On W[1]-Hardness as Evidence for Intractability

Ralph Christian Bottesch
University of Innsbruck, Innrain 52, 6020 Innsbruck, Austria
ralph.bottesch@uibk.ac.at

Abstract

The central conjecture of parameterized complexity states that FPT $\neq$ W[1], and is generally regarded as the parameterized counterpart to P $\neq$ NP. We revisit the issue of the plausibility of FPT $\neq$ W[1], focusing on two aspects: the difficulty of proving the conjecture (assuming it holds), and how the relation between the two classes might differ from the one between P and NP.

Regarding the first aspect, we give new evidence that separating FPT from W[1] would be considerably harder than doing the same for P and NP. Our main result regarding the relation between FPT and W[1] states that the closure of W[1] under relativization with FPT-oracles is precisely the class W[P], implying that either FPT is not low for W[1], or the W-Hierarchy collapses. This theorem also has consequences for the A-Hierarchy (a parameterized version of the Polynomial Hierarchy), namely that unless W[P] is a subset of some level A[t], there are structural differences between the A-Hierarchy and the Polynomial Hierarchy. We also prove that under the unlikely assumption that W[P] collapses to W[1] in a specific way, the collapse of any two consecutive levels of the A-Hierarchy implies the collapse of the entire hierarchy to a finite level; this extends a result of Chen, Flum, and Grohe (2005).

Finally, we give weak (oracle-based) evidence that the inclusion W[t] $\subseteq$ A[t] is strict for $t > 1$, and that the W-Hierarchy is proper. The latter result answers a question of Downey and Fellows (1993).

2012 ACM Subject Classification Theory of computation → Complexity classes

Keywords and phrases Parameterized complexity, Relativization

Digital Object Identifier 10.4230/LIPIcs.MFCS.2018.73

Related Version A full version of the paper is available at `https://arxiv.org/abs/1712.05766`.

Funding This work was supported by the ERC Consolidator Grant QPROGRESS 615307 for the majority of its duration, and by the Austrian Science Fund (FWF) project Y757 at the time of publication.

Acknowledgements I thank Harry Buhrman, Sándor Kisfaludi-Bak, and Ronald de Wolf for helpful discussions. I am especially grateful to Ronald de Wolf and Leen Torenvliet for helpful comments on drafts of the paper.

1 Introduction

The central conjecture of parameterized complexity theory states that FPT $\neq$ W[1]. The complexity class FPT is a generalization of P, and it also contains this class in the sense that regardless of which parameter we associate with the instances of a problem in P, the resulting *parameterized problem* is in FPT. This inclusion is strict, as FPT also contains parameterized versions of problems that are provably not in P. The class W[1] can be regarded as a parameterized counterpart to NP. It can be defined in different ways, all

43rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2018).
Editors: Igor Potapov, Paul Spirakis, and James Worrell; Article No. 73; pp. 73:1–73:15

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

of them quite technical, but the most common definition is in terms of a parameterized version of a particular NP-complete problem (much like NP can be defined in terms of a Boolean circuit satisfiability problem). However, W[1] is not known or believed to contain *all* parameterized versions of problems in NP, and by defining complexity classes in terms of parameterizations of other NP-complete problems, one actually obtains a large set of seemingly distinct parameterized analogues of NP, some of which we list here:

$$\mathsf{A}[1] = \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \mathsf{W}[3] \subseteq \ldots \mathsf{W}[t] \ldots \subseteq \mathsf{W}[\mathsf{P}] \subseteq \mathsf{para\text{-}NP}.$$

Among these, the most interesting classes are W[1] (a.k.a. A[1]) and W[P], due to having many natural complete problems.

The basic intuition for why W[1] (and hence all classes in the above sequence) should differ from FPT is the same as for P ≠ NP, namely that we do not know of any way to efficiently simulate nondeterministic computations deterministically. This intuition is often used to justify considering the W[1]-hardness of a problem as evidence for its intractability. But because FPT is strictly larger than P, while W[1] does not appear to capture all of the complexity of NP, it seems that proving the central conjecture of parameterized complexity theory may be harder than separating P and NP. We investigate qualitative differences between the two conjectures, as well as the more general question of whether FPT occupies the same place within W[1] as P does within NP. We start by giving a brief summary of some relevant prior results.

That the central parameterized conjecture is at least as strong as its classical counterpart is easy to prove: If NP = P, then, as noted above, every parameterized version of every problem in NP(= P) must be in FPT, hence W[1] = FPT (and, in fact, para-NP = W[P] = ... = FPT). Thus we have that FPT ≠ W[1] ⇒ P ≠ NP. The converse of this implication is not known to hold, but Downey and Fellows [9] were the first to observe that a collapse of W[1] to FPT would at least imply the existence algorithms with sub-exponential running time for the NP-complete problem 3Sat. This would contradict the *Exponential Time Hypothesis (ETH)*, first introduced by Imagliazzo, Paturi, and Zane [13], which states that for some constant $c > 0$, 3Sat can not be solved in time $O^*(2^{cn})$ by deterministic Turing machines (TMs). This conjecture has enjoyed much popularity recently, because, assuming ETH, for many problems it is possible to prove a complexity lower bound that matches that of the best known algorithm up to lower-order factors (see [14] for a survey of such results). Nevertheless, one should keep in mind that ETH is a much stronger statement than P ≠ NP, since it rules out not only the existence of polynomial-time algorithms for 3Sat, but also of those that run in up to exponential-time (for some bases). Putting all of these facts together, we have:

$$\text{ETH} \Longrightarrow \mathsf{FPT} \neq \mathsf{W}[1] \Longrightarrow \ldots \Longrightarrow \mathsf{FPT} \neq \mathsf{W}[\mathsf{P}] \Longrightarrow \mathsf{FPT} \neq \mathsf{para\text{-}NP} \Longrightarrow \mathsf{P} \neq \mathsf{NP}.$$

The above sequence relates parameterized complexity conjectures to two classical ones, but it does not say which of them are closer in strength to ETH and which are closer to P ≠ NP. The only known fact here is that FPT ≠ para-NP ⇔ P ≠ NP (see [11, Corollary 2.13]), but there is strong evidence suggesting that all of the other parameterized conjectures listed above are considerably stronger than P ≠ NP (although possibly still weaker than ETH[1]). First, Downey and Fellows [8] construct an oracle relative to which P and NP differ while W[P]

[1] There is, in fact, a subclass of W[1], called M[1], of which it is known that FPT ≠ M[1] is equivalent to ETH (see [10]). The similarities between M[1] and W[1] can be seen as a further indication that the conjecture FPT ≠ W[1] is nearly as strong as ETH, but, evidently, both FPT ≠ M[1] and M[1] ≠ W[1] are wide open conjectures.

collapses to FPT, so we know that any proof of the implication $\mathsf{P} \neq \mathsf{NP} \Rightarrow \mathsf{FPT} \neq \mathsf{W[P]}$ can not be as simple as the proof of the converse implication sketched above. More importantly, $\mathsf{FPT} \neq \mathsf{W[P]}$ can be related much more precisely to other classical complexity conjectures.

