Parameterized Analogues of Probabilistic Computation

Ankit Chauhan and B.V. Raghavendra Rao

Department of Computer Science and Engineering,
Indian Institute of Technology Madras,
Chennai, India
{ankitch,bvrr}@cse.iitm.ac.in

Abstract. We study structural aspects of randomized parameterized computation. We introduce a new class W[P]-PFPT as a natural parameterized analogue of PP. Our definition uses the machine based characterization of the parameterized complexity class W[P] obtained by Chen et.al [TCS 2005]. We translate most of the structural properties and characterizations of the class PP to the new class W[P]-PFPT.

We study a parameterization of the polynomial identity testing problem based on the degree of the polynomial computed by the arithmetic circuit. We obtain a parameterized analogue of the well known Schwartz-Zippel lemma [Schwartz, JACM 80 and Zippel, EUROSAM 79].

Additionally, we introduce a parameterized variant of permanent, and prove its $\#W[1]$ completeness.

1 Introduction

Parameterized Complexity Theory provides a formal framework for finer complexity analysis of problems by allowing a parameter along with the input. It was pioneered by Downey and Fellows [11,10] two decades ago. Since then, it has revolutionized algorithmic research [20], and led to the development of several important algorithmic techniques.

Fixed Parameter Tractability (FPT) forms the central notion of tractability in Parameterized Complexity Theory. Here, any problem that is decidable in deterministic time $f(k)\mathsf{poly}(n)$ is deemed to be tractable, where k is the parameter and f any computable function. Several NP hard problems including the vertex cover problem are known to be tractable under this notion [14].

The W-hierarchy serves as the basis for all intractable problems in the parameterized world. W[1], the smallest member of W-hierarchy, consists of problems that are FPT equivalent to the p-clique problem [14]. The limit of W-hierarchy, W[P] encapsulates all problems solvable in non-deterministic $f(k)\mathsf{poly}(n)$ time using at most $g(k)\log n$ non-deterministic bits [7,14], where f and g are arbitrary computable functions.

There have been significant efforts towards understanding the structure of parameterized complexity classes in the last two decades. Specifically exact characterizations of the W-hierarchy and other related hierarchies are known [12]. (See also [14,11].)

Apart from non-deterministic computation, probabilistic computation serves as one of the crucial building blocks of Complexity Theory. Probabilistic complexity classes have been well studied in the literature and has been an active area of research for more

S. Ganguly and R. Krishnamurti (Eds.): CALDAM 2015, LNCS 8959, pp. 181–192, 2015.

than three decades. There are a significant number of parameterized algorithms that use randomization [11, Chapter 8]. Hence, development of randomized complexity classes in the parameterized framework is necessary to understand the use of randomization in the parameterized setting.

Müller [19,18] was the first to do a systematic development and study of parameterized randomization. He defined bounded error probabilistic parameterized classes such as $\mathsf{W[P]\text{-}BPFPT}$ and $\mathsf{W[1]\text{-}BPFPT}$. Further, he obtained amplification results and conditions for derandomization of these classes. Further, Müller [19] studied several parameterizations of the well known polynomial identity testing problem (ACIT) and obtained several hardness results as well has upper bounds in terms of the newly defined randomized classes.

We continue the line of research initiated by Müller [19] and study a parameterized variant of probabilistic computation with unbounded error and establish a relationship with the corresponding parameterized counting class.

It should be noted that almost all of the randomized FPT algorithms use randomness of the same magnitude as their running times. However, such an algorithm cannot be visualized as a non-deterministic algorithm with $f(k)\log n$ random bits, where $f(k)$ is an arbitrary computable function. This is in stark contrast to the classical setting, where every randomized algorithm with bounded error probability can also be seen as a non-deterministic algorithm with the same time bound. So it is desirable to have randomized FPT algorithms that use at most $O(f(k)\log n)$ random bits instead of $f(k)\mathsf{poly}(n)$ random bits. As a first step towards this we obtain such an algorithm for a suitable parameterization of ACIT.

Finally, following the recent developments in the parameterized complexity theory of counting problems [5,8,9], we develop a parameterized variants of the problems of computing permanent and determinant of a matrix.

Our Results. We make an attempt at understanding the relations between counting and probabilistic classes. We focus on a probabilistic analogue of the class $\mathsf{W[P]}$. Using the notion of k-restricted Turing machines [7], we introduce $\mathsf{W[P]\text{-}PFPT}$ as a parameterized variant of the probabilistic polynomial time (PP). As in the classical complexity setting, we establish a close connection between $\mathsf{W[P]\text{-}PFPT}$ and the counting class $\#\mathsf{W[P]}$ (Theorem 2). Further, we show that $\mathsf{W[P]\text{-}PFPT}$ is closed under complementation and symmetric differences. (Theorem 3 and Lemma 3.)

We consider the polynomial identity testing problems (ACIT) with the *syntactic degree* (See Section 2 for a definition) as a parameter. Using the construction of hitting set generators by Shpilka and Volkovich [22], we obtain what can be called as a parameterized analogue of the celebrated Schwartz-Zippel Lemma [21,24]. (Theorem 4.)

Finally, we introduce a parameterized variant of the permanent function $\mathsf{p\text{-}perm}$ and prove that it characterizes the class $\#\mathsf{W[1]}$. (Theorem 6.) Analogously, a variant of the determinant function ($\mathsf{p\text{-}det}$) and show that it is Fixed Parameter Tractable (Theorem 5).

