m, k) \in$ p-WThreshSAT$_=(\{\wedge, \vee\})$ then
        $l \leftarrow m$
    else
        $h \leftarrow m - 1$
    end if
end while
if $(\alpha, t + l, k)$ then
    accept
else
    reject
end if
```

Note that this algorithm is an fpt-Turing reduction with fpt-many oracle calls, which is an extremely powerful form of reduction.
To conclude the proof, we now have to show p-WThreshSAT$_=(\{\wedge, \vee\}) \leq^{\text{fpt-T}}$ p-WThreshSAT$_=(B)$. Because $M \subseteq [B \cup \{0, 1\}]$, there are short $(B \cup \{0, 1\})$-representations of $\wedge$ and $\vee$ as shown in [Schn] so we get p-WThreshSAT$_=(\{\wedge, \vee\}) \leq^{\text{fpt-T}}$ p-WThreshSAT$_=(B \cup \{0, 1\})$. Now we can use Lemma 3.11 and Lemma 3.12 to get rid of the constants, since $\wedge, \vee \in \mathrm{M}_2 \subseteq [B]$.

p-WCIRCThreshSAT$_=(B)$: p-WCIRCThreshSAT$_=(\{\wedge, \vee, 0, 1\})$ is W[P]-PFPT-complete by Lemma 3.9. From that point on we can reduce analogously to the formula-version:
We get p-WCIRCThreshSAT$_=(\{\wedge, \vee, 0, 1\}) \leq^{\text{fpt}}$ p-WCIRCThreshSAT$_=(B \cup \{0, 1\})$ because there are short $B \cup \{0, 1\}$-representations of $\wedge$ and $\vee$. We can then apply Lemma 3.11 and Lemma 3.12 to get rid of the constants, since $\wedge, \vee \in \mathrm{M}_2 \subseteq [B]$. $\square$

Lemma 3.14. *If* $S_{10} \subseteq [B]$, *then* p-WThreshSAT$_=(B)$ *is* W[ThreshSAT]-*complete under* $\leq^{\text{fpt-T}}$ *and* p-WCIRCThreshSAT$_=(B)$ *is* W[P]-PFPT-*complete under* $\leq^{\text{fpt}}$.

Proof. As seen in Post's lattice, $\mathrm{M}_1 \subseteq [B \cup \{1\}]$, so p-WThreshSAT$_=(B \cup \{1\})$ is W[ThreshSAT]-complete under $\leq^{\text{fpt-T}}$ by Lemma 3.13. p-WThreshSAT$_=(B \cup \{1\}) \leq^{\text{fpt-T}}$ p-WThreshSAT$_=(B)$ follows by Lemma 3.11, since $\wedge \in \mathrm{S}_{10} \subseteq [B]$.
W[P]-PFPT-completeness under $\leq^{\text{fpt}}$ for p-WCIRCThreshSAT$_=(B)$ can be shown analogously - the difference here is that Lemma 3.13 provides completeness under $\leq^{\text{fpt}}$ instead of $\leq^{\text{fpt-T}}$. □

Lemma 3.15. *If* $S_{00} \subseteq [B]$, *then* p-WThreshSAT$_=(B)$ *is* W[ThreshSAT]-*complete under* $\leq^{\text{fpt-T}}$ *and* p-WCIRCThreshSAT$_=(B)$ *is* W[P]-PFPT-*complete under* $\leq^{\text{fpt}}$.

Proof. As seen in Post's lattice, $\mathrm{M}_0 \subseteq [B \cup \{0\}]$, so p-WThreshSAT$_=(B \cup \{0\})$ is W[ThreshSAT]-complete under $\leq^{\text{fpt-T}}$ by Lemma 3.13. p-WThreshSAT$_=(B \cup \{0\}) \leq^{\text{fpt-T}}$ p-WThreshSAT$_=(B)$ follows by Lemma 3.12, since $\vee \in \mathrm{S}_{00} \subseteq [B]$.
W[P]-PFPT-completeness under $\leq^{\text{fpt}}$ for p-WCIRCThreshSAT$_=(B)$ can be shown analogously - the difference again is that Lemma 3.13 provides completeness under $\leq^{\text{fpt}}$ instead of $\leq^{\text{fpt-T}}$. □

Lemma 3.16. *If* $D_2 \subseteq [B]$, *then* p-WThreshSAT$_=(B)$ *is* W[ThreshSAT]-*complete under* $\leq^{\text{fpt-T}}$ *and* p-WCIRCThreshSAT$_=(B)$ *is* W[P]-PFPT-*complete under* $\leq^{\text{fpt}}$.

Proof. As seen in Post's lattice $\mathrm{S}_{01}^2 \subseteq [B \cup \{1\}]$. Thus, p-WThreshSAT$_=(B \cup \{1\})$ is W[ThresSAT]-complete under $\leq^{\text{fpt-T}}$ by Lemma 3.15. By giving an fpt-reduction instead of an fpt-Turing reduction, we show p-WThreshSAT$_=(B \cup \{1\}) \leq^{\text{fpt-T}}$ p-WThreshSAT$_=(B)$, so the proof can be used for both formulae and circuits. This can be done using the following functions g_l, which are defined differently depending on the last input bit:

$$g_l(x_1, \ldots, x_l, 0) := \bigwedge_{i=1}^{l} x_i, \qquad g_l(x_1, \ldots, x_l, 1) := \bigvee_{i=1}^{l} x_i$$

These functions are self-dual due to the different definition depending on the last bit and De Morgans law, therefore $g_l \in \mathrm{D}_2$. Furthermore we need the ternary majority function, denoted here as maj. This function is self-dual as well and thus maj $\in \mathrm{D}_2$.
On input (φ, k, t) the reduction computes the new formula

$$\varphi' := \mathrm{maj}(\varphi[1/z],\ z,\ g_{k+2}(y_1, \ldots, y_{k+2}, z))$$

with new variables z and y_i for $1 \leq i \leq k+2$.
We now consider $\#\mathrm{Sat}_{k+1}(\varphi')$. For any satisfying assignment of φ', the following holds:

1. If $z = 0$: Both $\varphi[1/z]$ and $g_{k+2}(y_1, \ldots, y_{k+2}, z)$ are true. Then by definition of g_{k+2}, y_i is true for all $1 \leq i \leq k+2$ and this means the weight of the assignment must be $\geq k+2$.
2. If $z = 1$: If $g_{k+2}(y_1, \ldots, y_{k+2}, z)$ is true: Then it is irrelevant whether $\varphi \equiv \varphi[1/z]$ is satisfied and the only further condition is that the assignment has weight $k+1$.
   If $g_{k+2}(y_1, \ldots, y_{k+2}, z)$ is false: Then y_i is false for all $1 \leq i \leq k+2$ and therefore the assignment is a satisfying weight-$k+1$-assignment of φ' iff its restriction to $\mathrm{Var}(\varphi)$ is a satisfying weight k-assignment of φ.

This means that the satisfying weight-$k+1$-assignments of φ' are the union of

- the satisfying weight-k-assignments of φ extended by $z=1$ and $y_i=0$ for all i
- all assignments setting $z=1$, $y_i=1$ for at least one i and the needed number of variables in $\mathrm{Var}(\varphi)$ to 1 (in order to get a weight-$k+1$-assignment).

Hence,

$$\#\mathrm{Sat}_{k+1}(\varphi') = \#\mathrm{Sat}_k(\varphi) + \sum_{j=1}^{k} \binom{k+2}{j}\binom{n}{k-j}.$$

Therefore, $(\varphi,t,k) \mapsto (\varphi', t+\sum_{j=1}^{k}\binom{k+2}{j}\binom{n}{k-j}, k+1)$ is the desired fpt-reduction (note that the size of g_{k+2} only depends on the parameter).
W[P]-PFPT-completeness under $\leq^{\mathrm{fpt}}$ for p-WCIRCThreshSAT$_=$(B) can be shown analogously. □

Lemma 3.17. *If $B \subseteq V$, or $B \subseteq E$, or $B \subseteq L$:*

$$\text{p-WThreshSAT}_=(B), \text{p-WCIRCThreshSAT}_=(B) \in \mathrm{P}.$$

Proof. For these cases, FP-membership has been shown for the counting-versions p-#WSAT$_=(B)$ and p-#WCIRCSAT$_=(B)$. Hence, we can compute the number of satisfying assignments and then accept depending on whether that number is greater or equal the threshold with a polynomial-time algorithm. □

We can now combine the results we have shown into the following theorem, classifying the weighted threshold-satisfiability over all fragments from Post's lattice for both circuits and formulae:

Theorem 3.18. *Let B be a finite set of Boolean functions.*

1. *If $D_2 \subseteq [B]$ or $S_{10} \subseteq [B]$ or $S_{00} \subseteq [B]$, then* p-WThreshSAT$_=(B)$ *is* W[ThreshSAT]*-complete under* $\leq^{\text{fpt-T}}$ *and* p-WCIRCThreshSAT$_=(B)$ *is* W[P]-PFPT*-complete under* $\leq^{\mathrm{fpt}}$.
2. *In all other cases,* p-WThreshSAT$_=(B)$, p-WCIRCThreshSAT$_=(B) \in \mathrm{P}$.

4 The Boolean Algebra over W[P]

In classical complexity theory, a famous proof by Beigel, Reingold and Spielman shows that PP is closed under intersection. Combined with the fact that it is closed under complementation, it is easy to see that arbitrary Boolean combinations of languages in PP are also in PP. Before closure under intersection was shown, it was already known, that arbitrary Boolean combinations of languages in NP are in PP—note that NP $\subseteq$ PP is easily shown. We now want to follow the lead of those proofs and show that arbitrary Boolean combinations of languages in W[P] are contained in W[P]-PFPT.

4.1 Definitions

For the following definitions, let $\mathcal{K}$ be a complexity class.

Definition 4.1. The class of complements of languages in $\mathcal{K}$ is

$$\mathrm{co}\mathcal{K} := \{L \mid \overline{L} \in \mathcal{K}\}.$$

Definition 4.2. $\mathrm{BA}(\mathcal{K})$ is the smallest class that contains $\mathcal{K}$ and is closed under union, intersection and complementation. $\mathrm{BA}(\mathcal{K})$ is called the Boolean algebra over $\mathcal{K}$.

