RAIRO-Theor. Inf. Appl. 45 (2011) 197–223
DOI: 10.1051/ita/2011007

Available online at:
www.rairo-ita.org

ON THE PARAMETERIZED COMPLEXITY OF APPROXIMATE COUNTING

J. Andrés Montoya[1]

Abstract. In this paper we study the parameterized complexity of approximating the parameterized counting problems contained in the class $\#W[P]$, the parameterized analogue of $\#P$. We prove a parameterized analogue of a famous theorem of Stockmeyer claiming that approximate counting belongs to the second level of the polynomial hierarchy.

Mathematics Subject Classification. 68Q15, 68Q17.

1. Introduction

In this paper we analyze the parameterized complexity of approximating the counting problems in the class $\#W[P]$, the parameterized analogue of the class $\#P$. We compare those problems with the problems contained in the parameterized polynomial hierarchy. This way of analyzing the computational complexity of approximating hard counting problems resembles Stockmeyer's analysis [12] of the complexity of approximating $\#SAT$. Actually, in this paper we have tried to obtain, and we have obtained, a parameterized analogue of Stockmeyer's theorem.

1.1. Parameterized complexity

Parameterized complexity theory [3,5] provides a framework for a refined analysis of hard algorithmic problems.

Classical complexity theory [11] analyses problems by the amount of a resource, usually time or space, that is required by algorithms solving them. The amount of the resource required is measured as a function of the size of the input. Measuring

Keywords and phrases. Computational complexity, parameterized complexity, counting problems, approximate counting.

[1] Escuela de Matemáticas, Universidad industrial de Santander, Spain;
amontoyaa@googlemail.com; juamonto@uis.edu.co

Article published by EDP Sciences

complexity only in terms of the input size means ignoring any structural information about the input instances in the resulting complexity theory. Sometimes, this makes problems appear harder than they typically are. Parameterized complexity theory measures complexity not only in terms of the input size, but in addition in terms of a parameter, which is a numerical value that may depend on the input in an arbitrary way. The main intention of this theory is to address complexity issues in situations where we know that the parameter is comparatively small.

A good example is the problem of evaluating database queries. From the classical point of view this problem is tractable only in very restrictive cases, (the evaluation of conjunctive queries is already NP hard!). If one review the hardness proofs for the Database evaluation problem, it is easy to note that it is necessary to consider cases where the size of the query non trivially depends on the size of the database. In real life, databases are huge and queries are small. It suggests that we can consider the size of the query as a parameter and measure the complexity of the problem in terms of two independent quantities, database size and query size, if we want to obtain something new we have to consider a new (parameterized) notion of tractability.

The central notion of parameterized complexity theory is *fixed parameter tractability.* This relaxes the classical notion of tractability, polynomial time solvability, by admitting algorithms whose nonpolynomial behavior is restricted only by the parameter, in addition the theory provides us with a *parameterized intractability theory* allowing us to prove the intractability of certain problems by classifying them into parameterized complexity classes by means of suitable parameterized reductions.

1.2. Counting complexity

A typical class of computational problems is the class of counting problems. Counting problems are at least as hard as decision problems: if we can count the number of solutions we can decide if there exists at least one solution. Counting complexity, the complexity analysis of counting problems, was developed by Valiant with a series of ground breaking articles published in 1979 [14,15]. Valiant proved that some counting problems are harder than expected, he proved that the problem of counting the number of perfect matchings in a graph is $\#P$ complete. This is surprising because the corresponding decision problem, the problem of deciding if a graph has at least one perfect matching, belongs to P. The big surprise came next when Toda [13] proved that any problem in the polynomial hierarchy can be reduced to any $\#P$ complete problem, and it implies that any problem in the polynomial hierarchy can be reduced to the problem of counting the number of perfect matchings. Thus, we can conclude:

(1) Hard counting problems are much more difficult than the corresponding decision problems (if the polynomial hierarchy does not collapse).
(2) There are counting problems which are highly intractable, although the corresponding decision problems are tractable (even trivial).

When we cope with counting problems we have the following alternative: we can try to compute approximate solutions instead of computing exact solutions. Approximating a counting problem is easier than computing exact solutions to the problem. Stockmeyer proved [12] that approximating the problem $\#SAT$ can be done in probabilistic polynomial time if oracle access to the decision problem SAT is provided, from this theorem Stockmeyer obtained as a corollary that probabilistic approximate counting belongs to the second level of the polynomial hierarchy.

Parameterized counting complexity is not yet a mature theory, there are few works on the topic [2,4,8,10] and no structural theorems like the one of Toda. Muller [10] proved a parameterized analogue of Stockmeyer theorem, he proved that any problem in $\#W[P]$ can be approximated in randomized fpt time, if oracle access to $W[P]$ is provided. This result does not imply that parameterized approximate counting belongs to the parameterized polynomial hierarchy given that it is unknown if this hierarchy is closed under parameterized Turing reductions (actually, it is very unlikely that this hierarchy is closed under this type of reductions).

In this work we prove that approximate counting belongs to the parameterized polynomial hierarchy. We prove that the parameterized complexity of approximating the problems in $\#W[P]$ can be identified with the parameterized complexity of a family of parameterized gap problems which is contained in $BP \cdot \exists \cdot FPT$, the parameterized analogue of the Arthur–Merlin Class. Then, we prove a parameterized analogue of *Lautemann-Sipser theorem* which implies that $BP \cdot \exists \cdot FPT$ is included in the second level of the parameterized polynomial hierarchy.

1.3. Organization of the paper

The paper is organized into five sections. In section two we introduce the basic concepts of parameterized complexity theory. Section 3 is divided into three subsections. In Section 3.1 we introduce some parameterized operators, which are analogous to the classical operators $\forall, \exists$ and BP, those parameterized operators allow us to define new parameterized classes from old ones. Using operators $\forall$ and $\exists$ we define a hierarchy of parameterized classes analogous to the polynomial hierarchy. The BP operator allow us to define probabilistic parameterized classes. In Section 3.2 we introduce most of the hashing machinery that we will use in the proofs of our main results. In Section 3.3 we present a randomized fpt algorithm that can be used to approximate any parameterized counting problem in the class $\#W[P]$. The algorithm, based on hashing techniques, requires access to a $W[P]$ oracle. In Section 4 we study the complexity of approximating, within a constant range, the counting problems in the class $\#W[P]$. To this end we introduce a family of parameterized gap problems. Moreover, we introduce the parameterized Arthur-Merlin class $BP \cdot \exists \cdot FPT$, and we prove that the parameterized gap problems introduced in this section belong to this class. In Section 5 we prove a parameterized analogue of Lautemann-Sipser theorem which implies

that $BP \cdot \exists \cdot FPT$ is included in the second level of the parameterized polynomial hierarchy, we obtain as a corollary our parameterized Stockmeyer's theorem claiming that approximate counting belongs to second level of the same hierarchy.