How strong the assumption $\mathsf{FPT} \neq \mathsf{W[P]}$ is, can be elegantly expressed in terms of *limited nondeterminism*. If f is a poly-time-computable function, denote by $\mathsf{NP}[f(n)]$ the class of problems that can be solved by a nondeterministic TM in polynomial-time by using at most $O(f(n))$ bits of nondeterminism (n denotes the size of the input). Note that $\mathsf{NP}[\log n] = \mathsf{P}$, since a deterministic TM can cycle through all possible certificates of length $O(\log n)$ in polynomial-time. A remarkable theorem of Cai, Chen, Downey, and Fellows [6] states that $\mathsf{FPT} \neq \mathsf{W[P]}$ holds *if and only if* for *every* poly-time-computable non-decreasing unbounded function h, we have that $\mathsf{P} \neq \mathsf{NP}[h(n) \log n]$ (see [11, Theorem 3.29] for a proof of the theorem in this form). The class of functions referred to in this theorem contains functions with very slow growth, such as the iterated logarithm function, $\log^*$. In fact, there is no poly-time-computable non-decreasing unbounded function that has the slowest growth, because if some function h satisfies these conditions, then so does $\log^* h$. It is not even intuitively clear whether P is different from NP when the amount of allowed nondeterminism is arbitrarily close to trivial. At the very least, the fact that an infinite number of increasingly strong separations must hold in order for W[P] to not collapse to FPT, suggests that separating these two classes is much farther out of our reach than a separation of P and NP.

The evidence we have seen so far indicates that proving a separation of W[1] and FPT may be harder than proving $\mathsf{P} \neq \mathsf{NP}$. But assuming that both conjectures hold, it is meaningful to ask whether the internal structure of W[1] resembles that of NP, and there is indeed some positive evidence in this direction. For example, a parameterized version of Cook's Theorem connects Boolean circuit satisfiability to W[1]-completeness (see [9]), a parameterized version of Ladner's Theorem states that if $\mathsf{FPT} \neq \mathsf{W[1]}$, then there is an infinite hierarchy of problems with different complexities within W[1] (see [9]), and the machine-based characterizations of this class, due to Chen, Flum, and Grohe [7], establish that W[1] can indeed be defined in terms of nondeterministic computing machines. Nevertheless, there are also previously unexplored ways in which W[1] may not behave the same way as NP.

Our main goal in this work is to provide further evidence that the classes FPT and W[1] are close not only in the sense of being difficult to separate, but also in the sense that the relationship between the two differs from that of P and NP, in a way that indicates that FPT is larger within W[1] than P is within NP (assuming the latter pair does not collapse). These results contrast with those in [4], where we showed how certain theorems about FPT and the levels of the A-Hierarchy can be proved in the same way as for their classical counterparts.

1.1 Summary of our results

The difficulty of separating W[P] from FPT. Assuming that we could prove a separation of the form $\mathsf{P} \neq \mathsf{NP}[h(n) \log n]$ for a particular, slow-growing function h, how much progress would we have made towards proving the separation where $h(n)$ is replaced by $\log h(n)$? Intuitively, the difficulty of proving non-equality should increase when a function with a slower growth is chosen. On the other hand, if $\mathsf{FPT} \neq \mathsf{W[P]}$ holds, then all such classical separations hold as well (by the above-mentioned theorem of Cai *et al.* [6]), and therefore any one of them implies the others. It is not clear, however, whether a proof of $\mathsf{FPT} \neq \mathsf{W[P]}$ with $\mathsf{P} \neq \mathsf{NP}[h(n) \log n]$ as a hypothesis would be significantly simpler than a proof from scratch. We show that this is unlikely to be the case, by proving (Theorem 9) that for any poly-time-computable non-decreasing unbounded function h, there exists a computable oracle O_h such that:

$$\mathsf{P}^{O_h} \neq \mathsf{NP}[h(n) \log n]^{O_h}, \text{ but } \mathsf{W[P]}^{O_h} = \mathsf{FPT}^{O_h}.$$

Theorem 9 is an improvement over the above-mentioned oracle construction of Downey and Fellows [8][2]. It is weak as a barrier result, since the relativization barrier has been repeatedly overcome in the last three decades, but nevertheless the theorem succinctly expresses how much harder the conjecture $\mathsf{FPT} \neq \mathsf{W}[\mathsf{P}]$ is compared to classical questions regarding nondeterministic vs. deterministic computation: No matter how small the amount of nondeterminism that provably yields a class strictly containing P, we will always be a non-trivial proof step away from separating $\mathsf{W}[\mathsf{P}]$ (or $\mathsf{W}[1]$) from FPT.

The structure of W[1] and its relation to FPT. The class $\mathsf{A}[1](=\mathsf{W}[1])$[3] can be characterized in terms of random access machines that perform *tail-nondeterministic* computations [7]. Such computations consist of two phases: 1. a (deterministic) FPT-computation; 2. a short nondeterministic computation that can use any data computed in phase 1. Tail-nondeterministic machines that perform only the second phase of the computation (without a longer deterministic computation preceding it), can not solve every problem in FPT, but, paradoxically, they can solve many problems that are complete for $\mathsf{A}[1]$ (we give an example in Section 4). As we will see, this simple observation has important consequences for the structure of this class.

A first consequence is that giving an $\mathsf{A}[1]$-machine very restricted oracle access to even a tractable (FPT) problem, may increase its computational power, because then the use of nondeterminism can be combined with the ability to solve instances of an FPT-problem via the oracle. Thus, FPT-computations appear to constitute a non-trivial computational resource for $\mathsf{A}[1]$ (unlike P-computations for NP). Somewhat suprisingly, we can actually identify the complexity class resulting from endowing $\mathsf{A}[1]$ with FPT-oracles, if a suitable, highly restricted type of oracle access is used. We have (Theorem 12, Corollary 13) that:

$$\mathsf{A}[1]^{\mathsf{FPT}} = \mathsf{W}[\mathsf{P}] \quad \text{and} \quad \forall t \geq 1 : \mathsf{W}[t]^{\mathsf{FPT}} = \mathsf{W}[\mathsf{P}],$$

where we used the common notation $\mathcal{C}_1^{\mathcal{C}_2} := \bigcup_{Q \in \mathcal{C}_2} \mathcal{C}_1^Q$. This means that either $\mathsf{W}[\mathsf{P}] = \mathsf{W}[1]$, in which case $\mathsf{W}[\mathsf{P}]$ is smaller than generally believed, or FPT is larger within $\mathsf{W}[1]$ than P is within NP.

Putting the known and new facts together, Theorem 9 and the result of Cai *et. al.* [6] mentioned in the introduction indicate that $\mathsf{W}[\mathsf{P}]$ is likely to be closer to FPT than any class $\mathsf{NP}[h(n) \log n]$ is to P (see Figure 1). The case for this figure being accurate is further strengthened by Theorem 12 and Corollary 13, which exhibit another way in which at least two of the classes FPT, $\mathsf{W}[1]$, and $\mathsf{W}[\mathsf{P}]$ are close.