The Boolean operators $\wedge$ and $\vee$ on complexity classes are defined as follows:

Definition 4.3. Let $\mathcal{K}_1$ be another complexity class. Then

$$\mathcal{K} \wedge \mathcal{K}_1 = \{A \cap B \mid A \in \mathcal{K}, B \in \mathcal{K}_1\} \quad \text{and}$$

$$\mathcal{K} \vee \mathcal{K}_1 = \{A \cup B \mid A \in \mathcal{K}, B \in \mathcal{K}_1\}.$$

Definition 4.4. For $n \in \mathbb{N} \setminus \{0\}$ we define the classes of the Boolean hierarchy over $\mathcal{K}$ as

$$\mathrm{C}^{\mathcal{K}}_{2n-1} := \mathrm{co}\mathcal{K} \vee \bigvee_{n-1} (\mathcal{K} \wedge \mathrm{co}\mathcal{K}), \quad \mathrm{C}^{\mathcal{K}}_{2n} := \bigvee_{n} (\mathcal{K} \wedge \mathrm{co}\mathcal{K}),$$

$$\mathrm{D}^{\mathcal{K}}_{2n-1} := \mathcal{K} \vee \bigvee_{n-1} (\mathcal{K} \wedge \mathrm{co}\mathcal{K}), \qquad \mathrm{D}^{\mathcal{C}}_{2n} := \mathcal{K} \vee \mathrm{co}\mathcal{K} \vee \bigvee_{n-1} (\mathcal{K} \wedge \mathrm{co}\mathcal{K}).$$

Definition 4.5. We define the classes of the difference hierarchy over $\mathcal{K}$ as

$$\mathrm{DIFF}^{\mathcal{K}}_1 := \mathcal{K} \quad \text{and}$$

$$\mathrm{DIFF}^{\mathcal{K}}_{n+1} := \{A \triangle B \mid A \in \mathrm{DIFF}^{\mathcal{K}}_n, B \in \mathcal{K}\} \text{ for } n \in \mathbb{N} \setminus \{0\}.$$

Definition 4.6. Let f be a k-ary Boolean function. Then we define the class $f(\mathcal{K})$ as follows:

$$A \in f(\mathcal{K}) \Leftrightarrow \exists B_1, \ldots, B_k \in \mathcal{K} : c_A(x) = f(c_{B_1}(x), \ldots, c_{B_k}(x)) \; \forall x.$$

Beside these basic definitions we need a few more technical expressions concerning the so-called mind change technique, which will be used for the main result of this section.

Definition 4.7. 1. $\sqsubseteq$ is the initial word relation, a binary relation on words defined as

$$u \sqsubseteq w :\Leftrightarrow \exists v : uv = w.$$

2. $\leq$, a partial order on bit-strings of length k is defined as

$$(a_1,\dots,a_k) \leq (b_1,\dots,b_k) :\Leftrightarrow a_1 \leq b_1,\dots,a_k \leq b_k.$$

3. For any k-ary Boolean function f we define

$$c(f) := max \sqsubseteq \left\{ f(a_1)f(a_2)\dots f(a_r) \middle| \begin{matrix} a_i \in \{0,1\}^k, f(a_i) \neq f(a_{i+1}), \\ r \geq 1, \quad a_1 \leq a_2 \leq \dots \leq a_r \end{matrix} \right\}.$$

4. For any k-ary Boolean function f

$\mathrm{mc}(f) := |c(f)| - 1$ is the number of mind changes of f.

4.2 Preliminaries

We will now recall the known results [KSW] that allow to conclude $\mathrm{BA(NP)} \subseteq \mathrm{PP}$ without using the fact that PP is closed under intersection. Then, we will see that W[P] and W[P]-PFPT fulfill all required preconditions and deduce the corresponding result in parameterized complexity theory.

Theorem 4.8. *Let $\mathcal{K}$ be a class of languages that is closed under union and intersection. Then for every k-ary Boolean function f it holds:*

$$f(\mathcal{K}) = \begin{cases} \mathrm{C}^{\mathcal{K}}_{mc(f)}, & \text{if } c(f) \text{ ends with } 0 \\ \mathrm{D}^{\mathcal{K}}_{mc(f)}, & \text{if } c(f) \text{ ends with } 1 \end{cases}.$$

Theorem 4.9. *Let $\mathcal{K}$ be a class of languages that is closed under union and intersection. For every $n \geq 1$ it holds:*

(i) $\mathrm{C}^{\mathcal{K}}_{n+1} = \mathrm{D}^{\mathcal{K}}_{n} \wedge co\mathcal{K}$

(ii) $\mathrm{D}^{\mathcal{K}}_{n+1} = \mathrm{C}^{\mathcal{K}}_{n} \vee \mathcal{K}$

(iii) $\mathrm{C}^{\mathcal{K}}_{n} = co\mathrm{D}^{\mathcal{K}}_{n}$

(iv) $\mathrm{C}^{\mathcal{K}}_{n} \cup \mathrm{D}^{\mathcal{K}}_{n} \subseteq \mathrm{C}^{\mathcal{K}}_{n+1} \cap \mathrm{D}^{\mathcal{K}}_{n+1}$

(v) $\mathrm{BA}(\mathcal{K}) = \bigcup_{n\geq 1} \mathrm{C}^{\mathcal{K}}_{n} = \bigcup_{n\geq 1} \mathrm{D}^{\mathcal{K}}_{n}$

The following theorem has only been stated for the difference hierarchy over NP, but works identically for arbitrary classes. We will address this in Lemma 4.11.

Theorem 4.10. *For every $n \geq 1$ it holds that:*

(i) $\mathrm{DIFF}^{\mathrm{NP}}_{2n-1} = \mathrm{D}^{\mathrm{NP}}_{2n-1}$

(ii) $\mathrm{DIFF}^{\mathrm{NP}}_{2n} = \mathrm{C}^{\mathrm{NP}}_{2n}$

Using the fact that PP is closed under symmetric difference—which was known before closure of PP under intersection—one can directly conclude $\mathrm{BA(NP)} \subseteq \mathrm{PP}$.

4.3 BA(W[P]) $\subseteq$ W[P]-PFPT

We will now show Theorem 4.10 for arbitrary classes that are closed under union and intersection, which will then allow us to prove the main result of this chapter: Theorem 4.14.

Lemma 4.11. *Let $\mathcal{K}$ be a class of languages that is closed under union and intersection. For every $n \geq 1$ it holds:*

(i) $\mathrm{DIFF}^{\mathcal{K}}_{2n-1} = \mathrm{D}^{\mathcal{K}}_{2n-1}$ *and*

(ii) $\mathrm{DIFF}^{\mathcal{K}}_{2n} = \mathrm{C}^{\mathcal{K}}_{2n}$

Proof. Let $d_n(x_1,\dots,x_n) := x_1 \oplus \cdots \oplus x_n$. Then $\mathrm{DIFF}^{\mathcal{K}}_n = d_n(\mathcal{K})$. We have $|c(d_n)| \leq n+1$ since d_n is n-ary. It is easy to show that $|c(d_n)| \geq n+1$ holds as well, because $\oplus$ changes its value with any change of a variable:

$$d_n(0,0,\dots,0)d_n(1,0,0,\dots,0)d_n(1,1,0,0,\dots,0)\dots d_n(1,1,\dots,1) = \underbrace{0101\dots b}_{n+1},$$

where $b = 0$ if n is even and $b = 1$ if n is odd. Furthermore,

$$(0,0,\dots,0) \leq (1,0,0,\dots,0) \leq (1,1,0,0,\dots,0) \leq \cdots \leq (1,1,\dots,1).$$

Now we have $|c(d_n)| = n+1$ and $\mathrm{mc}(d_n) = n$. Using Theorem 4.8, we obtain:

$$d_n(\mathcal{K}) = \begin{cases} \mathrm{D}^{\mathcal{K}}_n, & \text{if } n \text{ is odd} \\ \mathrm{C}^{\mathcal{K}}_n, & \text{if } n \text{ is even} \end{cases} .$$

□

Corollary 4.12. *Let $\mathcal{K}$ be a class of languages that is closed under union and intersection. Then:*

$$\mathrm{BA}(\mathcal{K}) = \bigcup_{n\geq 1} \mathrm{DIFF}^{\mathcal{K}}_n .$$

We need the following basic closure properties of W[P] to use the previous results:

Lemma 4.13. W[P] *is closed under union and intersection.*

Proof. Let $L_1, L_2 \in$ W[P] via M_1 and M_2, respectively. For $L_1 \cup L_2$ ($L_1 \cap L_2$), M_1 and M_2 can be simulated in parallel (in succession). □

We are now in the position to prove the main result of this section:

Theorem 4.14. BA(W[P]) $\subseteq$ W[P]-PFPT*, that is, the Boolean algebra over* W[P] *is contained in* W[P]-PFPT*.*

Proof. Since W[P]-PFPT is closed under symmetrice difference (Lemma 2.16), $\mathrm{DIFF}^{\mathrm{W[P]\text{-}PFPT}}_n \subseteq$ W[P]-PFPT for any n. Additionally, W[P] $\subseteq$ W[P]-PFPT (Lemma 2.17) and thus $\mathrm{DIFF}^{\mathrm{W[P]}}_n \subseteq \mathrm{DIFF}^{\mathrm{W[P]\text{-}PFPT}}_n$ for any n. W[P] is closed under union and intersection (Lemma 4.13), so we can use Corollary 4.12 to get

$$\mathrm{BA}(\mathrm{W[P]}) = \bigcup_{n\geq 1} \mathrm{DIFF}^{\mathrm{W[P]}}_n \subseteq \bigcup_{n\geq 1} \mathrm{DIFF}^{\mathrm{W[P]\text{-}PFPT}}_n \subseteq \mathrm{W[P]\text{-}PFPT}.$$

□

5 The Class W[P]-$\mathrm{C}_=$FPT

In classical complexity theory the class $\mathrm{C}_=$P, which is defined quite similarly to PP, is examined as well. We want to define the corresponding parameterized class W[P]-⊕FPT and examine its basic properties, namely closure under Boolean operations, how it is connected to W[P]-PFPT and the existance of a complete problem for the class.