2. A technical preface

In this section we introduce the basic definitions of *parameterized complexity theory*, more detailed information can be found in [3] and [5].

Notation 2.1. Σ^* *is the set of finite 0-1 words.*

Definition 2.2. A parameterized problem is a subset of $\Sigma^* \times \mathbb{N}$.

Definition 2.3. Given $x \in \{0,1\}^n$, the *Hamming weight* of x is equal to

$$|\{i \leq n : x_i = 1\}|.$$

An important example of parameterized problem is the problem $p\text{-}WSAT\,(CIRC)$ defined below.

Fact 2.4. *(p-$WSAT\,(CIRC)$: parameterized circuit satisfiability)*

- *Input: (C,k), where C is a boolean circuit and $k \in \mathbb{N}$.*
- *Parameter: k.*
- *Problem: decide if there exists a satisfying assignment of C whose Hamming weight is equal to k.*

The first important definition is the definition of efficient algorithm. Efficient algorithms will be called fpt algorithms.

Definition 2.5. An fpt algorithm is an algorithm $\mathcal{M}$ whose running time, on input (x,k), is upperbounded by $f(k) \cdot p(|x|)$, where f is some computable function and $p(X)$ is some polynomial.

We can use the notion of efficient algorithms to define a suitable notion of parameterized feasible problems. To this end, we define the parameterized class FPT whose elements are the parameterized feasible problems.

Definition 2.6. The parameterized class FPT is the class of parameterized problems that can be decided using an fpt algorithm.

The next important definition is the notion of reducibility that will be used throughout the paper.

Definition 2.7. Given L, L^* two parameterized languages, L is fpt many-one reducible to L^*, ($L \preceq_{fpt} L^*$ for short), if and only if there exists an fpt algorithm $\mathcal{M}$ such that, on input $(x,k) \in \Sigma^* \times \mathbb{N}$, algorithm $\mathcal{M}$ computes a pair (x',k') that satisfies

(1) $k' \leq g(k)$ for some computable function g.
(2) $(x,k) \in L$ if and only if $(x',k') \in L^*$.

The following is a technical definition that will be used to define the parameterized class $W[P]$, the parameterized analogue of NP.

Definition 2.8. Given L a parameterized problem we have that

$$\langle L \rangle_{fpt} := \{L^* : L^* \preceq_{fpt} L\}.$$

Definition 2.9. $W[P] := \langle p - WSAT(CIRC) \rangle_{fpt}$.

The parameterized class $W[P]$ has a good machine characterization.

Definition 2.10. A $W[P]$ restricted Turing machine $\mathbb{M}$ is a nondeterministic Turing that satisfies:

(1) There exist a computable function f and a polynomial $p(X)$ such that, on every run of $\mathbb{M}$ with input (x,k), the running time of $\mathbb{M}$ is upperbounded by $f(k) \cdot p(|x|)$.
(2) There exists a computable function g such that, on every run of $\mathbb{M}$ with input (x,k), machine $\mathbb{M}$ guesses at most $g(k) \cdot \log(|x|)$ nondeterministic bits.

Theorem 2.11. *$L \in W[P]$ if and only if there exists a $W[P]$ restricted Turing machine $\mathbb{M}$ that decides L.*

A proof of this theorem can be found in [5], Thm. 3.14. Our aim is to analyze the parameterized complexity of some parameterized counting problems, to this end we introduce the notion of parameterized counting problem and a suitable notion of reducibility between parameterized counting problems.

Definition 2.12. A parameterized counting problem is a function $h : \Sigma^* \times \mathbb{N} \to \mathbb{N}$.

Definition 2.13. Given h, h^* two parameterized counting problems, h is parsimonious reducible to h^*, ($h \preceq_{par} h^*$ for short), if and only if there exists an fpt algorithm $\mathcal{M}$ such that, on input (x,k), algorithm M computes a pair (x',k') that satisfies:

(1) $k' \leq g(k)$ for some computable function g.
(2) $h((x,k)) = h^*((x',k'))$.

The following is a technical definition that will be used to define the class $\#W[P]$, the parameterized analogue of the counting class $\#P$.

Definition 2.14. Given a parameterized counting problem h we have that

$$\langle h \rangle_{par} := \{h^* : h^* \preceq_{par} h\}.$$

We define the parameterized counting class $\#W[P]$ as the closure under parameterized parsimonious reductions of a suitable parameterized analogue of the problem $\#SAT$, (it is well known that we can define the class $\#P$ as the closure under parsimonious reductions of the counting problem $\#SAT$).

Definition 2.15. (p-$\#WSAT\,(CIRC)$: parameterized counting of satisfying assignments)

- Input: (C, k), where C is a boolean circuit and $k \in \mathbb{N}$.
- Parameter: k.
- Problem: compute the number of satisfying assignments of C whose Hamming weight is equal to k.

Definition 2.16. $\#W\,[P] := \langle p\text{-}\#WSAT\,(CIRC)\rangle_{par}$.

3. Probabilistic approximate counting is easy

While the theorem of Toda states that exact counting is very hard [13], a well known theorem of Stockmeyer states that probabilistic approximate counting is not very hard [12], it says that probabilistic approximate counting belongs to the second level of the polynomial hierarchy. We want to prove a parameterized analogue of Stockmeyer's theorem.

3.1. Basic definitions

In this section we introduce some basic operators and list some of their basic properties. We introduce those operators to define some parameterized classes which are analogous to the classical classes that are introduced in the paper of Toda [13].

Notation 3.1. *From here on, if it is clear from the context, we will use the symbol* $\{0,1\}^f$ *to denote the set* $\{0,1\}^{f(k)\cdot\log(|x|)}$.