Theorem 12 and the observation preceding it also have consequences for the A-Hierarchy, which is a parameterized analogue of PH. Although they share some essential properties [7, 4], a corollary of Theorem 12 is that, unless some unlikely inclusions between complexity classes occur, the two hierarchies have structural differences that indicate that consecutive levels of the A-Hierarchy are closer to each other than the corresponding levels of PH (see

[2] Actually, Downey and Fellows [8] use a different computational model to define and relativize $\mathsf{W}[\mathsf{P}]$, so the two results, although in the same spirit, may not be directly comparable at a technical level.

[3] $\mathsf{W}[1]$ and $\mathsf{A}[1]$ coincide as complexity classes, but in [7], Chen, Flum, and Grohe give two machine-based characterizations, one which can be generalized to get the levels of the W-Hierarchy, and one which generalizes to the levels of the A-Hierarchy. The machine model for $\mathsf{A}[1]$ is easier to handle when working with oracles, so we typically use this model when relativizing this class, and write "$\mathsf{A}[1]$" to emphasize this fact. However, oracle $\mathsf{W}[1]$-machines can also be defined so that our theorems hold for this model as well (see Section 3).

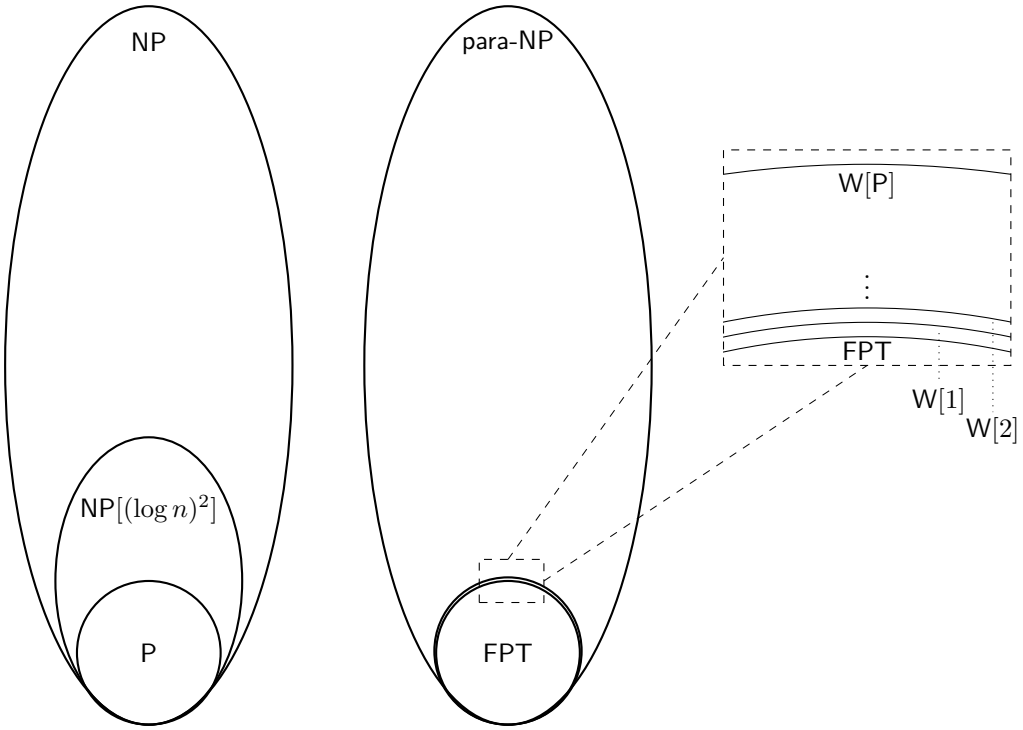

Figure 1 The mutual closeness of the parameterized complexity classes, compared to that of their classical analogues, as suggested by [6, 8], Theorems 9 and 12, and Corollary 13. Regardless of which class $\mathsf{NP}[h(n)\log n]$ we choose to represent between P and NP on the left side (whether it is $\mathsf{NP}[(\log n)^2]$ as in the picture, $\mathsf{NP}[\log^* n \log n]$, or something even smaller), it will be much larger compared to P than $\mathsf{W}[\mathsf{P}]$ is compared to FPT.

Section 4). Conversely, using a similar idea as in the proof of Theorem 12, we can show (Theorem 15) that if $\mathsf{W}[\mathsf{P}]$ were to collapse to $\mathsf{W}[1]$ in a specific way, we would get a *downward separation* theorem for the A-Hierarchy (i.e., that if two levels collapse, the entire hierarchy collapses to the smaller of the two). Proving such a theorem for the A-Hierarchy has been a long-standing open problem in parameterized complexity theory, and although our theorem falls short of this goal (since it requires an unlikely collapse to occur), it marks the first progress on this front in over a decade (since [7]).

Level-by-level relativized separations of the W- and the A-Hierarchy. We also give some evidence that certain collapses do not occur. The only relations that are known to hold between the classes $\mathsf{W}[t]$ and $\mathsf{A}[t]$ are that $\mathsf{W}[1] = \mathsf{A}[1]$ and that $\mathsf{W}[t] \subseteq \mathsf{A}[t]$ for $t \geq 2$. We show that in a relativized setting, the known inclusions can be made strict and some unexpected inclusions can be ruled out.

Separations of complexity classes relative to oracles count only as very weak evidence that the unrelativized versions of the classes are distinct, due to the fact that such oracles can in some cases be constructed even when two classes coincide (the most famous example being $\mathsf{IP} = \mathsf{PSPACE}$ [16] – see [12] for an oracle separating the two). Nevertheless, there are a few reasons why level-by-level relativized separations for the W- and the A-Hierarchy are interesting: First, since it is generally assumed that these hierarchies are proper and distinct, we should expect to have at least this weak form of evidence supporting the assumption.

Second, we have seen a number of results which suggest that the levels of these hierarchies are in various ways close to each other, so proving even relativized separations between them may be non-trivial. Finally, relativization in the parameterized setting is still mostly unexplored, and although the proofs of the following theorems rely on standard diagonalization arguments, the details of the machine models and how they are allowed to access oracles require special care in order to make the arguments work.

In Section 5 we show (Theorem 16, Corollary 17) that there exists a computable parameterized oracle O such that

$$\forall t \geq 2 : \mathsf{A}[t]^O \not\subset \mathsf{W}[t]^O.$$

Note that this is a *single* oracle relative to which all inclusions are simultaneously made strict. Also note that, although we use machine-based characterizations of classes $\mathsf{A}[t]$ and $\mathsf{W}[t]$ which result in distinct characterizations of the class $\mathsf{A}[1] = \mathsf{W}[1]$, fortunately, this oracle does not appear to separate $\mathsf{A}[1]$ from $\mathsf{W}[1]$. Such a separation would have suggested that the strict inclusions are mere artifacts of the machine models used.