Definition 5.1. A parameterized problem L is in W[P]-$\mathrm{C}_=$FPT if there is a function $f \in$ Gap-FPT such that $(x,k) \in L \Leftrightarrow f(x,k) = 0$.

We begin by showing the closure of W[P]-$\mathrm{C}_=$FPT under union and intersection. The proof that shows closure of $\mathrm{C}_=\mathrm{NC}^1$ under those operations [CKTV] can be used here as well.

Theorem 5.2. W[P]-$\mathrm{C}_=$FPT *is closed under union and intersection.*

Proof. Let $L_1, L_2 \in$ W[P]-$\mathrm{C}_=$FPT. Then there are two functions $f_1, f_2 \in$ Gap-FPT such that

$$(x,k) \in L_i \Leftrightarrow f_i(x,k) = 0, \quad i \in \{1,2\}.$$

<u>Union</u>: Let $g_1(x,k) = f_1(x,k) \cdot f_2(x,k)$. Then $g_1 \in$ Gap-FPT by Lemma 2.10 and

$$g_1(x,k) = 0 \text{ iff } (f_1(x,k) = 0 \text{ or } f_2(x,k) = 0).$$

Therefore we have $(x,k) \in L_1 \cup L_2 \Leftrightarrow g_1(x,k) = 0$ and thus $L_1 \cup L_2 \in$ W[P]-$\mathrm{C}_=$FPT.

<u>Intersection</u>: Let $g_2(x,k) = f_1(x,k)^2 + f_2(x,k)^2$. Then $g_2 \in$ Gap-FPT by Lemma 2.10 and

$$g_2(x,k) = 0 \text{ iff } (f_1(x,k) = 0 \text{ and } f_2(x,k) = 0).$$

Therefore we have $(x,k) \in L_1 \cap L_2 \Leftrightarrow g_2(x,k) = 0$ and thus $L_1 \cap L_2 \in$ W[P]-$\mathrm{C}_=$FPT. □

Next, we want to show how W[P]-PFPT and W[P]-$\mathrm{C}_=$FPT are related. For this, we need the following characterization of the class W[P]-$\mathrm{C}_=$FPT (the idea for this stems from [CKTV] again).

Lemma 5.3. *Let L be a parameterized problem. Then L is in* W[P]-$\mathrm{C}_=$FPT *iff there are $f_1, f_2 \in$* #W[P] *such that:*

$$\begin{aligned} (x,k) \in L \Rightarrow &\quad f_1(x,k) - f_2(x,k) = 0 \quad \textit{and} \\ (x,k) \notin L \Rightarrow &\quad f_1(x,k) - f_2(x,k) < 0. \end{aligned}$$

Proof. Let $L \in$ W[P]-$\mathrm{C}_=$FPT be a parameterized problem. By Lemma 2.11, there are $f, g \in$ #W[P] such that

$$(x,k) \in L \Leftrightarrow f(x,k) - g(x,k) = 0.$$

By squaring this difference we make sure that it is always non-negative. Reordering combined with the closure properties of #W[P] yields:

$$\begin{aligned} (f(x,k) - g(x,k))^2 = &\quad f(x,k)^2 - 2f(x,k)g(x,k) + g(x,k)^2 \\ = &\quad \underbrace{f(x,k)^2 + g(x,k)^2}_{\in \#\mathrm{W[P]}} - \underbrace{2f(x,k)g(x,k)}_{\in \#\mathrm{W[P]}}. \end{aligned}$$

This uses closure of #W[P] under p-bounded summations and products (shown in Lemma 2.8). Now let $f_1(x,k) = 2f(x,k)g(x,k)$, $f_2(x,k) = f(x,k)^2 + g(x,k)^2$. Then we have:

$$\begin{aligned}(x,k) \in L \Rightarrow \quad & f(x,k) - g(x,k) = 0 \\ \Rightarrow \quad & f_1(x,k) - f_2(x,k) = 0\end{aligned}$$

and

$$\begin{aligned}(x,k) \notin L \Rightarrow \quad & f(x,k) - g(x,k) \neq 0 \\ \Rightarrow \quad & f_2(x,k) - f_1(x,k) > 0 \\ \Rightarrow \quad & f_1(x,k) - f_2(x,k) < 0.\end{aligned}$$

□

Theorem 5.4. W[P]-$\mathrm{C}_=$FPT $\subseteq$ W[P]-PFPT

Proof. By Theorem 2.14, a parameterized problem L is in W[P]-PFPT iff there is a function $f \in$ Gap-FPT s. t.: $(x,k) \in L \Leftrightarrow f(x,k) > 0$. Thus, using Lemma 5.3, W[P]-$\mathrm{C}_=$FPT is just a restriction of W[P]-PFPT: Let L be a parameterized problem, $L \in$ W[P]-$\mathrm{C}_=$FPT. By Lemma 5.3 there are $f, g \in$ #W[P] such that:

$$\begin{aligned}(x,k) \in L \Rightarrow \quad & f(x,k) - g(x,k) = 0 \quad \Rightarrow \quad f(x,k) - g(x,k) \geq 0 \quad \text{and} \\ (x,k) \notin L \quad & \qquad\qquad\qquad\qquad\;\; \Rightarrow \quad f(x,k) - g(x,k) < 0\end{aligned}$$

Now $h(x,k) := f(x,k) - g(x,k) \in$ Gap-FPT by Lemma 2.11 and we have

$$\begin{aligned}(x,k) \in L \Leftrightarrow \quad & h(x,k) \geq 0 \\ \Leftrightarrow \quad & \underbrace{h(x,k) + 1}_{\in \text{Gap-FPT}} > 0\end{aligned}$$

and hence $L \in$ W[P]-PFPT. □

6 The Class W[P]-$\oplus$FPT

Another counting class examined in classical complexity theory is the class $\oplus$P—the class of all problems decidable by a nondeterministic Turing machine that accepts if an odd number of computation paths accept. We will now go on to define the parameterized version of this class and examine its basic properties: We will show that like $\oplus$P, the parameterized version is closed under all Boolean operations and define the satisfiability problem p-WCIRCOddSAT$_=$(BF), which is complete for said class.

Definition 6.1. A parameterized problem L is in the class W[P]-$\oplus$FPT if there is a k-restricted NTM M such that

$$(x,k) \in L \Leftrightarrow \#\text{acc}_M(x,k) \equiv 1 \mod 2.$$

Remark 6.2. *We get an equivalent definition if we use an even instead of an odd number of accepting paths.*

Proof. By adding one accepting path to all computations, one can change a machine from accepting a language with an odd number of accepting paths to accepting the same language with an even number of accepting paths and vice versa. ☐

We know from classical complexity theory that $\mathrm{P}^{\oplus\mathrm{P}} = \oplus\mathrm{P}$. Although we cannot expect to achieve an analogous result in parameterized complexity theory due to the logarithmic factor in the number of nondeterministic bits, at least the following restricted result holds and allows us to show the closure properties of W[P]-$\oplus$FPT under Boolean operations.

Lemma 6.3. $\mathrm{W[P]}\text{-}\oplus\mathrm{FPT} = \mathrm{FPT}^{\mathrm{W[P]}\text{-}\oplus\mathrm{FPT}[f(k),\kappa]}$.

Proof. We reproduce the proof for $\mathrm{P}^{\oplus\mathrm{P}} = \oplus\mathrm{P}$ from classical complexity theory (see e.g. [Vol]). As the other direction is trivial, we only have to show $\mathrm{FPT}^{\mathrm{W[P]}\text{-}\oplus\mathrm{FPT}[f(k),\kappa]} \subseteq \mathrm{W[P]}\text{-}\oplus\mathrm{FPT}$.
Let L be a parameterized problem and $L \in \mathrm{FPT}^{\mathrm{W[P]}\text{-}\oplus\mathrm{FPT}[f(k),\kappa]}$ via the oracle Turing machine M_1 with oracle $A \in$ W[P]-$\oplus$FPT. This means that $(x,k) \in L \Leftrightarrow M_1^A(x,k)$ accepts. Let M_2 be the k-restricted nondeterministic Turing machine that accepts A in parity-fashion.
W. l. o. g. we can assume that for every input all computation paths of M_2 use the same number of nondeterministic bits and then add one rejecting path so that the number of computation paths of M_2 is always odd. This means

$$\begin{aligned}(x,k) \in L \Rightarrow \quad & \#\text{acc}_{M_2}(x,k) \equiv 1 \mod 2 \quad \text{and} \quad \#\text{rej}_{M_2}(x,k) \equiv 0 \mod 2 \quad \text{and}\\ (x,k) \notin L \Rightarrow \quad & \#\text{acc}_{M_2}(x,k) \equiv 0 \mod 2 \quad \text{and} \quad \#\text{rej}_{M_2}(x,k) \equiv 1 \mod 2.\end{aligned}$$

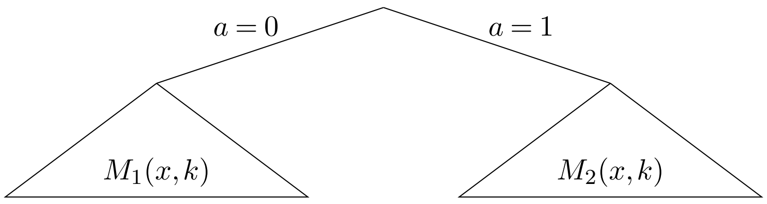
Now we define the nondeterministic Turing machine M as follows: M behaves like M_1, but whenever there is an oracle call to A, the whole computation of M_2 is copied. The computation then continues on all paths assuming that the query is in fact in A iff that particular computation path of M_2 accepts.
Now for every copy of M_2 within the computation of M, the number of paths assuming the right answer is odd and the number of paths assuming the wrong answer is even. Since for the acceptance behavior in parity-fashion the number of accepting paths is only counted modulo 2, the paths assuming the right answers contribute in the same way as if there was only one such path and the paths assuming the wrong answer do not contribute at all. In short, M accepts L in parity-fashion.