Definition 3.2. Let L be a parameterized language and let $\mathcal{C}$ be a parameterized class (*i.e.* a set of parameterized languages closed under fpt many-one reductions)

(1) $L \in \exists\cdot\mathcal{C}$ if and only if there exist $\Omega \in \mathcal{C}$ and a computable function f such that
$$(x,k) \in L \iff \exists y \in \{0,1\}^f\,((x,y,k) \in \Omega).$$
(2) $L \in \forall\cdot\mathcal{C}$ if and only if there exist $\Omega \in \mathcal{C}$ and a computable function f such that
$$(x,k) \in L \iff \forall y \in \{0,1\}^f\,((x,y,k) \in \Omega).$$
(3) $L \in BP\cdot\mathcal{C}$ if and only if there exist $\Omega \in \mathcal{C}$ and a computable function f such that
- $(x,k) \in L \Rightarrow \Pr_{y\in\{0,1\}^f}[(x,y,k) \in \Omega] \geq \frac{3}{4}$.
- $(x,k) \notin L \Rightarrow \Pr_{y\in\{0,1\}^f}[(x,y,k) \in \Omega] \leq \frac{1}{4}$.

Note 3.3. Let $\mathcal{C}$ be a parameterized class, the definition of the parameterized class $BP\cdot\mathcal{C}$ corresponds to the direct adaptation of the definition of the classical BP operator. In the classical setting the error probability associated to the BP operator can be bounded either by a constant, like in our definition of $BP\cdot\mathcal{C}$, or by a polynomially decaying function. Those two possible definitions are (almost always) equivalent because of the probability amplification properties of (most) classical

complexity classes. It is not the case in the parameterized framework: in the parameterized setting probability amplification is always a complex issue (which has been studied in some depth in [9]). In some cases probability amplification can be achieved with the help of efficient pseudorandom generators, but it is not always the case: probability amplification holds for a parameterized class $BP \cdot \mathcal{C}$ if and only if $\mathcal{C}$ is closed under some special type of parameterized true table reductions [9]. Also, we have to choose one of two possible definitions, we believe that working with the weakest one is the right choice, (though some proofs can become longer and more complicated).

We say that an operator F is monotone if and only if given $\mathcal{C}$ and $\mathcal{C}^*$ two parameterized classes:

(1) The containment $\mathcal{C} \subseteq F \cdot \mathcal{C}$ holds.
(2) $\mathcal{C} \subseteq \mathcal{C}^*$ implies $F \cdot \mathcal{C} \subseteq F \cdot \mathcal{C}^*$.

Lemma 3.4. *Let $\mathcal{C}$ be a parameterized class*

(1) $\exists$, $\forall$, *and BP are monotone.*
(2) $\exists \cdot FPT = W[P]$.
(3) $co - BP \cdot \mathcal{C} \subseteq BP \cdot (co - \mathcal{C})$.
(4) *If $co - \mathcal{C} = C$, then $co - BP \cdot \mathcal{C} = BP \cdot \mathcal{C}$.*
(5) $BP \cdot BP \cdot \mathcal{C} \subseteq BP \cdot \mathcal{C}$.

Proof.

- (item 3) Given $L \in co - BP \cdot \mathcal{C}$, there exist $\Omega \in \mathcal{C}$ and a computable function f such that
  (1) $(x,k) \in L \Rightarrow \Pr_{y \in \{0,1\}^f}[(x,y,k) \in \Omega] \leq \frac{1}{4}$.
  (2) $(x,k) \notin L \Rightarrow \Pr_{y \in \{0,1\}^f}[(x,y,k) \in \Omega] \geq \frac{3}{4}$.
  Hence
  (1) $(x,k) \in L \Rightarrow \Pr_{y \in \{0,1\}^f}[(x,y,k) \in \Omega^c] \geq \frac{3}{4}$.
  (2) $(x,k) \notin L \Rightarrow \Pr_{y \in \{0,1\}^f}[(x,y,k) \in \Omega^c] \leq \frac{1}{4}$.
  We can conclude that $L \in BP \cdot (co - \mathcal{C})$, since $\Omega^c \in co - \mathcal{C}$.
- (item 5) The containment $BP \cdot \mathcal{C} \subseteq BP \cdot BP \cdot \mathcal{C}$ holds, since BP is monotone. Given $L \in BP \cdot BP \cdot \mathcal{C}$, there exist $\Phi \in BP \cdot \mathcal{C}$ and a computable function f such that
  (1) $(x,k) \in L \Rightarrow \Pr_{y \in \{0,1\}^f}[(x,y,k) \in \Phi] \geq \frac{9}{10}$.
  (2) $(x,k) \notin L \Rightarrow \Pr_{y \in \{0,1\}^f}[(x,y,k) \in \Phi] \leq \frac{1}{10}$.
  Furthermore, there exist $\Omega \in \mathcal{C}$ and a computable function g such that
  (1) $(x,y,k) \in \Phi \Rightarrow \Pr_{z \in \{0,1\}^g}[(x,y,z,k) \in \Omega] \geq \frac{9}{10}$.
  (2) $(x,y,k) \notin \Phi \Rightarrow \Pr_{z \in \{0,1\}^g}[(x,y,z,k) \in \Omega] \leq \frac{1}{10}$.
  Thus, we have
  (1) $(x,k) \in L \Rightarrow \Pr_{(y,z) \in \{0,1\}^f \times \{0,1\}^g}[(x,y,z,k) \in \Omega] \geq \frac{81}{100} \geq \frac{3}{4}$.
  (2) $(x,k) \notin L \Rightarrow \Pr_{(y,z) \in \{0,1\}^f \times \{0,1\}^g}[(x,y,z,k) \in \Omega] \leq \frac{19}{100} \leq \frac{1}{4}$.
  So, we can conclude that $L \in BP \cdot \mathcal{C}$. □

In parameterized complexity two important hierarchies of parameterized classes have been considered: the W and the A hierarchies [3]. Both of these hierarchies

are in some sense analogous to the polynomial hierarchy. In this paper we introduce a third hierarchy which we consider the natural parameterized analogue of the polynomial hierarchy, we call this new hierarchy the *parameterized polynomial hierarchy* ($PH\left[P\right]$ hierarchy for short). The $PH\left[P\right]$ hierarchy is defined using the operators $\forall$ and $\exists$.