2 Preliminaries

We include some of the definitions from Parameterized Complexity theory and Complexity theory here. For Parameterized Complexity, the notations in [11,14] are followed. Definitions of complexity classes can be found in e.g., [13,4].

A *parameterized* language is a set $P \subseteq \Sigma^* \times \mathbb{N}$, where Σ is a finite alphabet. If $(x, k) \in \Sigma^* \times \mathbb{N}$ is an input instance of a parameterized language, then x is referred to as the *input* and k as the *parameter*.

A *parameterized counting* problem is a pair (f, k), where $f : \Sigma^* \to \mathbb{N}$ is a counting function and k is the parameter and Σ is a finite alphabet. For notational convenience, we will denote a parameterized counting problem as a function $f : \Sigma^* \times \mathbb{N} \to \mathbb{N}$, where the second argument to f is considered as the parameter.

A parameterized language $P \subseteq \Sigma^* \times \mathbb{N}$ is said to be *fixed-parameter tractable* if there is an algorithm that given a pair $(x, k) \in \Sigma^* \times \mathbb{N}$, decides if $(x, k) \in P$ in at most $O(f(k)|x|^c)$ steps, where $f : \mathbb{N} \to \mathbb{N}$ is a computable function and $c \in \mathbb{N}$ is a constant.

Definition 1. FPT *denotes the class of all parameterized languages that are fixed parameterized tractable.*

A parameterized language L is said to be in RFPT (Randomized FPT) if there is a $f(k)\mathsf{poly}(n)$ time bounded randomized machine accepting L with bounded one-sided error probability. See [14,11] for more details.

Definition 2. *A k-*restricted machine *is a non-deterministic $g(k)\mathsf{poly}(n)$ time bounded Random Access Machine (RAM) that uses at most $f(k)$ non-deterministic words, where f and g are arbitrary computable functions. Here we assume that the word size is $O(\log n)$, where n is the length of the input.*

*A k-*restricted Turing machine *is a non-deterministic $g(k)\mathsf{poly}(n)$ time Turing machine that makes at most $f(k)\log n$ non-deterministic moves, where f and g are arbitrary computable functions.*

Definition 3. *A* tail *non-deterministic machine is a k-restricted machine in which all non-deterministic steps are among last $f(k)$ steps.*

W[P] is the class of all parameterized problems (Q, k) that can be decided by a k-restricted non-deterministic machine (for more details see chapter 3 in [14]). W[1] is the class of all parameterized problems (Q, k) that can be decided by *tail* non-deterministic machine (for more details see [7]).

For a non-deterministic machine M, let $\#\mathsf{acc}_M(x, k)$ and $\#\mathsf{rej}_M(x, k)$ respectively denote the number of accepting and rejecting paths of M on input (x, k). Define $\mathsf{gap}_M(x, k) {=} \#\mathsf{acc}_M(x, k) - \#\mathsf{rej}_M(x, k)$.

Definition 4. *[14] A parameterized counting function (f, k) over the alphabet Σ is in $\#$W[P] if there is a k-restricted non-deterministic machine M such that $f(x, k) = \#\mathsf{acc}_M(x, k)$.*

Definition 5. *A probabilistic k-restricted machine is a probabilistic $g(k)\mathsf{poly}(n)$ time bounded RAM that make at most $f(k)$ probabilistic moves, where f and g are some computable functions. Here we assume that one probabilistic move involves choosing a random word of $O(\log n)$ bits.*

A language L is said to be in W[P]-RFPT [19] if there is a k-restricted probabilistic machine such that $(x, k) \in L \implies Pr[M \text{ accepts } (x, k)] \geq 2/3$; and $x \notin L \implies$

$Pr[M$ rejects $(x,k)] = 0$. It should be noted that for a langauage in RFPT the number of random bits can be a big as $f(k)\mathsf{poly}(n)$ whereas for languages in $\mathsf{W[P]}$-RFPT it is bounded by $O(f(k)\log n$.

An *arithmetic circuit* C is a directed acyclic graph with labelling on the vertices as follows. Nodes of in-degree zero are called *input* gates and are labelled from $\{-1,1\} \cup \{x_1,\ldots,x_n\}$ where $x_1,\ldots,x_n$ are the input variables. The remaining gates are labelled $\times$ or $+$. An arithmetic circuit has exactly one gate of zero out-degree called the *output* gate. Every gate v in an arithmetic circuit can naturally be associated with a polynomial $p_v \in \mathbb{Z}[x_1,\ldots,x_n]$, where the polynomials associated at input nodes are either constants or variables. If $v = v_1 + v_2$ then $p_v = p_{v_1} + p_{v_2}$ and if $v = v_1 \times v_2$ then $p_v = p_{v_1} \times p_{v_2}$. The polynomial computed by the circuit C is the polynomial associated with its only output gate and is denoted by p_C. The size of an arithmetic circuit is the number of gates in it and is denoted by $\mathsf{size}(C)$.

We associate a number called the *syntactic degree* ($\mathsf{syntdeg}$)[1] with every gate of an arithmetic circuit C. For a leaf node v, $\mathsf{syntdeg}(v) = 1$. If $v = v_1 + v_2$ then $\mathsf{syntdeg}(v) = \max\{\mathsf{syntdeg}(v_1), \mathsf{syntdeg}(v_2)\}$ and if $v = v_1 \times v_2$ then $\deg(v) = \mathsf{syntdeg}(v_1) + \mathsf{syntdeg}(v_2)$. It should be noted that the degree of the polynomial computed by a circuit is bounded by its syntactic degree.