To conclude the proof, we need to consider the running time as well as the used nondeterministic bits

of M: By restricting the parameter values of all oracle queries such that they must be computable from the original parameter, we ensure that all copies of M_2 within M have fpt-runtime and k-restricted nondeterminism (cf. proof for closure of W[P] under $\leq^{\text{fpt}}$ in [FG]) w. r. t. the original input. By restricting the number of oracle queries by $f(k)$ for a computable function f, we ensure that we get only a sum of $f(k)$-many fpt-runtimes and k-restricted nondeterminism, which remains within the required bounds. $\square$

Before we show closure of W[P]-$\oplus$FPT under all Boolean operations, we need another small lemma:

Lemma 6.4. W[P]-$\oplus$FPT *is closed under marked union.*

Proof. Let $A, B \in$ W[P]-$\oplus$FPT. There are k-restricted NTMs M_1, M_2 that accept A and B, respectively, in parity-fashion. A k-restricted NTM on input (ax, k) can simply simulate one of the machines M_1 and M_2 on (x, k) depending on a:



$\square$

Theorem 6.5. W[P]-$\oplus$FPT *is closed under union, intersection and complementation.*

Proof. Let A, B be two parameterized problems, $A, B \in$ W[P]-$\oplus$FPT.
We show that $\overline{A}, A \cup B, A \cap B \in \text{FPT}^{\text{W[P]-}\oplus\text{FPT}[f(k),\kappa]}$. An oracle Turing machine using A as its oracle can simply query the original input and reverse the answer to decide $\overline{A}$.
In order to decide $A \cup B$ and $A \cap B$ we use $A \uplus B$ as the oracle. On input (x, k) the machine then queries $(0x, k)$ and $(1x, k)$ to the oracle and then accepts either if at least one of the oracle answers is positive (for $A \cup B$) or if both oracle answers are positive (for $A \cap B$).
The described machine uses a constant number of queries and for all of them, the parameter is equal to the original parameter, so it is in fact in $\text{FPT}^{\text{W[P]-}\oplus\text{FPT}[f(k),\kappa]}$ and by Lemma 6.3 in W[P]-$\oplus$FPT. $\square$

We will now go on to define a satisfiability-problem that will turn out to be W[P]-$\oplus$FPT-complete under $\leq^{\text{fpt}}$.

Definition 6.6. The parameterized weighted odd-satisfiability-problem over a class of circuits $\mathcal{C}$ is defined as follows:

Problem:	p-WCIRCOddSAT$_=(\mathcal{C})$
Input:	Circuit $C \in \mathcal{C}$, $k \in \mathbb{N}$
Question:	Does C have an odd number of satisfying assignments of weight exactly k?
Parameter:	k

Theorem 6.7. p-WCIRCOddSAT$_=$(BF) *and* p-WCIRCOddSAT$_=$($\{\wedge, \vee, 0, 1\}$) *are* W[P]-$\oplus$FPT-*complete.*

Proof. Membership: When given a circuit C, a k-restricted NTM can guess a weight-k-assignment by guessing the indices of the variables set to 1 and then verify whether that assignment satisfies the circuit.
Hardness: The proof for W[P]-$\oplus$FPT-hardness of p-WCIRCOddSAT$_=$(BF) is completely analogous to the hardness proof of Theorem 3.7.The only difference is that the machine does not accept iff more than half of its computation paths accept, but iff an odd number of its computation paths accept. Therefore instead of Equivalences 1 and 2 (on page 10) we get

$$\begin{aligned}(x,k) \in L \Leftrightarrow \quad & \#\mathrm{acc}_M(x,k) \equiv 1 \mod 2\\ \Leftrightarrow \quad & \#\mathrm{Sat}(C_{(x,k)}) \equiv 1 \mod 2\end{aligned}$$

and in the end instead of Equivalences 5 and 6 (on page 11) we get

$$\begin{aligned}(x,k) \in L \Leftrightarrow \quad & \#\mathrm{Sat}(C_{(x,k)}) \equiv 1 \mod 2\\ \Leftrightarrow \quad & \#\mathrm{Sat}_{f(k)}(D_{(x,k)}) \equiv 1 \mod 2.\end{aligned}$$

For p-WCIRCOddSAT$_=$($\{\wedge, \vee, 0, 1\}$), the same modifications as in the proof for Lemma 3.9 work again. □

7 The Parameterized Counting Hierarchy

In classical complexity theory the polynomial hierarchy is defined, generalizing the concept of the classes NP and coNP. This hierarchy was extended by a counting notion to obtain the counting hierarchy. In order to define the classes in the polynomial and counting hierarchies, quantifiers are used. We want to define a parameterized counting quantifier and show some basic results for the hierarchy defined using only that single quantifier. Parameterized existential and universal quantifiers could be defined analogously, providing the means to define a hierarchy corresponding to the whole counting hierarchy from classical complexity theory. In this whole chapter we follow the lead of [Tor].

7.1 Definitions

We begin by defining the counting quantifiers we mentioned above.

Definition 7.1. Let Σ be an alphabet, $f \in \mathrm{FFPT}$, $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and Q a ternary predicate. Furthermore, $g(|x|,k) \le h(k) \cdot \log|x|$ f. a. x,k for some computable function h. Then the k-restricted counting quantifier $\mathrm{C}[\kappa]$ is defined as follows:

$$\mathrm{C}[\kappa]^g_f y : Q(x,y,k) :\Leftrightarrow |\{y \mid |y| \le g(|x|,k) \text{ and } Q(x,y,k)\}| \ge f(x,k)$$

Beside the k-restricted counting quantifier we want to define the k-restricted exact counting quantifier, which will help to prove Theorem 7.16.

Definition 7.2. Let Σ be an alphabet, $f \in \mathrm{FFPT}$, $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and Q a ternary predicate. Furthermore, $g(|x|,k) \le h(k) \cdot \log|x|$ f. a. x,k for some computable function h. Then the k-restricted exact counting quantifier $\mathrm{C}\!\!=\![\kappa]$ is defined as follows:

$$\mathrm{C}\!\!=\![\kappa]^g_f y : Q(x,y,k) :\Leftrightarrow |\{y \mid |y| \le g(|x|,k) \text{ and } Q(x,y,k)\}| = f(x,k)$$

Remark 7.3. *We can assume that all strings y of length exactly $g(|x|,k)$ instead of $\le g(|x|,k)$ are counted.*

Proof. This can be done by adding a new symbol to the alphabet over which y is built and use it as padding for the shorter words. □

The counting quantifiers are mainly used to define complexity classes:

Definition 7.4. Let $Q \in \{\mathrm{C}[\kappa], \mathrm{C}\!\!=\![\kappa]\}$ be a k-restricted quantifier. For any class of languages $\mathcal{K}$, let $Q\mathcal{K}$ be defined as follows:
A language A is in $Q\mathcal{K}$ if there are $f \in \mathrm{FFPT}$ with $f(x,k) > 0$ f. a. x,k, a computable function h, a function $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ with $g(|x|,k) \le h(k) \cdot \log|x|$ $\forall x,k$ and a language $B \in \mathcal{K}$ s. t. f. a. (x,k):

$$(x,k) \in A \Leftrightarrow Q^g_f y : (\langle x,y\rangle,k) \in B.$$

Definition 7.5. We call the hierarchy that is defined using any number of $\mathrm{C}[\kappa]$-quantifiers on FPT the k-restricted counting hierarchy. $\mathrm{CH}[\kappa]$ is the union of all classes in that hierarchy:

$$\mathrm{CH}[\kappa] := \bigcup_{i \in \mathbb{N}} \underbrace{\mathrm{C}[\kappa] \ldots \mathrm{C}[\kappa]}_{i} \mathrm{FPT}.$$

7.2 Preliminaries

We will show a few auxiliary results, which will be needed for the following proofs and start by showing a few simple closure properties for the classes in $\mathrm{CH}[\kappa]$.

Lemma 7.6. *Let $\mathcal{K} \in \mathrm{CH}[\kappa]$ be a complexity class, $L_1, L_2 \in \mathcal{K}$ and $A \in \mathrm{FPT}$. Then*

(i) $\mathcal{K}$ is closed under $\leq^{\mathrm{fpt}}$,

(ii) $L_1 \uplus L_2 \in \mathcal{K}$,

(iii) $L_1 \cap A \in \mathcal{K}$ and

(iv) $L_1 \cup A \in \mathcal{K}$

Proof. Let q be the length of the quantifier prefix of $\mathcal{K}$.

<u>(i)</u>: Let $B \in \mathrm{FPT}$, $f_1, \dots, f_q \in \mathrm{FFPT}$ and $g_1, \dots, g_q$ be functions that are bounded by $h(k) \cdot \log|x|$ for a computable function h s. t.

$$(x,k) \in L_1 \Leftrightarrow \mathrm{C}[\kappa]_{f_1}^{g_1}\ y_1 \dots \mathrm{C}[\kappa]_{f_q}^{g_q}\ y_q : (\langle x, y_1, \dots, y_q \rangle, k) \in B$$

Let L be a parameterized language that is fpt-reducible to L_1, r an fpt-reduction that shows $L \leq^{\mathrm{fpt}} L_1$ and r_1, r_2 its two components such that $r(x,k) = (r_1(x,k), r_2(x,k))$. Then for every input (x,k) we have

$$\begin{aligned}(x,k) \in L \Leftrightarrow \quad & (r_1(x,k), r_2(x,k)) \in L_1 \\ \Leftrightarrow \quad & \mathrm{C}[\kappa]_{f_1'}^{g_1'}\ y_1 \dots \mathrm{C}[\kappa]_{f_q'}^{g_q'}\ y_q : (\langle r_1(x,k), y_1, \dots, y_q \rangle, r_2(x,k)) \in B \\ \Leftrightarrow \quad & \mathrm{C}[\kappa]_{f_1'}^{g_1'}\ y_1 \dots \mathrm{C}[\kappa]_{f_q'}^{g_q'}\ y_q : (\langle x, y_1, \dots, y_q \rangle, k) \in B',\end{aligned}$$

where $B' = \{(\langle x, y_1, \dots, y_q \rangle, k) \mid (\langle r_1(x,k), y_1, \dots, y_q \rangle, r_2(x,k)) \in B\}$.