Definition 3.5. Given $\mathcal{C}$ a parameterized class we define, for each $i \in \mathbb{N}$, a new class $\Sigma_i \cdot \mathcal{C}$

(1) $\Sigma_0 \cdot \mathcal{C} := \Pi_0 \cdot \mathcal{C} := \mathcal{C}$.
(2) $\Sigma_{i+1} \cdot \mathcal{C} := \exists \cdot \Pi_i \cdot \mathcal{C}$.
(3) $\Pi_{i+1} \cdot \mathcal{C} := \forall \cdot \Sigma_i \cdot \mathcal{C}$.

We are ready to define our parameterized analogue of the polynomial hierarchy, the $PH\left[P\right]$ hierarchy.

$$PH\left[P\right] := \bigcup_{i \in \mathbb{N}} \left(\Sigma_i \cdot FPT \cup \Pi_i \cdot FPT\right).$$

Note 3.6. The $PH\left[P\right]$ hierarchy is investigated in more depth in author's Ph.D. thesis [8].

Definition 3.7. *p-CLOGSAT* is the following parameterized problem

- Input: (C, k), where $k \in \mathbb{N}$ and C is boolean circuit such that the number of its input gates is upperbounded by $k \log\left(|C|\right)$.
- Parameter: k.
- Problem: decide if (C, k) is satisfiable.

Lemma 3.8. *p-CLOGSAT is $W\left[P\right]$ complete.*

Proof. It is straightforward to verify that *p-CLOGSAT* belongs to $W\left[P\right]$. We only have to prove that *p-CLOGSAT* is $W\left[P\right]$ hard. To this end we will show that *p-WSAT (CIRC)* is *fpt* many one reducible to *p-CLOGSAT*. The proof is an easy application of the $k \log\left(n\right)$ trick of Downey and Fellows (also called $k \log\left(n\right)$ trick of Flum and Grohe [5]).

Let (C, k) be an instance of *p-WSAT (CIRC)* and let m be the number of input gates of C. We can compute in *fpt* time a circuit $D_{C,k}$ which maps $\{0,1\}^{k \log(m)}$ onto the set

$$\left\{s \in \{0,1\}^m : \text{the hamming weight of } s \text{ is lesser than or equal to } k\right\}.$$

If we hardwire the output gates of $D_{C,k}$ and the input gates of C we obtain a circuit $H_{C,k}$ such that

(1) $H_{C,k}$ is satisfiable if and only if $(C, k) \in$ *p-WSAT (CIRC)*.
(2) The size of $H_{C,k}$ is bigger than the size of C.
(3) The number of input gates of $H_{C,k}$ is equal to $k \log\left(m\right)$ and $k \log\left(m\right) \leq k \log\left(|H_{C,k}|\right)$.

Thus, $(H_{C,k}, k)$ is an instance of $p\text{-}CLOGSAT$ such that

(1) $(C,k) \in p\text{-}WSAT\,(CIRC)$ if and only if $(H_{C,k}, k) \in p\text{-}CLOGSAT$.
(2) $(H_{C,k}, k)$ can be computed from (C,k) in fpt time.

So, we have proven that $p\text{-}WSAT\,(CIRC)$ is fpt many one reducible to $p\text{-}CLOGSAT$. □

3.2. Hashing

In this section we introduce the hashing machinery that we will use in some of the proofs.

Definition 3.9. Given A and B two finite sets and given $H \subseteq B^A$ we say that H is a U_2-Hashing family if and only if for all $i, j \in A$ and for all $c, d \in B$ the equality

$$\Pr_{h \in H}\left[h\,(i) = c \;\&\; h\,(j) = d\right] = \frac{1}{|A|^2}$$

holds.

Example 3.10. Given $p, k, i \in \mathbb{N}, k \geq i$ and p prime, the set $\mathcal{H}^p_{k,i}$ of affine transformations from $GF(p)^k$ to $GF(p)^i$ is a U_2-*Hashing family* [6], where given $m \geq 1$ and p prime, the symbol $GF\,(p^m)$ denotes the finite field of size p^m.

Example 3.11. Given $r \leq n$ two natural numbers the hashing family $\mathcal{F}_{n,r}$

$$\left\{h_{a,b} \in (GF\,(2^r))^{GF(2^n)} : a, b \in \{0,1\}^n \;\&\; h_{a,b}\,(z) := (az + b) \restriction_r\right\}$$

is a U_2-Hashing family, where the operations employed in the definition of the function $h_{a,b} \in \mathcal{F}_{n,r}$ are the operations of the field $GF\,(2^n)$ [6].

Note 3.12. Any element $h \in \mathcal{H}^p_{k,i}$ is determined by a pair (A_h, v_h), where A_h is a $i \times k$ matrix with entries in $GF(p)$ and $v_h \in GF(p)^k$, it implies that we only need $O\,(ik \log\,(p))$ bits to specify a given element of $\mathcal{H}^p_{k,i}$.

Note 3.13. Note that the elements of $\mathcal{F}_{n,r}$ can be specified using $O\,(n)$ bits. It is the case because in order to specify an element, say $h_{a,b}$, of $GF\,(2^n)$ it is sufficient to specify a pair $(a,b) \in (GF\,(2^n))^2$.

U_2-Hashing families have interesting combinatorial properties, one of them is encoded in the so called *leftover hashing lemma.*

Let H be a U_2-Hashing family whose elements are functions with domain A and range B. Given $S \subset A$ and $b \in B$ we define a random variable $Y_{S,b} : H \to \mathbb{N}$ in the following way:

$$Y_{S,b}\,(h) = \left|S \cap h^{-1}(b)\right|.$$

Let ρ be the expected value of $Y_{S,b}$.

Lemma 3.14 (leftover hashing lemma I)**.**
Given $S \subset H$, given $b \in B$ and given $\epsilon \gneqq 0$ we have that

$$\Pr\left[\left|Y_{S,b}(h) - \rho\right| \geq \epsilon\rho\right] \leq \frac{1}{\epsilon^2\rho}.$$

The proof of the leftover hashing lemma can be found in [6].

3.3. A probabilistic approximation algorithm

In this section we will present a randomized algorithm, indebted to Muller [10], for approximating any problem in $\#W[P]$, the algorithm is based on Hashing techniques.

Recall the definition of the U_2-Hashing family $\mathcal{H}_{k,i}^p$ (Ex. 3.10). Given $S \subseteq GF(p)^k$ we consider the random variable $Y_i : \mathcal{H}_{k,i}^p \to \mathbb{N}$ defined by

$$Y_i(h) := \left|S \cap h^{-1}(0^i)\right|$$

where 0^i is the $\overrightarrow{0}$-vector of $GF(p)^i$.