Remark 1. the parameter $d_\times$ introduced in [19] is closely related to $\mathsf{syntdeg}$, in fact $\mathsf{syntdeg} \le 2^{d_\times} \le 2^{\mathsf{syntdeg}}$.

In [3], Alon obtained a characterization for multivariate polynomials that are not identically zero known as the Combinatorial Nullstellensatz:

Proposition 1 (Combinatorial Nullstellensatz, [3]). *Let $P \in \mathbb{K}[x_1,\ldots,x_n]$ be a polynomial where for every $i \in [n]$, the degree of x_i is bounded by t. Let $S \subseteq \mathbb{K}$ be a finite set of size at least $t+1$, and $A = S^n$. Then $P \equiv 0 \iff P(a) = 0,\ \forall a \in A$.*

3 Probabilistic Computation

In this section, we develop a parameterized analogue of the classical complexity class PP. Our definition of $\mathsf{W[P]}$-PFPT is based on k-restricted probabilistic Turing machines.

Throughout this section unless otherwise stated, $f(k)$ denotes an arbitrary computable function, and $P(n,k) = f(k)\log n$. For an input x, we denote $n = |x|$.

Definition 6. *A parameterized language L is said to be in the class $\mathsf{W[P]}$-PFPT if there is a k-restricted probabilistic Turing machine M such that for any $(x,k) \in \Sigma^* \times \mathbb{N}$ we have,*

$$(x,k) \in L \Rightarrow \Pr[M \textit{ accepts } (x,k)] > \frac{1}{2}$$
$$(x,k) \notin L \Rightarrow \Pr[M \textit{ accepts } (x,k)] \le \frac{1}{2}$$

where the probabilities are over the random choices made by M.

[1] Syntactic degree is also known as the formal degree [16] and is a standard parameter for arithmetic circuits.

Without loss of generality, we assume $\Sigma = \{0, 1\}$.

In the classical setting, PP is known to have several characterizations based on, 1) difference between two $\#$P functions [15], 2) difference between the number of accepting and rejecting paths of a polynomial time bounded non-deterministic Turing machine [15], 3) logics based on majority quantifiers [17] and 4) large fan-in circuits with threshold gates [2]. We observe that all of the characterizations except (3) hold for W[P]-PFPT. However, it is not clear if the majority quantifier logical characterization of PP [17] translates to the parameterized setting.

Definition 7 (Diff-FPT, Gap-FPT). *A parameterized function $f : \Sigma^* \times \mathbb{N} \to \mathbb{Z}$ is said to be in* Diff-FPT *if there are two functions $g, h \in \#\mathsf{W[P]}$ such that $f(x, k) = g(x, k) - h(x, k)$.*

f is said to be in Gap-FPT *if there is a k-restricted TM M such that $f(x, k) = \#\mathsf{acc}_M(x, k) - \#\mathsf{rej}_M(x, k)$, $\forall (x, k) \in \Sigma^* \times \mathbb{N}$.*

Firstly, we observe that the two classes Gap-FPT and Diff-FPT coincide.

Lemma 1. Gap-FPT = Diff-FPT

Proof. To show Gap-FPT $\subseteq$ Diff-FPT: Let $f \in$ Gap-FPT, then there is a k-restricted M with $f(x, k) = \#\mathsf{acc}_M(x, k) - \#\mathsf{rej}_M(x, k)$. Let M' be a new machine that simulates M on input (x, k) and accepts if and only if M rejects (x, k). Then we have $f(x, k) = \#\mathsf{acc}_M(x, k) - \#\mathsf{acc}_{M'}(x, k)$. For the converse inclusion, let $f \in$ Diff-FPT, and M_1, M_2 be such that $f(x, k) = \#\mathsf{acc}_{M_1}(x, k) - \#\mathsf{acc}_{M_2}(x, k)$. Let M be a new machine: on input (x, k), Guess a non-deterministic bit $b \in \{0, 1\}$. Run M_1 if $b = 0$, M_2 otherwise. If $b = 0$ and M_1 accepts then accept. If $b = 1$, and M_2 accepts then reject. In all other cases guess another non-deterministic bit b' and accept if $b' = 1$ and reject otherwise. Then $\#\mathsf{acc}_M(x, k) - \#\mathsf{rej}_M(x, k) = \#\mathsf{acc}_{M_1}(x, k) - \#\mathsf{acc}_{M_2}(x, k) = f(x, k)$

Lemma 2. Gap-FPT *is closed under taking p-*bounded *summations and products, i.e., if $g_1, \ldots, g_{t(k)} \in$ Gap-FPT, then so are $g_1 + g_2 \cdots + g_{t(k)}$ and $g_1 \times g_2 \times \cdots \times g_{t(k)}$, where t is any computable function.*

Proof. The arguments here are straightforward adaptations of proofs from classical complexity. We include it here for completeness. For summation, we can construct a new machine M that first guesses $i \in [1, t(k)]$ and and runs the k-restricted machine for g_i on (x, k).

For product, we will show for the case when $t(k) = 2$. Let $f_1, f_2 \in$ Gap-FPT. Let M_1 and M_2 as the k-restricted machines such that $f_i(x, k) = \#\mathsf{acc}_{M_i}(x, k) - \#\mathsf{rej}_{M_i}(x, k)$, $1 \leq i \leq 2$. Let $\overline{M_i}$ be the machine that flips the answers of M_i. Let M be k-restricted machine defined as follows: On input (x, k) first simulate M_1 on (x, k). If M_1 accepts then run M_2 on (x, k) and accept if and only if M_2 does so. If M_1 rejects then run $\overline{M_2}$ on (x, k) and accept if and only if $\overline{M_2}$ does so. It can be seen that $f_1(x, k) f_2(x, k) = \#\mathsf{acc}_M(x, k) - \#\mathsf{rej}_M(x, k)$.