The functions f_i and g_i have to be changed here, since the input is still (x,k). The new functions are defined as

$$\begin{aligned} f_i'(x,k) := \quad & f_i(r_1(x,k), r_2(x,k)) \quad \text{and} \\ g_i'(x,k) := \quad & g_i(r_1(x,k), r_2(x,k)). \end{aligned}$$

$g_i'(x,k)$ is still bounded as needed (this can be shown analogously to closure of W[P] under $\leq^{\mathrm{fpt}}$) and $f_i' \in \mathrm{FFPT}$ (this can be shown analogously to closure of FPT under $\leq^{\mathrm{fpt}}$). Moreover $B' \in \mathrm{FPT}$, which follows from closure of FPT under $\leq^{\mathrm{fpt}}$. Closure of W[P] and FPT under $\leq^{\mathrm{fpt}}$ were shown in [FG] for example.

<u>(ii)</u>: Let $f_1, \dots, f_q \in \mathrm{FFPT}$ and for $i \in \{1,2\}$ let $L_i' \in \mathrm{FPT}$ and $g_{i,1}, \dots, g_{i,q}$ be functions that are bounded by $h(k) \cdot \log|x|$ for a computable function h such that

$$(x,k) \in L_i \Leftrightarrow \mathrm{C}[\kappa]_{f_1}^{g_{i,1}}\ y_1 \dots \mathrm{C}[\kappa]_{f_q}^{g_{i,q}}\ y_q : (\langle x, y_1, \dots, y_q \rangle, k) \in L_i' \quad \text{for } i \in \{1,2\}.$$

Note that w. l. o. g. we can assume that for both languages the same functions f_i are used when we allow the g_i to be different (later we will see that even both functions could be assumed to be identical, see Lemma 7.8). Let

$$L' := \left\{ (\langle ax, y_1, \dots, y_q \rangle, k) \,\middle|\, \begin{array}{c} (a = 0, |y_i| \leq g_{1,i}(\langle x, y_1, \dots, y_{i-1} \rangle)\ \forall i \text{ and } (\langle x, y_1, \dots, y_q \rangle, k) \in L_1') \\ \text{or } (a = 1, |y_i| \leq g_{2,i}(\langle x, y_1, \dots, y_{i-1} \rangle)\ \forall i \text{ and } (\langle x, y_1, \dots, y_q \rangle, k) \in L_2') \end{array} \right\}.$$

L' is obviously in FPT and now it holds that

$$(ax,k) \in L_1 \uplus L_2 \Leftrightarrow \mathrm{C}[\kappa]_{f_1}^{\max(g_{1,1},g_{2,1})}\ y_1 \ldots \mathrm{C}[\kappa]_{f_q}^{\max(g_{1,q},g_{2,q})}\ y_q : (\langle ax, y_1, \ldots, y_q\rangle, k) \in L'.$$

<u>(iii)</u>: Let $B \in \mathrm{FPT}$, $f_1, \ldots, f_q \in \mathrm{FFPT}$ and $g_1, \ldots, g_q$ be functions that are bounded by $h(k) \cdot \log|x|$ for a computable function h such that

$$(x,k) \in L_1 \Leftrightarrow \mathrm{C}[\kappa]_{f_1}^{g_1}\ y_1 \ldots \mathrm{C}[\kappa]_{f_q}^{g_q}\ y_q : (\langle x, y_1, \ldots, y_q\rangle, k) \in B.$$

Then it holds that

$$(x,k) \in L_1 \cap A \Leftrightarrow \mathrm{C}[\kappa]_{f_1}^{g_1}\ y_1 \ldots \mathrm{C}[\kappa]_{f_q}^{g_q}\ y_q : (\langle x, y_1, \ldots, y_q\rangle, k) \in B',$$

where $B' := \{(\langle x, y_1, \ldots, y_q\rangle, k) \mid (\langle x, y_1, \ldots, y_q\rangle, k) \in B \text{ and } (x,k) \in A\} \in \mathrm{FPT}$.

<u>(iv)</u>: This can be shown analogously to (iii). □

Lemma 7.7. *Let $\mathcal{K} \in \mathrm{CH}[\kappa]$. All properties from Lemma 7.6 except for (iv) also hold for $\mathrm{C}_{=}[\kappa]\mathcal{K}$.*

Proof. This can be shown completely analogously to Lemma 7.6. The problem for (iv) is that we would need to change the FPT-language in the end of the quantifier characterization s. t. f. a. $(x,k) \in A$ the exact number of witnesses specified by the $\mathrm{C}_{=}[\kappa]$-quantifier exists. □

We will now show that when the $\mathrm{C}[\kappa]$-quantifier is used on classes from $\mathrm{CH}[\kappa]$, the function f from the definition can be changed to always be half of the strings y from the definition and that it is directly connected to the class W[P]-PFPT.

Lemma 7.8. *Let $\mathcal{K} \in \mathrm{CH}[\kappa]$ and f, g be functions as in the definitions above with the addition that $g \in \mathrm{FFPT}$. Then*

(i) $\mathrm{C}[\kappa]_f^g\mathcal{K} \subseteq \mathrm{C}[\kappa]_{2^{g(|x|,k)}+1}^{g+1}\mathcal{K}$ *and*

(ii) $\mathrm{C}[\kappa]\mathrm{FPT} = \mathrm{W[P]\text{-}PFPT}$.

Proof. <u>(i)</u>: Let $L \in \mathrm{C}[\kappa]_f^g\mathcal{K}$ be a parameterized problem. Then there is a $B \in \mathcal{K}$ s. t.

$$(x,k) \in L \Leftrightarrow \mathrm{C}[\kappa]_f^g y : (\langle x,y\rangle, k) \in B.$$

Let B' be the following language:

$$\begin{aligned} B' := \quad & \{(\langle x, ay\rangle, k) \mid a = 0, (\langle x,y\rangle, k) \in B\} \\ \cup\ & \{(\langle x, ay\rangle, k) \mid a = 1,\ y < 2^{g(|x|,k)} - f(x,k) + 1\}. \end{aligned}$$

Since $f, g \in \mathrm{FFPT}$ and the classes in $\mathrm{CH}[\kappa]$ are closed under union with FPT-languages and under $\leq^{\mathrm{fpt}}$—the latter allows us to add the bit a to the input—, we have $B' \in \mathcal{K}$ and it holds that

$$\begin{aligned} (x,k) \in L \quad & \Leftrightarrow \quad \mathrm{C}[\kappa]_f^g y : (\langle x,y\rangle, k) \in B \\ & \Leftrightarrow \quad |\{y \mid |y| \leq g(|x|,k) \text{ and } (\langle x,y\rangle, k) \in B\}| \geq f(x,k) \\ & \Leftrightarrow \quad |\{ay | a \in \{0,1\}, |y| \leq g(|x|,k) \text{ and } (\langle x, ay\rangle, k) \in B'\}| \geq 2^{g(|x|,k)} + 1 \\ & \Leftrightarrow \quad \mathrm{C}[\kappa]_{2^{g(|x|,k)}+1}^{g+1} y : (\langle x,y\rangle, k) \in B'. \end{aligned}$$

(ii): For $\mathrm{C}[\kappa]\mathrm{FPT} \subseteq \mathrm{W[P]\text{-}PFPT}$, let $L \in \mathrm{C}[\kappa]\mathrm{FPT}$ be a parameterized problem. We first use Remark 7.3 so that we only have to take care of strings of length exactly $g(|x|,k)$ and then (i) to shift the threshold to be more than half of the strings y. There are $B \in \mathrm{FPT}$, $f \in \mathrm{FFPT}$ and a function g that is bounded by $h(k) \cdot \log|x|$ for some computable function h s. t.

$$(x,k) \in L \Leftrightarrow \mathrm{C}[\kappa]^{g}_{(2^{g(|x|,k)}/2)+1} y : (\langle x,y\rangle,k) \in B.$$

Note that we can w. l. o. g. assume $g \in \mathrm{FFPT}$. Now on input (x,k), a k-restricted NTM M can guess a binary string of length $g(|x|,k)$ and accept iff $(\langle x,y\rangle,k) \in B$. The number of accepting paths will then be the number of y with $|y| = g(|x|,k)$ where $(\langle x,y\rangle,k) \in B$. Since $(x,k) \in L$ iff more than half of those y fulfill $(\langle x,y\rangle,k) \in B$, we get $L \in \mathrm{W[P]\text{-}PFPT}$ via M.