Let ρ_i be the expected value of Y_i, we have that $\rho_i = \frac{|S|}{p^i}$. The leftover hashing lemma implies that given $\epsilon \gneqq 0$ the inequality

$$\Pr\left[\left|Y_i(h) - \frac{|S|}{p^i}\right| \geq \epsilon\frac{|S|}{p^i}\right] \leq \frac{p^i}{\epsilon^2|S|}$$

holds.

Theorem 3.15 (Muller's algorithm)**.** *For all $c, c^* \in \mathbb{N}$ there exists a randomized algorithm $\mathcal{M}$ with oracle access to the problem p-CLOGSAT and such that*

1. *On every run of $\mathcal{M}$ with input (C,k), algorithm $\mathcal{M}$ uses $O\left(k^3\log(|C|)\right)$ random bits.*
2. *On every run of $\mathcal{M}$ with input (C,k), algorithm $\mathcal{M}$ makes less than $f(k)\,p(|C|)$ oracle queries, for some computable function f and some polynomial $p(X)$.*
3. *On every run of $\mathcal{M}$ with input (C,k), the running time of $\mathcal{M}$ is upper-bounded by $g(k)\,q(|C|)$, for some computable function g and some polynomial $q(X)$.*
4. *On every run r of $\mathcal{M}$ with input (C,k), algorithm $\mathcal{M}$ computes a computable number $\mathcal{M}_{C,k,r}$ (which depends on the run) such that*

$$\Pr_r\left[\#(C,k)\left(1 - \frac{1}{|C|^c}\right) \leq \mathcal{M}_{C,k,r} \leq \#(C,k)\left(1 + \frac{1}{|C|^c}\right)\right] \geq 1 - \frac{1}{|C|^{c^*}}$$

where $\#(C,k)$ denotes the number of satisfying assignments of C.

Proof. Given c and c^* algorithm M is defined in the following way:
On input (C,k)

(1) $\mathcal{M}$ computes $l = 2c + c^* + 1$ (to get a error probability of $1 - \frac{1}{|C|^{c^*}}$).
(2) $\mathcal{M}$ computes a prime number $p \in \{|C|, \ldots, 2|C|+1\}$.
(3) For all $i \in \{1, \ldots, k\}$, algorithm $\mathcal{M}$ randomly chooses $h \in \mathcal{H}^p_{k,i}$.
(4) $\mathcal{M}$ computes $i_0 := \min\left\{i \leq k : Y_i(h) \lneqq p^l\right\}$.
(5) $\mathcal{M}$ outputs $p^{i_0} Y_{i_0}(h_{i_0})$.

- We know, from the Bertrand's postulate, that there exists a prime number $p \in \{|C|, \ldots, 2|C|+1\}$. Such a number can be computed in polynomial time, wrt the size of C, using either a brute force algorithm or the algorithm of Agrawal-Saxena-Kayal [1].
- In the third step $\mathcal{M}$ chooses k random functions from $\mathcal{H}^p_{k,i}$, each one of those functions is determined by choosing at most $\left(k^2+k\right)\log(2|C|)$ random bits, that means that the number of random bits used by Muller's algorithm is $O\left(k^3 \cdot \log(|C|)\right)$.
- The checking in step 4 can be performed in fpt time using the p-$CLOGSAT$ oracle. This is the case, because l is a fixed number that does not depend on k and the problem p-$CLOGSAT$ is self-reducible. The self-reducibility of p-$CLOGSAT$ implies that the listing problem associated to p-$CLOGSAT$ can be solved in fpt-delay time using the oracle p-$CLOGSAT$ [10].

Claim. $\Pr\left[\#(C,k)\left(1-\frac{1}{|C|^c}\right) \leq \mathcal{M}_{C,k} \leq \#(C,k)\left(1+\frac{1}{|C|^c}\right)\right] \geq 1 - \frac{1}{|C|^{c^*}}$.

Let d be equal to $\#(C,k) \leq |C|^k \leq p^k$ and let m be a natural number such that $m \leq k$ and $p^{m-1} \leq d \leq p^m$.

- If $i \leq m-l-2$, then

$$\rho_i \geq \frac{p^{m-1}}{p^{m-l-2}} = p^{l+1}.$$

This is the case because

$$\begin{aligned}
\Pr_{h\in\mathcal{H}^p_{k,i}}\left[Y_i \lneqq p^l\right] &\leq \Pr_{h\in\mathcal{H}^p_{k,i}}\left[Y_i \lneqq \frac{1}{p}\rho_i\right] \\
&\leq \Pr_{h\in\mathcal{H}^p_{k,i}}\left[|Y_i - \rho_i| \geq \frac{(p-1)}{p}\rho_i\right] \\
&\leq \frac{p^2}{(p-1)^2\rho_i} \leq \frac{p^2}{(p-1)^2 p^{l+1}} \leq \frac{1}{(p-1)^2 p^{l-1}}.
\end{aligned}$$

- If $i = m-l+1$, then

$$p^{l-1} = \frac{p^m}{p^{m-l+1}} \geq \rho_i \geq \frac{p^{m-1}}{p^{m-l+1}} = p^{l-2}.$$

It follows from the following inequalities:

$$\begin{aligned}\Pr_{h\in\mathcal{H}^p_{k,i}}\left[Y_i \geq p^l\right] &\leq \Pr_{h\in\mathcal{H}^p_{k,i}}\left[|Y_i-\rho_i| \geq (p-1)\,\rho_i\right]\\ &\leq \frac{1}{(p-1)^2\,\rho_i} \leq \frac{1}{(p-1)^2\,p^{l-2}}.\end{aligned}$$

- If $i \in \{m-l-1, m-l, m-l+1\}$, then

$$\rho_i = \frac{d}{p^i} \geq \frac{p^{m-1}}{p^{m-l+1}} = p^{l-2}.$$

It follows from the following inequalities:

$$\begin{aligned}\Pr_{h\in\mathcal{H}^p_{k,i}}\left[\left|p^i Y_i\right| \geq \epsilon d\right] &= \Pr_{h\in\mathcal{H}^p_{k,i}}\left[|Y_i-\rho_i| \geq \epsilon\rho_i\right]\\ &\leq \frac{1}{\epsilon^2\rho_i} \leq \frac{1}{\epsilon^2 p^{l-2}}.\end{aligned}$$