The above argument can be generalized to the case $t(k) \geq 2$.

Theorem 1. *Let L be a parameterized language. The following are equivalent:*

1. $L \in \mathsf{W[P]}\text{-}\mathsf{PFPT}$.
2. *There is a $k-$restricted Turing machine M such that,* $(x,k) \in L \iff \#\mathsf{accept}_M(x,k) - \#\mathsf{reject}_M(x,k) > 0$.
3. *There is a function $f \in \mathsf{Gap}\text{-}\mathsf{FPT}$ such that $(x,k) \in L \iff f(x,k) > 0$*
4. *There is a $B \in \mathsf{FPT}$ and $P(n,k) = f(k)\log n$ such that $(x,k) \in L$ if and only if* $|\{y \in \{0,1\}^{P(n,k)} \mid (x,y,k) \in B\}| \geq 2^{P(n,k)-1} + 1$.

Proof (Theorem 1). ($1 \Rightarrow 2$) Let $L \in \mathsf{W[P]}\text{-}\mathsf{PFPT}$. Let M be a k-restricted probabilistic machine for L. Then,

$$(x,k) \in L \Rightarrow Pr[\text{M accept}(x,k)] > \frac{1}{2} \Rightarrow \frac{\#\mathsf{accept}_M(x,k)}{\#\mathsf{accept}_M(x,k) + \#\mathsf{reject}_M(x,k)} > \frac{1}{2}$$
$$\Rightarrow \#\mathsf{accept}_M(x,k) - \#\mathsf{reject}_M(x,k) > 0$$

($2 \Rightarrow 3$) This directly follows from the definition of $\mathsf{Gap}\text{-}\mathsf{FPT}$.

($3 \Rightarrow 4$) Let $f \in \mathsf{Gap}\text{-}\mathsf{FPT}$ with $(x,k) \in L \iff f(x,k) > 0$, and M be a k-restricted machine with $f(x,k) = \mathsf{gap}_M(x,k)$. Let $P(n,k)$ be the number of non-deterministic bits used by M on an input of length n with parameter k. Then $\mathsf{gap}_M(x,k) > 0 \implies \#\mathsf{acc}_M(x,k) > 2^{P(n,k)}/2 = 2^{P(n,k)-1}$. Let

$$B = \{\langle x,y,k\rangle \mid M \text{ on the non-deterministic path defined by } y \text{ accepts } x.\}$$

Clearly, $B \in \mathsf{FPT}$ and $\#\mathsf{acc}_M(x,k) = |\{y \in \{0,1\}^{P(n,k)} \mid \langle x,y,k\rangle \in B\}|$. Thus $(x,k) \in L \implies |\{y \in \{0,1\}^{P(n,k)} \mid \langle x,y,k\rangle \in B\}| > 2^{P(n,k)-1}$.

($4 \Rightarrow 1$) Let L as given in 4. Let M be k-restricted machine that on input (x,k) guesses a string $y \in \{0,1\}^{P(n,k)}$ and accepts if and only if $\langle x,y,k\rangle \in B$. Then we have $x \in L \iff \#\mathsf{acc}_M(x,k) > 2^{P(n,k)-1} \iff Pr[M \text{ accepts } (x,k)] > 1/2$.

Similar to the case of PP, we observe that an FPT machine with oracle access to a function in $\#W[P]$ is equivalent to an FPT machine with a language in $\mathsf{W[P]}\text{-}\mathsf{PFPT}$ as an oracle.

Theorem 2. $\mathsf{FPT}^{\#\mathsf{W[P]}} = \mathsf{FPT}^{\mathsf{W[P]}\text{-}\mathsf{PFPT}}$

Proof (Theorem 2). We show the containment in both the directions. We start with the easier direction, i.e., we show $\mathsf{FPT}^{\mathsf{W[P]}\text{-}\mathsf{PFPT}} \subseteq \mathsf{FPT}^{\#\mathsf{W[P]}}$.

Let $L \in \mathsf{FPT}^{\mathsf{W[P]}\text{-}\mathsf{PFPT}}$ and M be a deterministic oracle Turing machine that runs in time $f(k)\mathsf{poly}(n)$ and $A \in \mathsf{W[P]}\text{-}\mathsf{PFPT}$ with $L = L(M^A)$. We need to show that $L \in FPT^{\#W[P]}$. By Theorem 1, there are two parameterized functions $g, h : \{0,1\}^* \times k \to \mathbb{N}$ with $g, h \in \#\mathsf{W[P]}$ such that

$$(x,k) \in A \iff g(x,k) - h(x,k) > 0. \qquad (1)$$

Let $\gamma : \{0,1\}^* \times k \to \mathbb{N}$ where $\gamma(0x,k) = g(x,k)$, and $\gamma(1x,k) = h(x,k), \forall x \in \{0,1\}^*$. On strings of length 0 and 1, γ can be defined arbitrarily. We have $\gamma \in \#\mathsf{W[P]}$, since on input $y = ax$, with $a \in \{0,1\}$, the machine would run the machine for g if $a = 0$ and machine for h if $a = 1$.