For $\mathrm{W[P]\text{-}PFPT} \subseteq \mathrm{C}[\kappa]\mathrm{FPT}$, let $L \in \mathrm{W[P]\text{-}PFPT}$ be a parameterized problem. Then there is a k-restricted NTM M that decides L in PP-fashion. W. l. o. g. we can assume that all computation paths of M on input (x,k) use exactly $f(k) \cdot \lfloor \log_2 |x| \rfloor$ nondeterministic bits where f is some computable function. Let B be the path language of M, that is,

$$B := \{(\langle x,y\rangle,k) \mid y \text{ encodes an accepting computation path of } M \text{ on input } (x,k)\}.$$

Now we have

$$\begin{aligned}(x,k) \in L \Leftrightarrow \quad & \#\mathrm{acc}_M(x,k) \geq 2^{f(k)\cdot\lfloor \log_2 |x| \rfloor - 1} + 1 \\ \Leftrightarrow \quad & |\{y \mid (\langle x,y\rangle,k) \in B\}| \geq 2^{f(k)\cdot\lfloor \log_2 |x| \rfloor - 1} + 1 \\ \Leftrightarrow \quad & \mathrm{C}[\kappa]^{f(k)\cdot\lfloor \log_2 |x| \rfloor}_{2^{f(k)\cdot\lfloor \log_2 |x| \rfloor - 1}+1} y : (\langle x,y\rangle,k) \in B.\end{aligned}$$

□

Lemma 7.9. *Let $\mathcal{K} \in \mathrm{CH}[\kappa]$, $L \in \mathcal{K}$ and Σ be an alphabet. Then $L \times_\kappa (\Sigma^* \times \mathbb{N})$ and $(\Sigma^* \times \mathbb{N}) \times_\kappa L$ are also in $\mathcal{K}$.*

Proof. Both languages only increase the input size without changing the complexity of the language. We still want to show the result formally: Let q be the length of the quantifier prefix of $\mathcal{K}$ and let $B \in \mathrm{FPT}$, $f_1, \dots, f_q \in \mathrm{FFPT}$, and $g_1, \dots, g_q$ be functions that are bounded by $h(k) \cdot \log|x|$ for a computable function h such that

$$(x,k) \in L \Leftrightarrow \mathrm{C}[\kappa]^{g_1}_{f_1}\, y_1 \dots \mathrm{C}[\kappa]^{g_q}_{f_q}\, y_q : (\langle x, y_1, \dots, y_q\rangle, k) \in B.$$

We define a new parameterized problem B':

$$\begin{aligned}B' := \quad & B \times_\kappa (\Sigma^* \times \mathbb{N}) \\ = \quad & \{(\langle x_1, y_1, \dots, y_q\rangle, x_2, k) \mid (\langle x_1, y_1, \dots, y_q\rangle, k) \in B\}.\end{aligned}$$

Now we have $B' \in \mathrm{FPT}$ and

$$(x_1, x_2, k) \in L \times_\kappa (\Sigma^* \times \mathbb{N}) \Leftrightarrow \mathrm{C}[\kappa]^{g_1}_{f_1}\, y_1 \dots \mathrm{C}[\kappa]^{g_q}_{f_q}\, y_q : (\langle x_1, y_1, \dots, y_q\rangle, x_2, k) \in B'$$

and therefore $(L \times_\kappa (\Sigma^* \times \mathbb{N})) \in \mathcal{K}$. $((\Sigma^* \times \mathbb{N}) \times_\kappa L) \in \mathcal{K}$ can be shown analogously. □

7.3 Closure under Boolean Operations

We will now go on to show the expected closure properties of the classes in $\mathrm{CH}[\kappa]$: Like W[P]-PFPT, all classes in the hierarchy are closed under complementation and symmetric difference.

Theorem 7.10. *Let $\mathcal{K} \in \mathrm{CH}[\kappa]$. Then $\mathcal{K}$ is closed under complementation and symmetric difference.*

Proof. We show both results by induction over the length of the quantifier prefix of $\mathcal{K}$. Let q be this length.
Complement: Induction basis: For $q = 0$, we have $\mathcal{K} = \mathrm{FPT}$, which is closed under complementation as it is a deterministic class.
Induction step: $q \to q+1$. Let $\mathcal{K}' = \mathrm{C}[\kappa]\mathcal{K}$. We need to show that $\mathcal{K}'$ is closed under complementation. Let $L \in \mathcal{K}'$. There are $B \in \mathcal{K}$, a computable function h and $f, g \in \mathrm{FFPT}$ with $g(|x|,k) \leq h(k) \cdot \log|x|$ ($g \in \mathrm{FFPT}$ is assumed w. l. o. g.) s. t. f. a. (x,k):

$$(x,k) \in L \Leftrightarrow \mathrm{C}[\kappa]^g_f y : |y| = g(|x|,k), (\langle x,y\rangle,k) \in B.$$

Now we have

$$\begin{aligned}(x,k) \in \overline{L} \Leftrightarrow \quad & \neg(\mathrm{C}[\kappa]^g_f y : |y| = g(|x|,k), (\langle x,y\rangle,k) \in B) \\ \Leftrightarrow \quad & \mathrm{C}[\kappa]^g_{2^{g(|x|,k)}-f(x,k)} y : |y| = g(|x|,k), \underbrace{(\langle x,y\rangle,k) \notin B}_{(\langle x,y\rangle,k)\in\overline{B}}\end{aligned}$$

and by induction hypothesis, $\overline{B} \in \mathcal{K}$. Hence, $\overline{L} \in \mathrm{C}[\kappa]\mathcal{K} = \mathcal{K}'$.
Symmetric difference: Induction basis: For $q = 0$, we have $\mathcal{K} = \mathrm{FPT}$, which is closed under symmetric difference since the sum of fpt-runtimes is still an fpt-runtime and therefore two fpt-machines can be simulated successively.
Induction step: $q \to q+1$. Let $\mathcal{K}' = \mathrm{C}[\kappa]K$. We need to show that $\mathcal{K}'$ is closed under symmetric difference. Let $L_1, L_2 \in \mathcal{K}'$. W. l. o. g., we can assume that the first quantifier for both languages uses the same function g and that the threshold for both languages is "$>$ half of the strings". Furthermore, we assume that only strings of length exactly $g(|x|,k)$ are counted and that $g \in \mathrm{FFPT}$. This means there are $B_1, B_2 \in \mathcal{K}'$, a computable function h and $g \in \mathrm{FFPT}$ with $g(|x|,k) \leq h(k) \cdot \log|x|$ s. t. f. a. (x,k) and for $i \in \{1,2\}$:

$$(x,k) \in L_i \Leftrightarrow \mathrm{C}[\kappa]^g_{2^{g(|x|,k)-1}+1} y : |y| = g(|x|,k), (\langle x,y\rangle,k) \in B_i$$

Claim. *W. l. o. g. we can assume that the the number of witnesses in B_1 and B_2 for L_1 and L_2, respectively, is never exactly half of all strings y.*

Proof of claim. Assume that the condition does not hold for B_1, B_2. Then we can change those languages and the functions f, g in a way that it does. This is done in the same way as when proving closure of W[P]-PFPT (and similar classes) under symmetric difference:
We define languages B_1', B_2' as follows:

$$B_i' := \{(\langle x, y_1 \ldots y_{2n}\rangle, k) \mid (\langle x, y_1 \ldots y_n\rangle, k) \in B_i \text{ and } y_{n+1} \ldots y_{2n} \neq \underbrace{1 \ldots 1}_{n}\},\ i \in \{1,2\}.$$

For a fixed (x,k), let $A_i := \{y \mid |y| = 2g(|x|,k), (\langle x,y\rangle,k) \in B_i'\}$. Now we have

$$\begin{aligned}
(x,k) \in L_i \Rightarrow\ & |\{y \mid |y| = g(|x|,k), (\langle x,y\rangle,k) \in B_i\}| \geq 2^{g(|x|,k)-1} + 1 \\
\Rightarrow\ & |A_i| \geq (2^{g(|x|,k)-1} + 1)(2^{g(|x|,k)} - 1) \\
\Rightarrow\ & |A_i| \geq 2^{2g(|x|,k)-1} + \underbrace{2^{g(|x|,k)-1}}_{\substack{\text{we can assume this} \\ \text{is } \geq 2\ \forall (x,k)}} - 1 \\
\Rightarrow\ & |A_i| \geq 2^{2g(|x|,k)-1} + 1
\end{aligned}$$

and

$$\begin{aligned}
(x,k) \notin L_i \Rightarrow\ & |\{y \mid |y| = g(|x|,k), (\langle x,y\rangle,k) \in B_i\}| \leq 2^{g(|x|,k)-1} \\
\Rightarrow\ & |A_i| \leq (2^{g(|x|,k)-1})(2^{g(|x|,k)} - 1) \\
\Rightarrow\ & |A_i| \leq 2^{2g(|x|,k)-1} - \underbrace{2^{g(|x|,k)-1}}_{\substack{\text{we can assume this} \\ \text{is } \geq 1\ \forall (x,k)}} \\
\Rightarrow\ & |A_i| < 2^{2g(|x|,k)-1}.
\end{aligned}$$

From this, for $i \in \{1,2\}$ and for all (x,k) we get:

$$(x,k) \in L_i \Leftrightarrow \quad \mathrm{C}[\kappa]^{2g}_{2^{2g(|x|,k)-1}+1} y : (\langle x,y\rangle,k) \in B_i'$$

and the number of y that fulfill the condition is never exactly half of all y. ■

Now let (x,k) be fixed, A_i as in the proof of the claim and $a_1, a_2 \in \mathbb{Z}$ such that

$$|A_i| = 2^{2g(|x|,k)-1} + a_i$$

and let

$$\begin{aligned}
t :=\ & \left| \left\{ \langle y_1, y_2\rangle \;\middle|\; \begin{array}{c} ((\langle x,y_1\rangle,k) \in B_1 \text{ and } (\langle x,y_2\rangle,k) \notin B_2) \\ \text{or } ((\langle x,y_1\rangle,k) \notin B_1 \text{ and } (\langle x,y_2\rangle,k) \in B_2) \end{array} \right\} \right| \\
=\ & (2^{2g(|x|,k)-1} + a_1)(2^{2g(|x|,k)-1} - a_2) + (2^{2g(|x|,k)-1} - a_1)(2^{2g(|x|,k)-1} + a_2) \\
=\ & 2^{2g(|x|,k)-1} - 2a_1a_2.
\end{aligned}$$

We now have

$$\begin{aligned}
(x,k) \in L_1 \triangle L_2 \Rightarrow\ & ((x,k) \in L_1 \text{ and } (x,k) \notin L_2) \text{ or } ((x,k) \notin L_1 \text{ and } (x,k) \in L_2) \\
\Rightarrow\ & (a_1 > 0 \text{ and } a_2 < 0) \text{ or } (a_1 < 0 \text{ and } a_2 > 0) \\
\Rightarrow\ & a_1a_2 \leq -1 \\
\Rightarrow\ & t \geq 2^{2g(|x|,k)-1} + 2
\end{aligned}$$

and

$$\begin{aligned}
(x,k) \notin L_1 \triangle L_2 \Rightarrow\ & ((x,k) \in L_1 \text{ and } (x,k) \in L_2) \text{ or } ((x,k) \notin L_1 \text{ and } (x,k) \notin L_2) \\
\Rightarrow\ & a_1, a_2 > 0 \text{ or } a_1, a_2 < 0 \\
\Rightarrow\ & a_1a_2 \geq 1 \\
\Rightarrow\ & t \leq 2^{2g(|x|,k)-1} - 2.