Finally, we give evidence which suggests that the W-Hierarchy is not contained within any finite level of the A-Hierarchy (Theorem 18): For all $t \geq 1$, there exists a computable parameterized oracle O_t such that

$$\mathsf{W}[t+1]^{O_t} \not\subset \mathsf{A}[t]^{O_t}.$$

Since it holds that $\mathsf{W}[t]^{O_t} \subseteq \mathsf{A}[t]^{O_t}$, each oracle O_t separates two consecutive levels of the W-Hierarchy. This answers a question of Downey and Fellows [8], although we do not have a single oracle that simultaneously separates the entire hierarchy.

2 Preliminaries

We assume familiarity with standard facts and notations from both classical and parameterized complexity theory, and refer to [2] and to [11] for the necessary background in the respective branches. Since the characterizations of various parameterized complexity classes in terms of computing machines [7] are less well known, we give a brief overview of the main definitions.

Many parameterized complexity classes can only be naturally characterized in terms of *random access machines (RAMs)*, which can store entire integers in each of their registers, perform the operations addition, subtraction, and division by 2 on integers in unit time, and can access any part of their memory in constant time (see [15] or the introduction of [7]). The input of a RAM can be a sequence of non-negative integers, and we allow the instances of problems to be encoded in this way whenever we are working only with RAMs (as opposed to TMs). Since the size of a sequence of non-negative integers is calculated as the sum of the length of the binary representations of the individual numbers, RAMs have no significant computational advantage over TMs [15, Theorem 2.5]. However, this encoding does make a difference when considering oracle RAMs, because the query instances will also be encoded in this fashion.

We give two examples of definitions of complexity classes in terms of RAMs. It is not difficult to see that these are equivalent to the standard (TM-based) definitions (see [11]). Note that we use the Downey-Fellows definition of parameterized problems [9], where the parameter value, encoded in unary, is given together with the input.

▶ **Definition 1.** Let Q be a parameterized problem. We say that $Q \in \mathsf{FPT}$ if and only if there exists a RAM M, a computable function f, and a constant $c \geq 0$, such that for every input (x, k) with $x \in \mathbb{N}^*$ and $k \geq 0$, M runs in time $f(k)(|x| + k)^c$ and accepts if $(x, k) \in Q$, otherwise it rejects. The class $\mathsf{para\text{-}NP}$ is defined similarly, except with RAMs which can nondeterministically guess, in unit time, positive integers of size upper-bounded by the $f(k)(|x| + k)^c$ (the bound on the running time).

We also collect several useful definitions and notations in the following:

▶ **Definition 2.** Let $\mathcal{C}_1, \mathcal{C}_2$ be complexity classes, where $\mathcal{C}_1$ is defined in terms of computing machines that can be given access to an oracle, and let $P_0, P_1 \subseteq \{0,1\}^*$ be classical languages. We define: $\mathcal{C}_1^{\mathcal{C}_2} := \bigcup_{P \in \mathcal{C}_2} \mathcal{C}_1^P$ and $P_0 \oplus P_1 := \{0x \mid x \in P_0\} \cup \{1x \mid x \in P_1\}$. We say that P_0 is *low for* $\mathcal{C}_1$ if $\mathcal{C}_1^{P_0} = \mathcal{C}_1$, and we say that $\mathcal{C}_2$ is *low for* $\mathcal{C}_1$ if $\mathcal{C}_1^{\mathcal{C}_2} = \mathcal{C}_1$.

2.1 The A-Hierarchy and the W-Hierarchy

The following classes are defined in terms of *alternating random access machines (ARAMs)*, which are RAMs that can nondeterministically guess, in unit time, integers of size bounded by the running time of the machine on a given input, either in the existential or the universal mode (see [7]).

▶ **Definition 3** ([7])**.** For each $t \geq 1$, let $\mathsf{A}[t]$ be the class of parameterized problems that are solved by some ARAM A which, for some computable functions f and h, and a constant $c \geq 0$, satisfies the following conditions on every input (x, k):

1. A runs in time at most $f(k)(|x| + k)^c$;
2. throughout the computation, the values in A's registers do not exceed $f(k)(|x| + k)^c$;
3. all nondeterministic guesses are made during the last $h(k)$ steps of the computation;
4. the first nondeterministic guess is existential and the machine alternates at most $t - 1$ times between existential and universal guesses.

The class $\mathsf{co\text{-}A}[1]$ is defined in terms of ARAMs which satisfy conditions 1–3, but only make universal nondeterministic guesses (one can verify, just as in the classical setting, that a problem is in $\mathsf{co\text{-}A}[1]$ if and only if it is the complement of a problem in $\mathsf{A}[1]$). ARAMs satisfying conditions 1 and 2 are called *parameter-bounded* in [4], those satisfying conditions 3 and 4 are called, respectively, *tail-nondeterministic* and *t-alternating* [7].

The classes $\mathsf{W}[t]$ ($t \geq 1$) can be defined in terms of $\mathsf{A}[t]$-machines (parameter-bounded tail-nondeterministic t-alternating ARAMs) that are further restricted so that: 1. Every block of nondeterministic guess instructions of the same kind, except the first one, is made up of at most c' guess instructions, where c' is a constant that is independent of the input. 2. All nondeterministically guessed integers are placed in a special set of *guess registers*, which can not be read from directly, and can only be accessed via special instructions that use the guessed values as indices for accessing standard registers. We will not need further details regarding these machines, and therefore refer the reader to [7] or [5] for more complete definitions. We will, however, define oracle $\mathsf{W}[t]$-machines (Definition 10).

▶ **Definition 4** ([4])**.** An *oracle (A)RAM* is a machine with an additional set of registers called *oracle registers*, instructions that allow the machine to copy values from its standard registers to the oracle registers, as well as a QUERY instruction, the execution of which results in one of the values 1 or 0 being placed into the first standard register of the machine, depending on whether the instance encoded in the oracle registers at that time constitute a 'yes'- or a 'no'-instance of a problem for which the machine is said to have an oracle.

An oracle (A)RAM has *balanced oracle access* to a parameterized oracle, if there is a computable function g such that on every input (x, k), the machine queries the oracle only with instances whose parameter value is $\leq g(k)$ (in other words, the parameter values of the instances for which the oracle is called should be upper-bounded by some function of k, but may not depend on n, even though the machine may have time to construct such a query instance). An oracle (A)RAM has *tail-restricted oracle access*, if its access to the oracle is balanced and, furthermore, there is a computable function h such that the machine makes oracle queries only within the last $h(k)$ steps of the computation on input (x, k). Note that tail-restricted access is also balanced.

For a parameterized complexity class $\mathcal{C}$ that is defined in terms of (A)RAMs, we write $\mathcal{C}(O)$ if $\mathcal{C}$ has unrestricted access to the oracle O, $\mathcal{C}(O)_{bal}$ if it has balanced access, and $\mathcal{C}(O)_{tail}$ if it has tail-restricted access. If $\mathcal{C}$ is defined in terms of tail-nondeterministic ARAMs, we also write $\mathcal{C}^O$ instead of $\mathcal{C}(O)_{tail}$ (so $\mathsf{A}[1]^O$ means $\mathsf{A}[1]$ with tail-restricted access to O). Note that for $\mathcal{C}(O)$, the oracle can be either classical or parameterized, but for balanced or more restricted oracle access, it must be parameterized.