We can simulate a query (y, k') made by the machine M to A by two queries to the function γ: (1) Query $(0y, k')$ and (2) $(1y, k')$, compute the difference of the values obtained and use (1) to decide the membership of (y, k') in A. Thus we can conclude $L \in \mathsf{FPT}^{\#\mathsf{W[P]}}$.

For the reverse containment, given a Turing machine M, let L_M be the language defined as : $L_M = \{((x, k, y) \in \Sigma^* \times \mathbb{N} \times \mathbb{N} \mid \#\mathsf{acc}_M(x, k) > y\}$

Claim. Let M be a k-restricted Turing machine, then $L_M \in \mathsf{W[P]}\text{-}\mathsf{PFPT}$.

Proof (Claim). Let M' be a Turing machine runs in FPT time computing function $t(x, k, y)$, that on input (x, k, y), produces exactly y accepting paths, where y is represented in binary, and $y \in [0, 2^{P(n,k)}]$. Clearly, M' is a k-restricted Turing machine, since it needs to use only $P(n, k)$ many non-deterministic bits. Thus the function $t(x, k, y) = y$ is in $\#\mathsf{W[P]}$. Let $f_M(x, k, y) = \#\mathsf{acc}_M(x, k) - y$. Then by Lemma 1 f_M is in $\mathsf{gapW[P]}$ and the claim now follows from Theorem 1. □

Let $L \in \mathsf{FPT}^{\#\mathsf{W[P]}}$, then there is a deterministic oracle Turing machine M' that runs FPT time, and a function $g \in \#\mathsf{W[P]}$ such that $L = M'^g$. Let M be a k-restricted Turing machine that uses at most $f(k) \log n$ non-deterministic steps such that $g(x, k) = \#\mathsf{acc}_M(x, k)$. We use the standard binary search technique to show that $g(x, k)$ can be computed using $O(kn)$ many queries to the language L_M.

```
Input $(x, k)$, oracle access to $L_M$. Output $g(x, k)$.
1. Initialize $p = P(|x|, k)$, $y = 2^p$.
2. Repeat steps 3 & 4 until $p \geq 0$
3. Query $(x, k, y)$ to the oracle; If YES, then set $b_p = 1$ and $y = y + 2^{p-1}$; Else set $b_p = 0$.
4. Set $p = p - 1$
5. Return $a = \mathsf{binary}(b_p b_{p-1} \dots b_0)$.
```

In the above $\mathsf{binary}(b_p b_{p-1} \dots b_0) = \sum_{i=0}^{p} 2^i b_i$. Clearly, the algorithm above runs in time $f(k)\mathsf{poly}(n)$, and hence computing g can be done in FPT with oracle access to $L_M \in \mathsf{W[P]}\text{-}\mathsf{PFPT}$. This concludes the inclusion in the converse direction. □

Theorem 3. $\mathsf{W[P]}\text{-}\mathsf{PFPT}$ *is closed under complementation.*

Proof can be found in the complete version of paper [6].

Lemma 3. $\mathsf{W[P]}\text{-}\mathsf{PFPT}$ *is closed under symmetric difference.*

Proof. The proof essentially follows the ideas in the classical setting [2]. Let $L_1, L_2 \in \mathsf{W[P]}\text{-}\mathsf{PFPT}$. By Theorem 1, we have $B_1, B_2 \in \mathsf{FPT}$ and $P(n, k) = f(k) \log n$ such that for any $x \in \{0, 1\}^*$, $k \in \mathbb{N}$ and $i \in \{1, 2\}$,

$$(x, k) \in L_i \iff |\{y_i \in \{0, 1\}^{P(n,k)} |(x, k, y_i) \in B_i\}| \geq 2^{P(n,k)-1} + 1$$

Using a construction similar to the one used in the proof of Theorem 3, we get parameterized languages B_1' and B_2', and a function $P'(n, k) = f'(k) \log n$ with the following property for $1 \leq i \leq 2$:

$$(x, k) \in L_i \implies |\{y_i \in \{0, 1\}^{P'(n,k)} |(x, k, y_i) \in B_i'\}| \geq 2^{P'(n,k)-1} + 1; \text{and}$$
$$(x, k) \notin L_i \implies |\{y_i \in \{0, 1\}^{P'(n,k)} |(x, k, y_i) \in B_i'\}| \leq 2^{P'(n,k)-1} - 1.$$

Let $a_1(x), a_2(x) \in \mathbb{Z}$ such that $|\{y \in \{0,1\}^{P(n,k)} \mid \langle x, y\rangle \in B'_i\}| = 2^{P(n,k)-1} + a_i(x)$ for $1 \leq i \leq 2$. Thus $|\{y \in \{0,1\}^{P(n,k)} \mid \langle x, y\rangle \notin B'_i\}| = 2^{P(n,k)-1} - a_i(x)$. For $x \in \Sigma^*$, let

$$\begin{aligned}\ell(x,k) &\overset{\triangle}{=} |S(x,k)|\\ &= |(2^{P'(n,k)-1} + a_1)(2^{P'(n,k)-1} - a_2)\\ &\quad + (2^{P'(n,k)-1} - a_1)(2^{P'(n,k)-1} + a_2)|\\ &= (2^{2P'(n,k)-1} - a_1 a_2),\end{aligned}$$