From the first item we obtain that

$$\Pr\left[i_0 \leq m-l-2\right] \leq \frac{m-l-2}{(p-1)^2\,p^{l-1}}.$$

And from the second item we obtain

$$\Pr\left[i_0 \gneqq m-l+1\right] \leq \frac{2}{(p-1)^2\,p^{l-2}}.$$

Therefore

$$\Pr\left[i_0 \notin \{m-l-1, m-l, m-l+1\}\right] \leq \frac{1}{(p-1)^2\,p^{l-3}}.$$

Now, if we take $\epsilon := \frac{1}{p^c}$ we get

$$\frac{1}{\epsilon^2 p^{l-2}} = \frac{p^{2c}}{p^{l-2}} \leq p^{-c^*-1} \leq |C|^{-c^*-1}.$$

Altogether we have

$$\Pr\left[\#(C,k)\left(1-\frac{1}{|C|^c}\right) \leq \mathcal{M}_{C,k} \leq \#(C,k)\left(1+\frac{1}{|C|^c}\right)\right] \geq 1-\frac{1}{|C|^{c^*}}. \qquad \square$$

Corollary 3.16. *Given $\chi \in \#W[P]$ there exists a randomized fpt algorithm $\mathcal{M}$ with access to the oracle p-CLOGSAT that approximates the function χ.*

Note 3.17. Given L a $W[P]$ complete problem we could use, in Muller's algorithm, an oracle L instead of the oracle $p\text{-}CLOGSAT$[1].

In the following sections we will prove that approximate counting belongs to the $PH[P]$ hierarchy. The proof is divided in the following stages:

(1) Given $\chi \in \#W[P]$, we prove that the complexity of approximating χ, within a given constant range, is equal to the complexity of a parameterized gap problem that we call $p\text{-}approx(\chi)$.
(2) Using hashing techniques we prove that for every $\chi \in \#W[P]$, the problem $p\text{-}approx(\chi)$ belongs to $BP \cdot \exists \cdot FPT$.
(3) We prove that, if $W[P]$ is $\wedge$-closed, then $BP \cdot \exists \cdot FPT$ is included in the second level of the $PH[P]$ hierarchy, that is: we prove a parameterized analogue of Lautemann-Sipser theorem. The proof of this theorem is based on Lautemann's proof but in addition we have to use the pseudorandom generator of Ajtai, Komlos and Szemeredi to reduce the number of random bits used in the probabilistic constructions included in the proof.

4. Approximate counting and gap problems: approximate counting belongs to $BP \cdot \exists \cdot FPT$

The main theorem in last subsection says that we can probabilistic approximate every problem in $\#W[P]$ using a $W[P]$ oracle, this theorem gives us strong evidence against the $PH[P]$ hardness of approximate counting. While in the classical framework we could already claim that probabilistic approximate counting belongs to the second level of the polynomial hierarchy, we have still not proven that this is the case in the parameterized framework because we don't know if the levels of the $PH[P]$ hierarchy are closed under parameterized Turing reductions, (it is very unlikely that $PH[P]$ is closed under parameterized Turing reductions).

Given $\mathcal{M}$ a randomized algorithm and given (x,k) an input of $\mathcal{M}$ we use the symbol $I_{x,k}$ to denote the set of possible random choices of algorithm $\mathcal{M}$, on input (x,k). Let $\mathcal{O}$ be the set of possible outputs of algorithm $\mathcal{M}$, we use the symbol $\mathcal{M}(x,k)$ to denote the random variable $\mathcal{M}(x,k) : I_{x,k} \to \mathcal{O}$ defined by:

$$\mathcal{M}(x,k)(\xi) = \mathcal{M}(x,k,\xi)$$

where $\mathcal{M}(x,k,\xi)$ is the output of $\mathcal{M}$, on input (x,k), determined by the random choice ξ.

Definition 4.1. Given a parameterized counting problem χ and given $c \geq 1$, a randomized fpt algorithm $\mathcal{M}$ approximates χ within the range c if and only if for all instance (x,k) of χ the inequality

$$\Pr\left[\chi(x,k)\frac{1}{c} \leq \mathcal{M}(x,k) \leq \chi(x,k)\,c\right] \geq \frac{2}{3}$$

holds, where the probability is computed with respect to the random choices of $\mathcal{M}$.

[1] A detailed and alternative proof of this theorem can be found in [10].

In this section we will investigate the hardness of approximating, within a constant range, a given problem $\chi \in \#W[P]$. First we have to state and prove a technical lemma.

Lemma 4.2. *Given $\chi \in \#W[P]$ and given $c \in \mathbb{N}$, the function $\chi_c : \Sigma^* \times \mathbb{N} \to \mathbb{N}$ defined by*

$$\chi_c(x,k) = (\chi(x,k))^c$$

belongs to $\#W[P]$.

Proof. If $\chi \in \#W[P]$, we know that there exist a $W[P]$ Turing machine $\mathbb{M}$ and a computable function g such that

(1) For every run of $\mathbb{M}$ with input (x,k), machine $\mathbb{M}$ makes $g(k)\log(|x|)$ nondeterministic guesses.
(2) The running time of every run of $\mathbb{M}$, with input (x,k), is bounded by $f(k)\,p(|x|)$.
(3) $\chi(x,k) = \#acc(\mathbb{M}(x,k))$.

Let $\mathbb{M}_c$ be a $W[P]$ Turing machine such that

- For every run of $\mathbb{M}_c$ with input (x,k), machine $\mathbb{M}_c$ nondeterministically guesses $r_1,\ldots,r_c \in \{0,1\}^g$.
- For all $i \leq c$, machine $\mathbb{M}_c$ simulates the run of $\mathbb{M}$, on input (x,k), when $\mathbb{M}$ uses the nondeterministic choices codified by r_i.
- $\mathbb{M}_c$ accepts (x,k) if and only if for all $i \leq c$ the simulation of the deterministic computation of $\mathbb{M}$, on input (x,r_i,c), ends in an accepting state.

It is clear that $\chi_c(x,k) = (\#acc(\mathbb{M}(x,k)))^c$. □

Next lemma says that if we can compute approximations within the range 2, we can compute approximations within any constant range $c \gneqq 1$. Because of this, we will only investigate the complexity of computing approximations within the range 2.