2.2 W[P] and the W[P]-Hierarchy

We define W[P] both in terms of TMs and in terms of RAMs, and use both definitions at different points in the paper.

▶ **Definition 5** ([7]). Let Q be a parameterized problem. We say that $Q \in \mathsf{W}[\mathsf{P}]$ if and only if there exists a nondeterministic TM M, computable functions f and h, and a constant $c \geq 0$, such that for any input (x, k) with $x \in \{0, 1, \#\}^*$ and $k \geq 0$, M runs in time $f(k)(|x| + k)^c$, uses at most $h(k)\lceil \log(|x| + k) \rceil$ nondeterministic bits, and accepts if and only if $(x, k) \in Q$.

The following problem is complete for W[P] under fpt-reductions.

p-WSatCircuit	
Input:	A Boolean circuit C with n input bits, $k \in \mathbb{N}$.
Parameter:	k
Problem:	Decide whether C has a satisfying assignment of weight k.

The class W[P] can also be defined in terms of RAMs [7]. One can also define a hierarchy that is similar to PH, except in terms of alternating nondeterminism that matches the nondeterminism of W[P].

▶ **Definition 6** ([4]). For each $t \geq 1$, let $\Sigma_t^{[P]}$ be the class of parameterized problems that are solved by some ARAM A which, for some computable functions f and h, and a constant $c \geq 0$, satisfies, on every input (x, k), conditions 1, 2, and 4 from Definition 3, as well as:

3'. *A* nondeterministically guesses at most $h(k)$ numbers throughout the computation.

We denote the class $\bigcup_{t=1}^{\infty} \Sigma_t^{[P]}$ by W[P]H, the W[P]-*Hierarchy*.

It is not difficult to see that $\mathsf{W}[\mathsf{P}] = \Sigma_1^{[P]}$. Note that we will use the term "W[P]-machine" to designate both the TMs from Definition 5 as well as the $\Sigma_1^{[P]}$-machines from Definition 6 (which are RAMs), but it should be clear from the context which type of machine is meant.

For each $t \geq 1$, the following generalizations of the problem p-WSatCircuit can easily be seen to be, respectively, complete problems for $\Sigma_t^{[P]}$.

p-AWSatCircuit$_t$	
Input:	A Boolean circuit C with n input bits, $k \in \mathbb{N}$, and a partition of the input variables of C into t sets $I_1, \ldots, I_t$.
Parameter:	k
Problem:	Decide whether there exists a set $J_1 \subseteq I_1$ of size k such that for all subsets $J_2 \subseteq I_2$ of size k there exists ... such that setting precisely the variables in $J_1 \cup \ldots \cup J_t$ to 'true' results in a satisfying assignment of C.

As in the case of the Polynomial Hierarchy, and unlike the case of the A-Hierarchy, it is known that the collapse of levels of the W[P]-Hierarchy would propagate upward:

▶ **Fact 7** (Corollary 17 of [4]). *If for any $t \geq 1$, $\Sigma_{t+1}^{[P]} = \Sigma_t^{[P]}$, then* $\mathsf{W[P]H} = \Sigma_t^{[P]}$.

▶ **Definition 8** ([4]). An oracle ARAM has *parameter-bounded oracle access* to a parameterized oracle, if its access to the oracle is balanced and, furthermore, there is a computable function g such that on every input (x, k), the machine makes at most $g(k)$ oracle queries.

If $\mathcal{C}$ is a class that is defined in terms of ARAMs, we write $\mathcal{C}(O)_{para}$ to denote that $\mathcal{C}$ has parameter-bounded access to the oracle O. If $\mathcal{C} = \Sigma_t^{[P]}$, for some $t \geq 1$, we may also write $\mathcal{C}^O$ to mean $\mathcal{C}(O)_{para}$ (so $\mathsf{W[P]}^O = \mathsf{W[P]}(O)_{para}$).

3 The difficulty of separating W[P] from FPT

In this section we show that there is likely no shortcut to proving $\mathsf{FPT} \neq \mathsf{W[P]}$ via any finite number of separations of the form $\mathsf{P} \neq \mathsf{NP}[h(n) \log n]$. Due to space restrictions, the proofs of the theorems in this section can be found in the full version of the paper.

To prove the theorem, we need to construct an oracle relative to which two conditions hold simultaneously: the collapse of one pair of complexity classes and the separation of another. One approach to achieving this is to construct the oracle in stages, and to work towards one goal in the odd-numbered stages and towards the other in the even-numbered ones, while ensuring that the two constructions do not interfere with each other (see [3, Theorem 5.1] for one example of an application of this technique). However, this approach does not always work, and in this case it fails because one pair of classes is parameterized (specifically, it is not possible to computably list all FPT- or W[P]-machines, but this appears to be necessary in this type of staged construction). To overcome this obstacle, we use an idea of Allender [1], who constructs an oracle with two parts: the first part is designed so as to ensure that one pair of classes collapses *regardless of what the second part of the oracle is*; the second part can then be freely used in a diagonalization argument to separate the remaining pair of classes.

▶ **Theorem 9.** *For every polynomial-time-computable non-decreasing unbounded function h, there exists a computable oracle B such that*

$$\mathsf{P}^B \neq \mathsf{NP}[h(n) \log n]^B, \quad \textit{but} \quad \mathsf{W[P]}(B) = \mathsf{FPT}(B).$$

For this result we have used unrestricted oracle access to relativize the parameterized complexity classes, rather than the parameter-bounded type that we argued is natural for W[P] [4]. This is because restricting the oracle access to being balanced (or more) makes it possible to collapse even para-NP to FPT, with the classical separation unchanged. Thus we would get an oracle relative to which NP and P differ while para-NP and FPT coincide, which is clearly an artifact of the restrictions placed on the oracle access, since we know that, unrelativized, a collapse of para-NP to FPT is equivalent to a collapse of NP to P (see [11]).

It seems reasonable to expect that if W[P] collapses to FPT relative to some oracle, then so should any class W[t]. But to show that this is indeed the case, we first need to define oracle W[t]-machines. Recall that a W[t]-machine is similar to an A[t]-machine, but the numbers it guesses nondeterministically are placed in a special set of *guess registers*, to which the machine has only limited access [7] (see also [5]). Naturally, an oracle W[t]-machine should then have three sets of registers: the standard registers, guess registers (which the machine can not read from directly), and oracle registers. For such machines, the usual way to read from or write to the oracle registers is very limiting, because the machines' nondeterminism would only weakly be able to influence the query instances. For example, nondeterministically guessed numbers could not be written to the oracle registers. The interaction between the nondeterminism of such machines and their ability to form query instances can be strengthened without allowing the W[t]-specific restrictions to be circumvented. We achieve this by making the oracle registers *write-only*, and adding instructions that allow the machine to copy values from the guess registers to the oracle registers and to use numbers from the guess registers to address oracle registers. In this way, the machine can still not read the guessed numbers directly or use them in arithmetic computations, but can nevertheless use them for oracle queries. In many cases, this allows oracle W[t]-machines to match A[t]-machines in the way the oracle is used.