where $S(x,k) = \{\langle y_1, y_2\rangle \mid (< x, y_1 > \in B_1 \wedge \langle x, y_2\rangle \notin B_2) \vee (\langle x, y_1\rangle \notin B_1 \wedge \langle x, y_2\rangle \in B_2)\}$. Now, if $x \in L_1 \triangle L_2$ then either ($a_1 \geq 1$ and $a_2 < 0$) or ($a_1 < 0$ and $a_2 \geq 1$) then $\ell > 2^{2P(n,k)-1}$ and if $x \notin L_1 \triangle L_2$ then either both a_1 and a_2 are greater than equal to 1 or both are less than 1, and in both the cases $\ell \leq 2^{2P(n,k)-1}$. Let M' be a k-restricted Turing machine that on input (x,k) guesses two strings y_1 and y_2 of length $P'(n,k)$ each, and queries (x,k,y_i) to B'_i, $1 \leq i \leq 2$, accepts if and only if exactly one of the oracle answers is YES. It can be seen that $\#\mathsf{acc}_{M'}(x,k) = \ell(x,k)$. We conclude $L_1 \triangle L_2 \in \mathsf{W[P]\text{-}PFPT}$.

4 Polynomial Identity Testing

Müller [19] studied the Arithmetic Circuit Identity Testing (ACIT) problem with various parameters and obtained upper bounds as well as hardness results for each of the parameters considered. However none of the parameters considered in [19] seem adequate for developing a complexity theory for the parameterized probabilistic and counting classes along the lines of classical complexity classes.

Recall that, in ACIT we are given an arithmetic circuit C as an input and the task is to test if the polynomial computed by C is identically zero. We consider the degree of the polynomial computed by C as a parameter.

Problem 1 (p-acit). *Input*: Arithmetic circuit C, $\mathsf{syntdeg}(C) \leq k$.
Parameter: k.
Task: Test if the polynomial computed by C is identically zero.

Our main objective now is to show that $\mathsf{p\text{-}acit} \in \mathsf{W[P]\text{-}RFPT}$. However, it should be noted that this does not follow directly from the Schwartz-Zippel Lemma, since it would require $O(n \log k)$ random bits. So the challenge here is to reduce the number of random bits required to $f(k)\log n$. Towards this, we use a mapping defined by Shpilka and Volkovich [22] that reduces the number of variables from n to $2k$. Then we apply Alon's Combinatorial Nullstellensatz [3] to obtain what can be treated as a parameterized version of the Schwartz-Zippel lemma.

We begin with a few observations on polynomials of degree at most k. Let S be any finite subset of $\mathbb{K}$ that includes $0 \in \mathbb{K}$ and let $W_n^k(S)$ denote the set of all vectors in S^n with at most k non zero entries i.e, the set of all vectors of Hamming weight at most k.

Lemma 4. *Let f be an n-variate polynomial of degree at most k. Then*

$$f \equiv 0 \iff \forall a \in W_n^k(S) \;\; f(a) = 0,$$

where $S \subset \mathbb{K}$ has at least $k+1$ elements.

Proof. For simplicity, we denote $W_n^k(S)$ by W_n^k. The proof is by induction on n. For the base case, suppose $n \leq k$. Since individual degrees of each variable is bounded by k, by Proposition 1, we have $f \equiv 0 \iff f(a) = 0 \; \forall \; a \in S^n$, for an S with $|S| \geq k$.

For the induction step, let $n > k$, and $f(a) = 0 \; \forall \; a \in W_n^k$. For $i \in \{1, \ldots, n\}$, let $f_i = f|_{x_i=0}$, i.e., f substituted with $x_i = 0$. Note that each of the f_i is a degree k polynomial on at most $n-1$ variables, and $\forall a \in W_{n-1}^k \;\; f_i(a) = 0$. By the induction hypothesis, we have $f_i \equiv 0$, and hence x_i divides f. Repeating the argument for all $i \in [1, \ldots, n]$, we have $x_1 x_2 \cdots x_n$ divides f, and hence $\mathsf{deg}(f) \geq n > k$, a contradiction since $\mathsf{deg}(f) = k < n$. Thus we conclude $\forall a \in W_n^k \; f(a) = 0 \implies f \equiv 0$. The converse direction is trivially true.

We need a function introduced by Shpilka and Volkovich [22], that gives a map $G_k : \mathbb{K}[x_1, \ldots, x_n] \to \mathbb{K}[y_1, \ldots, y_{2k}]$ and serves as a non-identity preserving for a large class of polynomials. We observe that G_k also functions as a non-identity preserving map for the class of all n variate polynomials of degree at most k. We begin with the definition of the generator G_k.

Definition 8 (Shpilka-Volkovich Hitting Set Generator, [22]). *Let $a_1, \ldots, a_n$ be distinct elements in $\mathbb{K}$. Let $G_k^i \in \mathbb{K}[y_1, \ldots, y_k, z_1, \ldots, z_k]$ be the polynomial defined as follows:*

$$G_k^i(y_1, \ldots, y_k, z_1, \ldots, z_k) = \sum_{j=1}^{k} L_i(y_i) z_i, \;\; \textit{where} \;\; L_i(x) = \frac{\prod_{j \neq i}(x - a_j)}{\prod_{j \neq i}(a_i - a_j)}.$$

The generator G_k is defined as $G_k \triangleq (G_k^1, \ldots, G_k^n)$.