Lemma 4.3. *Given χ a $\#W[P]$ complete problem, if there exists an algorithm $\mathcal{M}$ that approximates χ within the range 2, then for all $c \gneqq 1$, there exists an algorithm $\mathcal{M}_c$ that approximates χ within the range c.*

Proof. Let d be a natural number such that $(2)^{\frac{1}{d}} \leq c$. The algorithm $\mathcal{M}_c$ is the following one:

On input (x,k)

(1) $\mathcal{M}_c$ computes a pair (x^*,k^*) such that $\chi(x^*,k^*) = (\chi(x,k))^d$.
(2) $\mathcal{M}_c$ simulates the computation of $\mathcal{M}$, on input (x^*,k^*).
(3) Given t the output of $\mathcal{M}$ on input $\left(x^*,k^*\right)$, algorithm $\mathcal{M}_c$ outputs $(t)^{\frac{1}{d}}$.

The computation in step 1 can be performed in fpt time given that $\mathcal{M}_c$ only has to compute a pair (x^*,k^*) such that $\chi(x^*,k^*) = \chi_c(x,k)$, remember that $\chi_c \in \#W[P]$ and χ is $\#W[P]$ complete.

In step 2, algorithm M_c computes a number t such that

$$\Pr\left[\frac{1}{2}(\chi(x,k))^d \leq t \leq 2(\chi(x,k))^d\right] \geq \frac{2}{3}.$$

Therefore

$$\begin{aligned}\frac{2}{3} &\leq \Pr\left[\left(\frac{1}{2}\right)^{\frac{1}{d}}\chi(x,k) \leq (t)^{\frac{1}{d}} \leq (2)^{\frac{1}{d}}\chi(x,k)\right] \\ &\leq \Pr\left[\frac{1}{c}\chi(x,k) \leq (t)^{\frac{1}{d}} \leq c\chi(x,k)\right].\end{aligned}$$

We can conclude that

$$\Pr\left[\frac{1}{c}\chi(x,k) \leq M_c(x,k) \leq c\chi(x,k)\right] \geq \frac{2}{3}.$$

Given that $(t)^{\frac{1}{d}}$ is the output $\mathcal{M}_c$. □

Given $\chi \in \#W[P]$ we define a parameterized *gap problem* that corresponds in some sense to the problem of approximating χ within the range 2.

Definition 4.4. *p-approx* (χ) is the parameterized gap problem defined by
Input: (x,k,c), where (x,k) is an instance of χ and $c \in \mathbb{N}$.
Parameter: k.
Yes-instances: (x,k,c) is a Yes-instance if and only if $\chi(x,k) \geq 2c$.
Not-instances: (x,k,c) is a Not-instance if and only if $\chi(x,k) \leq c$.

We prove that we can approximate χ within the range 2, (in fpt time), if oracle access to *p-approx*(χ) is provided.

Let (x,k) be an instance of χ, there exists a natural number $m \leq |x|^k$ such that

$$2^m \leq \chi(x,k) \leq 2^{m+1}.$$

It follows from the definition of m that

(1) $\frac{1}{2}\chi(x,k) \leq 2^m \leq 2(\chi(x,k))$.
(2) $\frac{1}{2}\chi(x,k) \leq 2^{m+1} \leq 2\chi(x,k)$.

So, in order to approximate $\chi(x,k)$ within the range 2 it is sufficient to compute a number $n \in \{m, m+1\}$. The proof of the following theorem is based on this fact.

Theorem 4.5. *Given $\chi \in \#W[P]$ there exists an fpt algorithm $\mathcal{M}$ with access to the oracle p-approx(χ) and such that*

(1) *$\mathcal{M}$ approximates χ within the range 2.*
(2) *$\mathcal{M}$ queries the oracle at most $g(k)\log(|x|)$ times, where g is a computable function.*

Proof. On input (x,k) the algorithm $\mathcal{M}$ computes a number $n \in \mathbb{N}$ such that $n \in \{m, m+1\}$, after that $\mathcal{M}$ outputs 2^n.

First two easy facts.

- If $i \leq m-1$, then $(x,k,2^i)$ is a Yes-instance of *p-approx*(χ).
- If $i \geq m+1$, then $(x,k,2^i)$ is a Not-instance of *p-approx*(χ).

Let g be a computable function such that for all $(x,k) \in \Sigma^* \times \mathbb{N}$ we have that $\chi(x,k) \leq 2^{g(k)\log(|x|)}$. Algorithm $\mathcal{M}$ works in the following way:

On input (x,k)

(1) For all $i \leq g(k)\log(|x|)$, algorithm $\mathcal{M}$ computes $v_i \in \{0,1\}$ such that:

$$v_i = \begin{cases} 0, \text{ if the answer to the oracle query } \left(x,k,2^i\right) \text{ is NO;} \\ 1, \text{ otherwise.} \end{cases}$$

(2) $\mathcal{M}$ computes $n := \min\{i \leq g(k)\log(|x|) : v_i = 0\}$.
(3) $\mathcal{M}$ outputs 2^n.

Fact. $n \in \{m, m+1\}$.

This is the case because

(1) If $i \leq m-1$, then $v_i = 1$.
(2) If $i \geq m+1$, then $v_i = 0$.

Then, it is clear that:

- $2^n \in \left\{2^m, 2^{m+1}\right\}$.
- $\frac{1}{2}\chi(x,k) \leq 2^n \leq 2\chi(x,k)$.

We have that the output of $\mathcal{M}$ is an approximation of $\chi(x,k)$ within the range 2. □

Last theorem allows us to identify the problem of computing approximations to χ within the range 2 with the gap problem *p-approx*(χ).

4.1. Approximate counting belongs to $BP \cdot \exists \cdot FPT$

In the following we will analyze the complexity of the parameterized gap problems *p-approx*(χ), with $\chi \in \#W[P]$. We want to prove that for all $\chi \in \#W[P]$ the gap problem *p-approx*(χ) belongs to some level of the $PH[P]$ hierarchy. To this end, we prove:

(1) For all $\chi \in \#W[P]$ we have that *p-approx*$(\chi) \in BP \cdot \exists \cdot FPT$.
(2) A parameterized analogue of the Lautemann-Sipser theorem, that is: we prove that $BP \cdot \exists \cdot FPT$ is included in the second level of the parameterized polynomial hierarchy [7].

4.1.1. *Approximate counting belongs to the parameterized Arthur-Merlin class*

In this section we prove that *p-approx*$(\chi) \in BP \cdot \exists \cdot FPT$. The core of the argument is an standard Hashing argument. Hashing allow us to transform a dicothomy of the form:

Either there are so many certificates (*more than* $2c$) *or there are so few* (*less than* c).