\end{aligned}$$

This leads to the following characterization of $L_1 \triangle L_2$:

$$\begin{aligned}(x,k) \in L_1 \triangle L_2 \Leftrightarrow \quad & \mathrm{C}[\kappa]^{2g}_{2^{2g(|x|,k)-1}+1} \langle y_1, y_2 \rangle : \begin{array}{c}[(((\langle x,y_1\rangle,k) \in B_1 \text{ and } (\langle x,y_2\rangle,k) \notin B_2) \text{ or} \\ (((\langle x,y_1\rangle,k) \notin B_1 \text{ and } (\langle x,y_2\rangle,k) \in B_2))]\end{array} \\ \Leftrightarrow \quad & \mathrm{C}[\kappa]^{2g}_{2^{2g(|x|,k)-1}+1} \langle y_1, y_2 \rangle : (\langle x,y_1\rangle, \langle x,y_2\rangle, k) \in B_1' \triangle B_2',\end{aligned}$$

where $B_1' := B_1 \times_\kappa (\Sigma^* \times \mathbb{N})$ and $B_2' := (\Delta^* \times \mathbb{N}) \times_\kappa B_2$ (Δ and Σ are the input alphabets of L_1 and L_2, respectively). By Lemma 7.9 $B_1', B_2' \in \mathcal{K}$ and by induction hypothesis $B_1' \triangle B_2' \in \mathcal{K}$. Hence, $L_1 \triangle L_2 \in \mathrm{C}[\kappa]\mathcal{K} = \mathcal{K}'$. $\square$

7.4 Machine Characterization of classes in $\mathrm{CH}[\kappa]$

In classical complexity theory, the classes in the counting hierarchy are characterized in terms of decidability by certain oracle Turing machines. We want to transfer this result to the parameterized version. Therefore, we need a few auxiliary results.

Lemma 7.11. *Let $\mathcal{K} \in \mathrm{CH}[\kappa]$. Then $\mathrm{C}_{=}[\kappa]\mathcal{K} \subseteq \mathrm{C}[\kappa]\mathcal{K} \triangle \mathrm{C}[\kappa]\mathcal{K}$.*

Proof. Let $L \in \mathrm{C}_{=}[\kappa]\mathcal{K}$ be a parameterized problem. Then there are functions f, g, h with $f \in \mathrm{FFPT}$, h is computable and $g(|x|,k) \leq h(k) \cdot \log|x|$ $\forall x, k$ as well as $A \in \mathcal{K}$ s. t. f. a. (x,k):

$$(x,k) \in L \Leftrightarrow \quad \mathrm{C}_{=}[\kappa]^g_f y : (x,y,k) \in A$$

We define two parameterized problems L_1, L_2 as follows:

$$\begin{aligned} L_1 := \quad & \{(x,k) \mid \mathrm{C}[\kappa]^g_f y : (x,y,k) \in A\} \quad \text{and} \\ L_2 := \quad & \{(x,k) \mid \mathrm{C}[\kappa]^g_{f+1} y : (x,y,k) \in A\}. \end{aligned}$$

Now we have $L_1, L_2 \in \mathrm{C}[\kappa]\mathcal{K}$ and $L = L_1 \triangle L_2$. $\square$

Corollary 7.12. *Let $\mathcal{K} \in \mathrm{CH}[\kappa]$. Then it holds that $\mathrm{C}_{=}[\kappa]\mathcal{K} \subseteq \mathrm{C}[\kappa]\mathcal{K}$.*

Proof. $\mathrm{C}_{=}[\kappa]\mathcal{K} \subseteq \mathrm{C}[\kappa]\mathcal{K} \triangle \mathrm{C}[\kappa]\mathcal{K} = \mathrm{C}[\kappa]\mathcal{K}$ by Lemma 7.11 and Theorem 7.10. $\square$

In order to prove the final result of this section, we need to show closure of classes defined with $\mathrm{C}_{=}[\kappa]$ under the following kind of unbounded Cartesian product:

Definition 7.13. A k-bounded Cartesian product of a language L is a language

$$L^{\times_\kappa} := \{(\langle x_1, \ldots, x_n \rangle, k) \mid \bigwedge_i x_i \in L, n \leq f(k)\},$$

where f is an arbitrary computable function.

Lemma 7.14. *Let $\mathcal{K} \in \mathrm{CH}[\kappa]$. Then $\mathrm{C}_{=}[\kappa]\mathcal{K}$ is closed under k-bounded Cartesian products.*

Proof. Let $L \in \mathrm{C}_{=}[\kappa]\mathcal{K}$. There are $B \in \mathcal{K}$, a computable function h and $f, g \in \mathrm{FFPT}$ with $g(|x|,k) \leq h(k) \cdot \log|x|$ $\forall x, k$ ($g \in \mathrm{FFPT}$ is assumed w. l. o. g.) s. t. f. a. (x,k):

$$(x,k) \in L \Leftrightarrow \mathrm{C}_{=}[\kappa]^g_f y : (\langle x,y \rangle, k) \in B$$

Let f_2 be the computable function bounding the number of strings x_i for words in $L^{\times_\kappa}$. Now let $(\langle x_1,\dots,x_n\rangle,k)$ be an input of $L^{\times_\kappa}$. We want to quantify a single string and then count the number of witnesses for all x_i. In order to do this, we quantify not only the string y, but a number as well. This number will be used as the index to determine for which input x_i we check membership. To get the number of witnesses for all x_i seperately we multiply it for all x_i by a high enough number dependent on i so that in their binary representation the numbers of witnesses for different x_i always use different parts. For this, we assume that g is monotonously increasing. We define $f'(\langle x_1,\dots,x_n\rangle,k)$ accordingly:

$$f'(\langle x_1,\dots,x_n\rangle,k) = f(x_1,k) + f(x_2,k)\cdot 2^{g(m,k)+1} + \dots + f(x_n,k)\cdot 2^{(g(m,k)+1)(n-1)}$$

with $m = \max\{|x_1|,\dots,|x_n|\}$. Now we have

$$\begin{aligned}(\langle x_1,\dots,x_n\rangle,k) \in L^{\times_\kappa} \Leftrightarrow &\quad \mathrm{C}\!\!=\![\kappa]^{g}_{f(x_1,k)} y_1 : |y_1| \le g(|x_1|,k), (\langle x_1,y_1\rangle,k) \in B \ \wedge\ \dots \\ &\wedge\ \mathrm{C}\!\!=\![\kappa]^{g}_{f(x_n,k)} y_n : |y_n| \le g(|x_n|,k), (\langle x_n,y_n\rangle,k) \in B \\ \Leftrightarrow &\quad \mathrm{C}\!\!=\![\kappa]^{g'}_{f'(\langle x_1,\dots,x_n\rangle,k)} \langle y,i,z\rangle : (\langle x_1,x_2,y,i,z\rangle,k) \in B',\end{aligned}$$

where B' is defined as

$$B' := \left\{ (\langle x_1,\dots,x_n,y,i,z\rangle,k) \;\middle|\; \begin{array}{c} n \le f_2(k), |y| \le g(|x_i|,k) and \\ (\langle x_i,y\rangle,k) \in B \text{ and } z_2 < 2^{(g(m,k)+1)(i-1)} \end{array} \right\}$$

and $g'(\langle x_1,\dots,x_n\rangle,k) := |\langle 2^{g(m,k)}, n, 2^{(g(m,k)+1)(n-1)}\rangle|$. Since n is bounded by $f_2(k)$, g' is bounded as required. Note, that z_2 is used to multiply the number of witnesses for the different x_i as described above.
We now need $B' \in \mathcal{K}$. For this, notice that almost the same quantifier-characterization as for B can be used for B' as well. The additional conditions can be "moved" over all quantifiers into the FPT-language after them (cf. proof of Lemma 7.6). □

Remark 7.15. *We will use a slightly modified version of Lemma 7.14 where the k-bounded Cartesian product is not built using only one language L, but a number of languages that are uniform in a certain way. We leave out the technical details, since closure under that operation can be shown analogously.*

Theorem 7.16. *Let* $\mathcal{K} \in \mathrm{CH}[\kappa]$. *Then* $\mathrm{W[P]\text{-}PFPT}^{\mathcal{K}[f(k),\kappa]} = \mathrm{C}[\kappa]\mathcal{K}$.

Proof. For $\mathcal{K} = \mathrm{FPT}$ the result is trivial using Lemma 7.8. Therefore we can assume that $\mathcal{K} = \mathrm{C}[\kappa]\mathcal{K}'$ for some $\mathcal{K}' \in \mathrm{CH}[\kappa]$.

$\supseteq$: Let $L \in \mathrm{C}[\kappa]\mathcal{K}$. There are $B \in \mathcal{K}$, a computable function h and $f, g \in \mathrm{FFPT}$ with $g(|x|,k) \le h(k)\cdot \log|x|$ $\forall x,k$ ($g \in \mathrm{FFPT}$ is assumed w. l. o. g.) s. t. f. a. (x,k):

$$(x,k) \in L \Leftrightarrow \mathrm{C}[\kappa]^{g}_{f} y : |y| = g(|x|,k), (\langle x,y\rangle,k) \in B$$

The following k-restricted nondeterministic oracle Turing machine with oracle B accepts L in PP-fashion:

The additional nondeterministic bit is used to shift the needed number of y with $(\langle x,y\rangle,k) \in B$ for which the machine accepts to the threshold $f(x,k)$. The machine only uses one oracle query per computation path and the parameter of that query is the parameter of the input.