▶ **Definition 10.** An *oracle* W[t]*-machine* is a W[t]-machine that, in addition to the standard registers $r_0, r_1, \ldots$, and guess registers $g_0, g_1, \ldots$ (to which the machine has only restricted access), also possesses a set of oracle registers $o_0, o_1, \ldots$. The contents of oracle registers are never read from and are only affected by the following new instructions:

SO_MOVE - copy the contents of standard register r_0 to oracle register o_{r_1};
GO_MOVE - copy the contents of guess register g_{r_0} to oracle register o_{r_1};
ADDR_GO_MOVE - copy the contents of register r_0 to $o_{g_{r_1}}$;
OO_MOVE - copy the contents of o_{r_0} to o_{r_1}.

Additionally, the machine has a QUERY instruction that places either the value 0 or 1 into r_0, depending on whether the contents of the oracle registers at the time when the instruction is executed represent a 'no'- or a 'yes'-instance of the problem to which the machine has oracle access.

Note that for such machines, it again makes sense to speak of unrestricted, balanced, parameter-bounded, or tail-restricted oracle access. With the above definition in mind, we can now prove Corollary 11, where oracle access is unrestricted.

▶ **Corollary 11.** *For any function h as in Theorem 9, and the corresponding oracle B, we have that* $\mathsf{FPT}(B) = \mathsf{W}[1](B) = \mathsf{A}[1](B) = \mathsf{W}[2](B) = \mathsf{A}[2](B) = \ldots = \mathsf{W}[\mathsf{P}](B)$.

4 The structure of W[1] and its relation to FPT

Under the assumption that $\mathsf{P} \neq \mathsf{NP}$, it is meaningful to ask whether the relation between the two classes is the same as the one between FPT and W[1]. So far, we have seen evidence that the two parameterized classes are closer to each other in the sense that proving a separation between them is more difficult than proving $\mathsf{P} \neq \mathsf{NP}$. In this section we look at other ways in which W[1] is closer to FPT than NP is to P.

In this section, the definitions of classes in terms of RAMs are used, instances of problems and oracles query instances are encoded as integer sequences, and oracles are parameterized.

Is FPT *low for* W[1]*?*

Given that FPT is the class of tractable problems in parameterized complexity, and that P-oracles add no computational power to NP (or to any class $\mathsf{NP}[h(n)\log n]$), one might expect FPT to also be low for W[1]. It turns out, however, that allowing tail-nondeterministic machines to make even tail-restricted queries to an FPT-oracle can increase their computational strength to that of W[P]. We prove this for A[1] first, since this machine model is more easily relativizable.

▶ **Theorem 12.** $\mathsf{A}[1]^{\mathsf{FPT}} = \mathsf{W}[\mathsf{P}]$. *Therefore,* FPT *is low for* A[1] *if and only if* $\mathsf{W}[\mathsf{P}] = \mathsf{A}[1]$ *and the* W*-Hierarchy collapses to its first level.*

Proof. We have that $\mathsf{A}[1]^{\mathsf{FPT}} \subseteq \mathsf{A}[1](\mathsf{FPT})_{bal} \subseteq \mathsf{W}[\mathsf{P}](\mathsf{FPT})_{bal} = \mathsf{W}[\mathsf{P}]$, with the final equality holding because a W[P]-machine can replace balanced oracle calls to FPT-problems by fpt-length computations.

To show that $\mathsf{W}[\mathsf{P}] \subseteq \mathsf{A}[1]^{\mathsf{FPT}}$, we define the following problem:

p-WSatCircuit-with-assignment	
Input:	A circuit C with n inputs, $k \in \mathbb{N}$, and vector $v \in \{0,1\}^n$ of weight k.
Parameter:	k.
Problem:	Decide whether v is a satisfying assignment for C.

Since the output of a circuit can be computed in time polynomial in its size, the above problem is obviously in FPT[4]. Any problem $Q \in \mathsf{W}[\mathsf{P}]$ can be solved by some A[1]-machine A with tail-restricted access to p-WSatCircuit-with-assignment as an oracle: First, A reduces in fpt-time the input instance (x,k) to an instance (y,k') of p-WSatCircuit, where k' depends computably only on k. Let m be the number of input bits of the circuit encoded in y. If $m < k'$, A rejects, otherwise it writes y, 0^m, and $1^{k'}$ to its oracle registers, thus forming a valid instance of p-WSatCircuit-with-assignment, except that the assignment vector has weight 0. Now A enters the nondeterministic phase of its computation by guessing k' pairwise distinct integers $i_1, \ldots, i_{k'} \in [m]$. It then modifies the assignment vector in the oracle registers by changing the zeroes at positions $i_1, \ldots, i_{k'}$ of the vector 0^m to 1, queries the oracle, and accepts if the answer is 'yes', otherwise it rejects.

It is easy to see that what this machine actually does is nondeterministically guess a satisfying assignment of the p-WSatCircuit-instance, if one exists, and delegate the verification to the oracle. The trick here is that the all-zero assignment vector must be written to the oracle registers deterministically, because during the nondeterministic phase at the end of the computation there may not be enough time to do so. Then the machine only needs to change the vector at k' positions to obtain an assignment with the right weight, which takes only $O(k')$ steps with random access memory. ◀

Note that the proof that a W[P]-machine can simulate A[1]-machines with FPT-oracles only works if the oracle access of the A[1]-machines is tail-restricted or at least parameter-restricted. On the other hand, the proof that $\mathsf{A}[1]^{\mathsf{FPT}} \supseteq \mathsf{W}[\mathsf{P}]$ only requires tail-restricted oracle access. We regard this as further evidence (in addition to the results from [4]) that tail-restricted oracle access is the natural type to consider for the class A[1].

[4] In fact, this problem is clearly in P, meaning that we can actually prove the stronger statement $\mathsf{W}[\mathsf{P}] \subseteq \mathsf{A}[1]^{\mathsf{P}}$. However, we choose FPT instead of P because the instance with which the oracle is queried will be fpt-sized, and because it is more natural to have A[1]-machines query a parameterized oracle, rather than a classical one.

Since $\mathsf{A}[1] \subseteq \mathsf{W}[t] \subseteq \mathsf{W}[\mathsf{P}]$ holds for all $t \geq 1$, and by Theorem 12 we have that $\mathsf{A}[1]^{\mathsf{FPT}} = \mathsf{W}[\mathsf{P}] = \mathsf{W}[\mathsf{P}](\mathsf{FPT})_{tail}$, it seems reasonable to expect that $\mathsf{W}[t]^{\mathsf{FPT}} = \mathsf{W}[\mathsf{P}]$ holds for all t as well. In order to prove that this is indeed the case, we need to use oracle $\mathsf{W}[t]$-machines (Definition 10), with tail-restricted access to the oracle. Then the proof is based on a combination of ideas from the proofs of Corollary 11 and Theorem 12.