Lemma 5. *For any finite set $S \subset \mathbb{K}$, then $W_n^k(S) \subseteq \{(G_k^1(a), \ldots, G_k^n(a)) \mid a \in (S \cup \{a_1, \ldots, a_n\})^{2k}\}$.*

Proof. The proof essentially follows the arguments in [22]. We include a sketch here for the sake of completeness. Note that,

$$L_i(\alpha) = \begin{cases} 0 \;\; \alpha = a_j, \text{ if } j \neq i \\ 1 \text{ if } \alpha = a_i. \end{cases}$$

Thus if we set $y_\ell = a_i$, then the image of G_k^i contains z_i as a summand. By ensuring that $y_j, i \neq j$ gets some $a_\ell, i \neq \ell$, we get $G_k^i = z_i$. In this way we can obtain all vectors of Hamming weight k, by setting y_i's and z_i's accordingly. □

Combining Lemma 5 with Lemma 4 we have:

Lemma 6. *Let f be a polynomial of degree at most k. Then $f \equiv 0 \iff f(G_k) \equiv 0$.*

Theorem 4. p-acit *is in* W[P]-RFPT

Proof. By Lemma 6 p-acit reduces to testing identity of $2k$-variate polynomials of degree $O(nk)$ (since the polynomials L_i have degree n). Now applying the Schwartz-Zippel lemma [21,24], we obtain a randomized algorithm that uses $O(2k \log(nk))$ random bits and runs in time polynomial in n and k.

5 Parameterized Permanent vs Determinant

The determinant (det) permanent (perm) functions are defined as

$$\mathsf{det}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} \mathsf{sgn}(\sigma) a_{i,\sigma(i)} \tag{2}$$

$$\mathsf{perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)}, \tag{3}$$

where $A = (a_{i,j})_{1 \le i,j \le n} \in \mathbb{N}^{n \times n}$, S_n is the set of all permutations on n symbols and sgn is the sign function for permutations. It is known that, given an integer matrix A, computing A can be done in polynomial time (e.g., Gaussian Elimination method). In his celebrated paper, Valiant [23] showed that computing perm of an integer even for 0 or 1 matrix is complete for #P. Though there are several natural counting problems that characterize $\#W[1]$, it is desirable to have a parameterized variant of permanent so that we get access to the algebraic properties of the permanent function.

Naturally, we expect any parameterized variant of permanent to be a function of degree k in n^2 variables, where k is the parameter. One way to achieve this would be to restrict the summation given in (3) that move exactly k-elements. Formally, a permutation $\sigma \in S_n$ is said to be a *k-permutation*, if $|\{i \mid \sigma(i) \neq i\}| = k$. Let $S_{n,k}$ denote the set of all k-permutations on n symbols.

Definition 9. *Let k be a parameter. The parameterized determinant (*p-det*) and permanent (*p-perm*) functions of a matrix $A \in \mathbb{Z}^{n \times n}$ are defined as follows:*

$$\mathsf{p\text{-}det}(A, k) = \sum_{\sigma \in S_{n,k}} \prod_{i \neq \sigma(i)} \mathsf{sgn}(\sigma) a_{i\sigma(i)}$$

$$\mathsf{p\text{-}perm}(A, k) = \sum_{\sigma \in S_{n,k}} \prod_{i \neq \sigma(i)} a_{i\sigma(i)},$$

where k is a parameter. By abusing the notation, we also let p-perm *denote the problem of computing* p-perm *of an $n \times n$ matrix, where k is the parameter.*

Quite expectedly, p-det is FPT and p-perm can be shown to be $\#W[1]$ complete under fpt-reductions. We start with the tractability of p-det.

Theorem 5. p-det *on integer matrices is fixed parameter tractable.*

Proof. Let $A \in \mathbb{Z}^{n \times n}$, and k be the parameter. Let A' be the matrix obtained from A by replacing the diagonal entries in A by zeroes. Clearly $\mathsf{p\text{-}det}(A, k) = \mathsf{p\text{-}det}(A', k)$. Let x be a formal variable. Then $\mathsf{det}(xA')$ is a univariate polynomial of degree bounded by n, and the coefficient of x^k in $\mathsf{det}(xA')$ is equal to $\mathsf{p\text{-}det}(A, k)$. The value $\mathsf{p\text{-}det}(A, k)$ be recovered using the standard interpolation of univariate polynomials. □

Theorem 6. p-perm *on matrices in $\mathbb{N}^{n \times n}$ is $\#W[1]$ complete. The hardness holds even in the case of 0-1 matrices.*

Proof. It is known that counting k-matchings in a bipartite graph is complete for $\#\mathsf{W}[1]$ even in the weighted case [9]. We prove a parameter preserving equivalence between p-perm and the problem of counting k-matchings in a bipartite graph which completes the proof. For the upper bound, we give a reduction from p-perm to counting k-matchings in a bipartite graph. For a given matrix $A \in \mathbb{N}^{n\times n}$ define the matrix A' by setting the diagonal entries of A to zero, i.e., $A'[i,j] = A[i,j]$ if $i \neq j$ and $A'[i,i] = 0$, $1 \leq i \leq n$. Note that $\mathsf{p\text{-}perm}(A) = \mathsf{p\text{-}perm}(A')$. Every k-permutation of $\{1,\ldots,n\}$ corresponds to a matching of size k in the bipartite graph G' with A' is the bipartite adjacency matrix. Thus $\mathsf{p\text{-}perm}(A')$ =the sum of weights of k-matchings in G'.

For the hardness we give a reduction in the reverse direction, i.e., a parameter preserving reduction from counting the number of k-matchings in a bipartite graph $G = (U,V,E)$ to computing p-perm of an integer matrix.