```
Input: $(x,k)$
Nondeterministically guess $b \in \{0,1\}$ and $y \in \{0,1\}^{g(|x|,k)}$
if $b = 0$ then
    Query the oracle with $(\langle x,y\rangle,k)$
    Accept iff the oracle answer is yes
else
    Accept iff $y < 2^{g(|x|,k)} + 1 - f(x,k)$
end if
```

$\subseteq$: Let $L \in \mathrm{W[P]\text{-}PFPT}^{\mathcal{K}[f(k),\kappa]}$. There are $g_1 \in \mathrm{FFPT}$ with $g_1(|x|,k) \leq h_1(k) \cdot \log|x|$ for some computable function h_1, a k-restricted oracle NTM M that on every input (x,k) uses exactly $g_1(|x|,k)$ nondeterministic bits on all computation paths and $A \in \mathcal{K}$ s. t. $L = L(M,A)$. Therefore it holds that

$$(x,k) \in L \Leftrightarrow \mathrm{C}[\kappa]^{g_1}_{2^{g_1(|x|,k)-1}+1} y : |y| = g_1(|x|,k), M^A_y(x,k) \text{ accepts.}$$

Additionally, there is a computable function f s. t. f. a. (x,k) the number of oracle queries of $M(x,k)$ is exactly $f(k)$ on each computation path regardless of the oracle answers. We now need to show that the condition "$M^A_y(x,k)$ accepts" can be formulated as a language in $\mathcal{K}$. By Theorem 7.10 and Lemma 7.6, $A \uplus \overline{A} \in \mathcal{K} = \mathrm{C}[\kappa]\mathcal{K}'$. Hence, there are $B \in \mathcal{K}'$, a computable function h_2 and $f_2, g_2 \in \mathrm{FFPT}$ with $g_2(|x|,k) \leq h_2(k) \cdot \log|x|$ $\forall x,k$ ($g_2 \in \mathrm{FFPT}$ is assumed w. l. o. g.) s. t. f. a. (x,k):

$$(x,k) \in A \uplus \overline{A} \Leftrightarrow \mathrm{C}[\kappa]^{g_2}_{f_2} v : (\langle x,v\rangle,k) \in B$$

Now let

$$B' := \left\{ (\langle x,y,(a_1,z_1),\ldots,(a_m,z_m)\rangle,k) \;\middle|\; \begin{array}{c} m = f(k), \\ M_y(x,k) \text{ accepts iff the oracle replies with } a_1,\ldots,a_m \\ \text{and for } i = 1,\ldots,m : (\langle a_i q_i, z_i\rangle,k_i) \in B \text{ and there are} \\ (*)\,\text{exactly } f_2(a_i q_i,k_i) \text{ strings } z_i' \geq z_i \text{ with } (\langle a_i q_i, z_i'\rangle,k_i) \in B \end{array} \right\},$$

where (q_i,k_i) are the oracle queries of $M_y(x,k)$ when the oracle answers a_i are used.
$(*)$ in this definition for $i = 1,\ldots,m$ means that $(a_i q_i,k_i) \in A \uplus \overline{A}$ and z_i is the maximal string (in lexikographic order) such that no witnesses $\leq z_i$ are needed to show this. By definition, for any y there can only be one w such that $(\langle x,y,w\rangle,k) \in B'$. This leads to

$$\begin{array}{c} \mathrm{C}[\kappa]^{g_1}_{2^{g_1(|x|,k)-1}+1} y : \\ |y| = g_1(|x|,k), M^A_y(x,k) \text{ accepts} \end{array} \quad \Leftrightarrow \quad \begin{array}{c} \mathrm{C}[\kappa]^{g_1'}_{2^{g_1(|x|,k)-1}+1} \langle y,w\rangle : \\ |y| = g_1(|x|,k), (\langle x,y,w\rangle,k) \in B' \end{array} \quad .$$

We have to make sure that the length of the quantified string is bounded as needed. For y and the a_i this is trivial; for z_i it can be shown analogously to closure of W[P] under $\leq^{\mathrm{fpt}}$ (see e.g. [FG]). Since the number of strings is bounded by $f(k)$ the length is at most multiplied by $f(k)$, which is not a problem. We will not go into any more detail on the function g_1'.
We now rewrite $(*)$ for $i = 1,\ldots,m$ as follows:

$$\mathrm{C}\!\!=[\kappa]^{g_2(|a_i q_i|,k_i)}_{f_2(a_i q_i,k_i)} z_i' : [(((\langle a_i q_i, z_i\rangle,k_i) \in B), z_i \leq z_i' \text{ and } (\langle a_i q_i, z_i'\rangle,k_i) \in B] \quad .$$

Obviously, for all i for $|z_i'|$ the same bounds as for $|z_i|$ hold. This means that $(*)$ is fpt-reducible to a k-bounded Cartesian product of $\mathrm{C}\!\!=[\kappa]\mathcal{K}'$-languages (the fpt-reduction is only used to transform the input to fit for the Cartesian product). Thus, by Lemma 7.14 and closure of $\mathrm{C}\!\!=[\kappa]\mathcal{K}'$ under $\leq^{\mathrm{fpt}}$, $(*)$ is a $\mathrm{C}\!\!=[\kappa]\mathcal{K}'$-condition. Since $\mathrm{C}\!\!=[\kappa]\mathcal{K}'$ is closed under intersection with FPT-languages, we obtain $B' \in \mathrm{C}\!\!=[\kappa]\mathcal{K}'$.
Combining our results, we obtain $L \in \mathrm{C}[\kappa]\mathrm{C}\!\!=[\kappa]\mathcal{K}'$ and by Corollary 7.12 $L \in \mathrm{C}[\kappa]\mathrm{C}[\kappa]\mathcal{K}' = \mathrm{C}[\kappa]\mathcal{K}$.
$\square$

Remark 7.17. *In classical complexity theory, the previous theorem holds without bounding the oracle queries in any way. It is unlikely that this is possible in our case, as there could be up to fpt-many oracle calls so that it would not even be possible to quantify the oracle answers (one bit per query).*

8 Conclusion and Outlook

In this work, we further studied different counting classes within parameterized complexity theory. We began by showing a few basic properties of parameterized function classes as auxiliary results for the later chapters.
We classified the parameterized weighted threshold-satisfiability problem over all fragments from Post's lattice for both circuits and formulae using the techniques from [CV]. For circuits, we even got a slightly better result due to the nature of the problem: Over the difficult fragments, the problem is complete under $\leq^{\text{fpt}}$—compared to $\leq^{\text{fpt-T}}$, which is known for the counting version [CV].
Concerning structural properties of counting classes, we have shown that the Boolean algebra over W[P] is contained in W[P]-PFPT. Furthermore, we introduced the classes W[P]-$\mathrm{C}_=$FPT and W[P]-$\oplus$FPT and examined their closure properties. For both classes most of the closure properties of the analogue classes from classical complexity theory could be transferred. In addition, we gave an $\leq^{\text{fpt}}$-complete satisfiability problem for W[P]-$\oplus$FPT and investigated the connection between W[P]-PFPT and W[P]-$\mathrm{C}_=$FPT. Again, we achieved an analogous result to classical complexity theory, that is, W[P]-$\mathrm{C}_=$FPT $\subseteq$ W[P]-PFPT.
Finally, we generalized the class W[P]-PFPT using the counting quantifier $\mathrm{C}[\kappa]$. We also defined the exact counting quantifier $\mathrm{C}_=[\kappa]$, which was used auxiliarily. We defined the parameterized counting hierarchy using $\mathrm{C}[\kappa]$ and studied the closure properties of the classes in this hierarchy. The closure under Boolean operations was the same as in classical complexity theory, again. Concerning a characterization of the classes in $\mathrm{CH}[\kappa]$ in terms of oracle Turing machines the oracle queries had to be bounded: We showed $\text{W[P]-PFPT}^{\mathcal{K}[f(k),\kappa]} = \mathrm{C}[\kappa]\mathcal{K}$.

We now want to give a few suggestions for further research concerning the topics studied in this thesis. The parameterized weighted threshold-satisfiability problem for circuits has been classified over all clones from Post's lattice. For formulae on the other hand, it could be interesting to try to show completeness under $\leq^{\text{fpt}}$ instead of $\leq^{\text{fpt-T}}$ over the clones over which we showed the latter.
The counting hierarchy in classical complexity theory is defined as an extension of the polynomial time hierarchy using arbitrary combinations of the quantifiers $\exists, \forall$ and C. Thus, a next step could be to introduce parameterized versions of $\exists$ and $\forall$ in order to define a parameterized analogon of the polynomial time hierarchy and combine it with the counting hierarchy defined here to obtain a full counting hierarchy. It is likely that many structural properties of this hierarchy can be transferred from the classical version. In addition, complete problems for the classes in $\mathrm{CH}[\kappa]$ (or even the full counting hierarchy) could be defined.
Apart from this, the main open problems in this area are the results that could not be transferred from classcial complexity theory due to the k-restriction. Examples for such results are closure of W[P]-PFPT under intersection, the different types of amplification (known from PP) for W[P]-PFPT, $\text{W[P]-}\oplus\text{FPT}^{\text{W[P]-}\oplus\text{FPT}} = \text{W[P]-}\oplus\text{FPT}$ and $\text{W[P]-PFPT}^{\mathcal{K}} = \mathrm{C}[\kappa]\mathcal{K}$. All of these have the same problem: When directly transferring the known proofs, the number of nondeterministic bits (or the length of the quantified string) would not be k-restricted at some point during the proof. This problem is inherent to k-restricted machines and cannot be bypassed generally. Hence, searching for negative results might be better in most cases.

9 References

[FG] Flum, J., Grohe, M.: *Parameterized Complexity Theory*, Springer, 2006

[CR] Chauhan, A., Rao, B. V. R.: *Parameterized Analogues of Probabilistic Computation*, 2014, `arxiv.org/abs/1409.7790`

[CV] Craignou, N., Vollmer, H.: Parameterized Complexity of Weighted Satisfiability Problems: Decision, Enumeration, Counting, *Fundamenta Informaticae*, **136**(4), 2015, 297-316

[Schn] Schnoor, H.: The Complexity of Model Checking for Boolean Formulas, *Int. J. Found. Comput. Sci.*, **21**(3), 2010, 289-309

[Tho] Thomas, M.: On the applicability of Post's lattice, *Information Processing Letters*, **112**(10), 2012, 386-391

[Vol] Vollmer, H.: Skript zur Vorlesung Komplexitätstheorie (Wintersemester 2013/2014)

[Tor] Torán, J.: Complexity Classes Defined by Counting Quantifiers, *Journal of the ACM*, **38**(3), 1991, 752-773

[KSW] Köbler, J., Schöning, U., Wagner, K. W.: The Difference and Truth-table Hierarchies for NP, *RAIRO - Theoretical Informatics and Applications*, **21**(4), 1987, 419-435

[CKTV] Caussinus, H., McKenzie, P., Thérien, D., Vollmer, H.: Nondeterministic NC^1 Computation, *Journal of Computer and System Sciences*, **57**(2), 1998, 200-212