▶ **Corollary 13.** *For every $t \geq 1$ it holds that $\mathsf{W}[t]^{\mathsf{FPT}} = \mathsf{W}[\mathsf{P}]$.*

In [4] we showed that for every $t \geq 1$ the class $\mathsf{A}[t+1]$ can be obtained as $\mathsf{A}[1]^{O_t}$, where O_t is a specific $\mathsf{A}[t]$-complete oracle, but we also observed that $\mathsf{A}[1]^{\mathsf{FPT}}$ does not appear to be a subset of $\mathsf{A}[t]$ for any t. Theorem 12 provides support for this intuition by identifying $\mathsf{A}[1]^{\mathsf{FPT}}$ as a class which is not known or believed to be a subset of any class $\mathsf{A}[t]$.

▶ **Corollary 14.** *For every $t \geq 1$ we have that if $\mathsf{A}[t+1] = \mathsf{A}[1]^{\mathsf{A}[t]}$, then $\mathsf{W}[\mathsf{P}] \subset \mathsf{A}[t+1]$. In particular, if $\mathsf{W}[\mathsf{P}] \not\subset \mathsf{A}[2]$, we have that $\mathsf{A}[1]^{\mathsf{A}[1]} \neq \mathsf{A}[2]$.*

Corollary 14 shows that the above-mentioned oracle characterization of the A-Hierarchy from [4] can probably not be improved significantly: Although it may be possible to obtain $\mathsf{A}[t+1]$ by providing $\mathsf{A}[1]$ with other $\mathsf{A}[t]$-complete oracles, it is unlikely that O_t can be replaced by the entire class $\mathsf{A}[t]$, for the somewhat counter-intuitive reason that $\mathsf{A}[t]$ contains all *tractable* problems. More importantly, Corollary 14 implies, assuming $\mathsf{W}[\mathsf{P}] \not\subset \mathsf{A}[t+1]$ and that PH is proper, that each class $\mathsf{A}[t+1]$ is closer to the class $\mathsf{A}[t]$ than Σ_{t+1}^P is to Σ_t^P, in the precise sense that $\mathsf{A}[t+1] \subsetneq \mathsf{A}[1]^{\mathsf{A}[t]}$, whereas $\Sigma_{t+1}^P = \mathsf{NP}^{\Sigma_t^P}$. The use of a highly restricted type of oracle access for the parameterized classes can only make this conclusion more legitimate.

A weak downward separation theorem for the A-Hierarchy.

It is a long-standing open problem whether the collapse of any class $\mathsf{A}[t+1]$ to $\mathsf{A}[t]$ would cause all higher levels of the A-Hierarchy to coincide with $\mathsf{A}[t]$ (or, equivalently, whether a separation of two classes $\mathsf{A}[t+1]$ and $\mathsf{A}[t]$ would imply that all levels below $\mathsf{A}[t]$ are distinct, whence the name "downward separation"). Given the similarities with PH, one might expect such a theorem to hold for the A-Hierarchy as well. Nevertheless, the proof of the downward separation theorem for the Polynomial Hierarchy does not appear to carry over directly to the parameterized setting. So far, the best result in this direction has been a theorem of Chen *et al.* [7], who showed that $\mathsf{W}[\mathsf{P}] = \mathsf{FPT}$ implies $\mathsf{FPT} = \mathsf{A}[1] = \mathsf{A}[2] = \ldots$. This result is already non-trivial, since $\mathsf{A}[t]$ is not known to be a subset of $\mathsf{W}[\mathsf{P}]$ for $t > 1$, and can be viewed as a parameterized version of $\mathsf{P} = \mathsf{NP} \Rightarrow \mathsf{PH} = \mathsf{P}$, except that the stronger collapse $\mathsf{W}[\mathsf{P}] = \mathsf{FPT}$ is required instead of $\mathsf{A}[1] = \mathsf{FPT}$ (in fact, the proof of the parameterized theorem in [7] is adapted from the proof of the corresponding classical theorem). Previously it was not known whether assuming a weaker collapse, for example $\mathsf{W}[\mathsf{P}] = \mathsf{A}[1]$, might also suffice to prove that $\forall t \geq 1 : \mathsf{A}[t] = \mathsf{A}[t+1] \Rightarrow \text{A-Hierarchy} = \mathsf{A}[t]$. In what follows we prove such a theorem.

Let $\mathsf{A}[1]_c$ be the class of parameterized problems Q such that there exists an $\mathsf{A}[1]$-machine that solves any instance (x, k) of Q in a number of steps depending only on k. This subclass of $\mathsf{A}[1]$ contains the problems that can be solved by $\mathsf{A}[1]$-machines *without* the need for a precomputation that runs in fpt-time. It is provably not closed under fpt-reductions, but contains many important $\mathsf{W}[1]$-complete problems, provided that the input is given in an appropriate format. For example, p-IndependentSet $\in \mathsf{A}[1]_c$, if the input graph is given in the form of an adjacency matrix, because then an $\mathsf{A}[1]$-machine can first guess k vertices (recall that a nondeterministic RAM can guess an integer between 1 and n in a single step; see Section 2.1) and use its random access memory to verify in $O(k^2)$ steps that none of the edges between two guessed vertices are in the graph. One can similarly show that p-ShortTMAcceptance and other $\mathsf{W}[1]$-complete problems are in $\mathsf{A}[1]_c$.

If W[P] were to collapse to A[1], then the W[P]-complete problem p-WSatCircuit would also be A[1]-complete, and therefore it would seem reasonable to expect that it is also in $\mathsf{A}[1]_c$, given an appropriate, efficiently computable encoding of the input. Thus, p-WSatCircuit $\in \mathsf{A}[1]_c$ seems only slightly less likely than $\mathsf{W}[\mathsf{P}] = \mathsf{A}[1]$ (although, strictly speaking, both p-WSatCircuit $\in \mathsf{A}[1]_c$ and p-WSatCircuit $\in \mathsf{FPT}$ (used by Chen *et al.* [7]) are strictly stronger assumptions than p-WSatCircuit $\in \mathsf{A}[1]$, and probably mutually incomparable). Under this assumption, we can prove the following:

▶ **Theorem 15.** *Assume that p-WSatCircuit $\in \mathsf{A}[1]_c$, meaning that there exists an* A[1]*-machine which solves any instance (x, k) of p-WSatCircuit in a number of steps depending computably on k alone. Then $\forall t \geq 1$ we have that $\mathsf{A}[t] = \mathsf{A}[t+1] \Rightarrow (\forall u \geq 1 : \mathsf{A}[t] = \mathsf{A}[t+u])$.*

Proof. We show that under the first assumption in the theorem statement, we have for every $t \geq 1$ that $\mathsf{A}[t+1] = \Sigma_{t+1}^{[P]}$. Since we already have a downward separation theorem for W[P]H (Fact 7), it follows that the desired conclusion holds for the A-Hierarchy.

First, we have for every $t \geq 1$ that $\Sigma_{t+1}^{[P]} \subseteq \mathsf{A}[t]^{p\text{-WSatCircuit}}$, by a similar proof as that of Theorem 12: To solve a problem $Q \in \Sigma_{t+1}^{[P]}$, an $\mathsf{A}[t]^{p\text{-WSatCircuit}}$-machine will first compute a reduction to the canonical $\Sigma_{t+1}^{[P]}$-complete problem p-AWSatCircuit$_{t+1}$, and, if t is odd, modify the resulting circuit so that its output is flipped. The machine then uses its t-alternating nondeterminism to guess the variables to set to 1 in the first t sets of the partition of the circuit's inputs, and hardwires this partial assignment into the circuit. The result is an instance of p-WSatCircuit, which can be solved with a single query to the oracle, and the oracle A[t]-machine now outputs the oracle's answer if t is odd, otherwise it outputs the opposite answer. It is easy to verify that this solves the problem Q.