Let $G = (U,V,E)$ be a bipartite graph. Without loss of generality, assume that $U = V = \{1,\ldots,n\}$. Construct a new bipartite graph $G' = (U',V',E')$ with $U' = V' = \{1,\ldots,2n\}$. For every edge of the form $(i,j) \in E, i \neq j$, G' contains the edge $(i,j) \in E'$. For edges of the form $(i,i) \in E$, G' contains the edge $(i, n+i) \in E'$. Note that the vertices $n+1,\ldots,2n$ in U' are isolated vertices.

Note that the number of matchings in G and those in G' of a given size k are equal. Let A' be the bipartite adjacency matrix of G'. Every k-permutation of $[2n]$ that contributes a non-zero value to $\mathsf{p\text{-}perm}(A')$ corresponds to a matching of size k in G'. Moreover, none of the k-matchings in G' will have an edge of the form $(i,i), i \in [2n]$. Thus, $\mathsf{p\text{-}perm}(A',k) = \#$matchings of size k in G'. This completes the proof. □

6 Conclusions

We have studied parameterized variants of probabilistic computation. We hope that our definition of W[P]-PFPT leads to further developments in the structural aspects of probabilistic and counting complexities in the parameterized world. Further, W[P]-PFPT might be useful in defining a parameterized variant of the Counting Hierarchy (CH) which could in turn have implications to parameterized complexity of numerical and algebraic computation [1]. Though definition of a parameterized CH based on W[P]-PFPT is straightforward, the usefulness of such a definition would rely on W[P]-PFPT being closed under intersection, which is not known currently.

Further, we believe any fixed parameter tractable randomized algorithm should naturally place the problem in W[P]. One way to achieve this is to obtain randomized FPT algorithms that use at most $O(f(k)\log n)$ random bits. As a first step towards this direction, we introduce a natural parameterization to the polynomial identity testing for which we obtain such an algorithm. We hope our observations will lead to further development of randomness efficient parameterized algorithms.

Acknowledgements. We thank anonymous reviewers for their comments on an earlier version of this paper which helped in improving the presentation of the article.

References

1. Allender, E., Bürgisser, P., Kjeldgaard-Pedersen, J., Miltersen, P.B.: On the complexity of numerical analysis. SIAM J. Comput. 38(5), 1987–2006 (2009)
2. Allender, E., Wagner, K.W.: Counting hierarchies: Polynomial time and constant. Bulletin of the EATCS 40, 182–194 (1990)
3. Alon, N.: Combinatorial nullstellensatz. Combinatorics, Problem and Computing 8 (1999)
4. Arora, S., Barak, B.: Computational Complexity: A Modern approach. Cambridge Univeristy Press (2009)
5. Bläser, M., Curticapean, R.: Weighted counting of k-matchings is #W[1]-hard. In: IPEC, pp. 171–181 (2012)
6. Chauhan, A., Rao, B.V.R.: Parameterized analogues of probabilistic computation. CoRR, abs/1409.7790 (2014)
7. Chen, Y., Flum, J., Grohe, M.: Machine-based methods in parameterized complexity theory. Theor. Comput. Sci. 339(2-3), 167–199 (2005)
8. Curticapean, R.: Counting matchings of size k is $\sharp$W[1]-hard. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M., Peleg, D. (eds.) ICALP 2013, Part I. LNCS, vol. 7965, pp. 352–363. Springer, Heidelberg (2013)
9. Curticapean, R., Marx, D.: Complexity of counting subgraphs: only the boundedness of the vertex-cover number counts. CoRR, abs/1407.2929 (2014); to Appear in FOCS 2014
10. Downey, R.G., Fellows, M.R.: Fixed-parameter intractability. In: Structure in Complexity Theory Conference, pp. 36–49 (1992)
11. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer (1997)
12. Downey, R.G., Fellows, M.R., Regan, K.W.: Parameterized circuit complexity and the W hierarchy. Theor. Comput. Sci. 191(1-2), 97–115 (1998)
13. Du, D.-Z., Ko, K.-I.: Theory of Computational Complexity. Springer (2000)
14. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer (2008)
15. Fortnow, L.: Counting complexity. In: Hemaspaandra, L., Selman, A. (eds.) Complexity Theory Retrospective II, pp. 81–107 (1997)
16. Kayal, N., Saha, C., Saptharishi, R.: A super-polynomial lower bound for regular arithmetic formulas. In: STOC, pp. 146–153 (2014)
17. Kontinen, J.: A logical characterization of the counting hierarchy. ACM Trans. Comput. Log. 10(1) (2009)
18. Müller, M.: Parameterized derandomization. In: Grohe, M., Niedermeier, R. (eds.) IWPEC 2008. LNCS, vol. 5018, pp. 148–159. Springer, Heidelberg (2008)
19. M üller, M.: Parameterized Randomization. PhD thesis, Albert-Ludwigs-Universität Freiburg im Breisgau (2008)
20. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford Lecture Series in Mathematics and Its Applications, vol. 31. Oxford University Press (2006)
21. Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. J. ACM 27(4), 701–717 (1980)
22. Shpilka, A., Volkovich, I.: Improved polynomial identity testing for read-once formulas. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX and RANDOM 2009. LNCS, vol. 5687, pp. 700–713. Springer, Heidelberg (2009)
23. Valiant, L.G.: The complexity of computing the permanent. Theor. Comput. Sci. 8, 189–201 (1979)
24. Zippel, R.: Probabilistic algorithms for sparse polynomials. In: Ng, K.W. (ed.) EUROSAM 1979 and ISSAC 1979. LNCS, vol. 72, pp. 216–226. Springer, Heidelberg (1979)