Finally, we outline the proof that $\mathsf{A}[t]^{p\text{-WSatCircuit}} \subseteq \mathsf{A}[t+1]$, under the assumption that the algorithm for p-WSatCircuit mentioned in the theorem statement exists. This inclusion is proved in the same manner as $\mathsf{A}[1]^{p\text{-MC}(\Sigma_t[3])} \subseteq \mathsf{A}[t+1]$ [4, Theorem 13], which is itself a parameterized version of the proof of the well-known fact that $\mathsf{NP}^{\Sigma_t\mathrm{SAT}} \subseteq \Sigma_{t+1}^P$ (see [2, Section 5.5]). An $\mathsf{A}[t+1]$-machine can first perform the deterministic part of the oracle A[t]-machine's computation, and then use its $(t+1)$-alternating nondeterminism to guess the answers to the subsequent oracle queries of the simulated machine (existentially), all of its t-alternating nondeterministic guesses, as well as (suitably quantified) witnesses for the query instances. Oracle queries are then replaced by computations in which the guessed witnesses are used instead of nondeterministic guesses. The fact that evaluations of p-WSatCircuit-queries can be performed in this manner, is due to the assumption that this problem has a nondeterministic algorithm running in time dependent on k alone.

Since $\mathsf{A}[t+1] \subseteq \Sigma_{t+1}^{[P]}$ holds unconditionally, we conclude that the two classes are equal, which completes the proof. ◀

5 Level-by-level relativized separations of the W- and the A-Hierarchy

In this section we give oracle-based evidence that the main parameterized hierarchies do not collapse in unforeseen ways. We start by constructing a single oracle relative to which the inclusion of every W[t] in A[t] is strict, except for the first level. In fact, we accomplish this by proving the strongest possible relativized separation between co-nondeterminism and (existential) nondeterminism in the parameterized setting: the weakest co-nondeterministic class with tail-restricted oracle access, against the strongest nondeterministic class with unrestricted oracle access.

The proofs of the theorems in this section are based on standard diagonalization arguments that have been adapted to the parameterized setting, and can be found in the full version of the paper.

▶ **Theorem 16.** *There exists a computable oracle O such that* $\mathsf{co\text{-}A}[1]^O \not\subset \mathsf{para\text{-}NP}(O)$.

Since $\mathsf{co\text{-}A}[1]^O \subseteq \mathsf{A}[t]$ for all $t \geq 2$, and $\mathsf{W}[t]^O \subseteq \mathsf{para\text{-}NP}(O)$ for all $t \geq 1$, we immediately get the next corollary. Note, however, that the oracle constructed here does not appear to separate $\mathsf{A}[1]$ from $\mathsf{W}[1]$, since the separating problem in $\mathsf{co\text{-}A}[1]^O \setminus \mathsf{para\text{-}NP}(O)$ is not in $\mathsf{A}[1]^O$. Had a separation of two coinciding classes occured, this would have made the conclusion of Theorem 16 much less convincing.

▶ **Corollary 17.** *There exists a computable oracle O such that for every $t \geq 2$,* $\mathsf{W}[t]^O \subsetneq \mathsf{A}[t]^O$.

Finally, we show that the W-Hierarchy is not likely to be contained in any finite level of the A-Hierarchy.

▶ **Theorem 18.** *There exists for each $t \geq 1$ a computable oracle O_t such that* $\mathsf{W}[t+1]^{O_t} \not\subset \mathsf{A}[t]^{O_t}$*, where both machines have tail-restricted access to O_t, but the* $\mathsf{W}[t]$*-machine has the stronger type of oracle access mentioned in Section 3.*

As mentioned in the introduction, each oracle O_t also separates the classes $\mathsf{W}[t]$ and $\mathsf{W}[t+1]$ in the relativized setting, since $\mathsf{W}[t]^{O_t} \subseteq \mathsf{A}[t]^{O_t}$ but $\mathsf{W}[t+1]^{O_t} \not\subset \mathsf{A}[t]^{O_t}$.

6 Conclusions

Our results, together with the previously known theorems mentioned in the introduction, strongly indicate that if the central conjecture of parameterized complexity theory holds at all, proving it may be hard *even under the additional assumption of a separation between arbitrarily-weakly-nondeterministic polynomial-time and* P (and, in particular, that $\mathsf{P} \neq \mathsf{NP}$). Of course, the same also applies to the nowadays "standard" conjecture ETH. Additionally, we have seen that $\mathsf{W}[1]$ and FPT are in some ways unexpectedly close, unless much of what is generally assumed in parameterized complexity theory (such as the W-Hierarchy not collapsing) is false. All of this suggests that the hardness of a problem for up to $\mathsf{W}[\mathsf{P}]$ should not be treated as strong evidence of intractability, at least not with a similar level of confidence as when NP-hardness is considered evidence of computational intractability.

References

1. E. Allender. Limitations of the upward separation technique. *Mathematical Systems Theory*, 24(1):53–67, 1991.
2. S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge, 2009.
3. R.V. Book, C.B. Wilson, and M. Xu. Relativizing time, space, and time-space. *SIAM J. Comput.*, 11(3):571–581, 1982.
4. R.C. Bottesch. Relativization and Interactive Proof Systems in Parameterized Complexity Theory. In *12th International Symposium on Parameterized and Exact Computation (IPEC 2017)*, volume 89, pages 9:1–9:12, 2018. URL: `http://drops.dagstuhl.de/opus/volltexte/2018/8571`.
5. J.F. Buss and T. Islam. Simplifying the Weft hierarchy. *Theoretical Computer Science*, 351(3):303–313, 2006.
6. L. Cai, J. Chen, R.G. Downey, and M.R. Fellows. On the structure of parameterized problems in NP. *Information and Computation*, 123:38–49, 1995.

7 Y. Chen, J. Flum, and M. Grohe. Machine-based methods in parameterized complexity theory. *Theoretical Computer Science*, 339:167–199, 2005.

8 R.G. Downey and M.R. Fellows. Fixed-parameter tractability and completeness III - Some structural aspects of the W hierarchy. In K. Ambos-Spies, S. Homer, and U. Schoning, editors, *Complexity Theory*, pages 166–191. Cambridge University Press, 1993.

9 R.G. Downey and M.R. Fellows. *Parameterized Complexity*. Springer, Berlin, 1999.

10 R.G. Downey and M.R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.

11 J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, Berlin, 2006.

12 L. Fortnow and M. Sipser. Are there interactive protocols for co-NP languages? *Information Processing Letters*, 28(5):249–251, 1988.

13 R. Imagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4):512–530, 2001.

14 D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–71, 2011.

15 C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

16 A. Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.