Into a dicothomy of the form:

The probability that there are at least one certificate is either very high (*bigger then* $\frac{3}{4}$) *or very small* (*less than* $\frac{1}{4}$).

Note that this is the type of transformation that we need if we want to prove that $p\text{-}approx(\chi) \in BP \cdot \exists \cdot FPT$.

Let us begin with the proof. First we have to define the meaning of a gap problem belonging to $BP \cdot \exists \cdot FPT$.

We will say that $p\text{-}approx(\chi) \in BP \cdot \exists \cdot FPT$ if and only if there exist $\Omega \in FPT$ and two computable functions h, g such that:

- If (x,k,c) is a Yes-instance of $p\text{-}approx\,(\chi)$, then

$$\Pr_{z \in \{0,1\}^g}\left[\exists y \in \{0,1\}^h \left((x,y,z,k) \in \Omega\right)\right] \geq \frac{3}{4}.$$

- If (x,k,c) is a Not-instance of $p\text{-}approx(\chi)$, then

$$\Pr_{z \in \{0,1\}^g}\left[\exists y \in \{0,1\}^h \left((x,y,z,k) \in \Omega\right)\right] \leq \frac{1}{4}.$$

Now we prove that given $\chi \in \#W\,[P]$ the parameterized gap problem $p\text{-}approx\,(\chi)$ belongs to $BP \cdot \exists \cdot FPT$. The proof relies on the leftover hashing lemma.

Given χ a problem in $\#W\,[P]$, there exists $\Omega \in FPT$ such that $\chi(x,k) = |S_{x,k}|$, where $S_{x,k}$ is the set

$$\left\{y \in \{0,1\}^h : (x,y,k) \in \Omega\right\}.$$

We consider the language $\Omega^6 \in FPT$ defined by

$$\Omega^6 := \left\{(x,y_1,\ldots,y_6,k) : y_1,\ldots,y_6 \in \{0,1\}^h \ \&\ (x,y_1,k) \in \Omega,\ldots,(x,y_6,k) \in \Omega\right\}.$$

Let $S^6_{x,k}$ be the set

$$\{(y_1,\ldots,y_6) : y_1,\ldots,y_6 \in S_{x,k}\}.$$

Fact 4.6. *Given* (x,k,c) *an instance of p-approx* (χ)*:*

(1) *If* (x,k,c) *is a Yes-instance of p-approx*(χ)*, then* $|S^6_{x,k}| \geq 2^6 c^6$.
(2) *If* (x,k,c) *is a Not-instance of p-approx*(χ)*, then* $|S^6_{x,k}| \leq c^6$.

Let (x,k,c) be an instance of $p\text{-}approx(\chi)$ and let n,m be natural numbers such that $n = 6h\,(k)\log\,(|x|)$ and $m = \log\left(4c^6\right)$.

Recall the definition of the U_2-Hashing family $\mathcal{F}_{n,m}$ (Ex. 3.11).

Lemma 4.7. *If* (x,k,c) *is a Yes-instance of p-approx*(χ)*, then*

$$\Pr_{r \in \mathcal{F}_{n,m}}\left[\exists y \in \{0,1\}^{6\cdot h}\left(y \in S^6_{x,k}\ \&\ r\,(y) = 0^m\right)\right] \geq \frac{3}{4}.$$

Proof. Let (x,k,c) be a Yes-instance of *p-approx*(χ) and let $Y_m : \mathcal{F}_{n,m} \to \mathbb{N}$ be the random variable defined by

$$Y_m(r) := \left|S^6_{x,k} \cap r^{-1}(0^m)\right|$$

where $r \in \mathcal{F}_{n,m}$. If ρ_m is the expected value of Y_m we have that $\rho_m \geq 16$. Now if we use the leftover hashing lemma (Lem. 3.14), choosing $\epsilon = \frac{1}{2}$, we obtain

$$\Pr_{r\in\mathcal{F}_{n,m}}[Y_m(r) = 0] \leq \Pr_{r\in\mathcal{F}_{n,m}}\left[|Y_m(r) - \rho_m| \gneqq \frac{1}{2}\rho_m\right] \leq \frac{1}{4}. \qquad \square$$

Lemma 4.8. *If (x,k,c) is a Not-instance of p-approx(χ), then*

$$\Pr_{r\in\mathcal{F}_{n,m}}\left[\exists y \in \{0,1\}^{6\cdot h}\left(y \in S^6_{x,k} \;\&\; r(y) = 0^m\right)\right] \leq \frac{1}{4}.$$

Proof. First we note that for all $y \in \{0,1\}^{6\cdot h}$ the inequality

$$\Pr_{r\in\mathcal{F}_{n,m}}[r(y) = 0^m] \leq 2^{-m}$$

holds. Therefore

$$\begin{aligned}\Pr_{r\in\mathcal{F}_{n,m}}\left[\exists y \in S^6_{x,k}\ (\ r(y) = 0^m)\right] &\leq \sum_{y\in S^6_{x,k}} \Pr_{r\in\mathcal{F}_{n,m}}[r(y) = 0^m] \\ &\leq \left|S^6_{x,k}\right| 2^{-m} \leq c^6 2^{-m} \leq \frac{1}{4}. \qquad \square\end{aligned}$$

Theorem 4.9. *Given $\chi \in \#W[P]$, the gap language p-approx(χ) belongs to $BP\cdot\exists\cdot FPT$.*

Proof. Given x, k and c we take $m = \log\left(4c^6\right)$ and $n = 6h(k)\log(|x|)$ and we consider the language Ω^* defined by

$$\Omega^* := \{(x, y_1, \ldots, y_6, m, r, k) : \varphi_{n,m} \;\&\; \psi_{n,m}\}$$

where:

(1) $\varphi_{n,m} := y_1, \ldots, y_6 \in \{0,1\}^h \;\&\; (x, y_1, k), \ldots, (x, y_6, k) \in \Omega$.
(2) $\psi_{n,m} := m \leq n \;\&\; r \in \mathcal{F}_{n,m} \;\&\; r(y_1, \ldots, y_6) = 0^m$.
(3) h is a computable function.
(4) Ω is a language in FPT such that for all instance (x,k) of χ the equality

$$\chi(x,k) = \left|\left\{y \in \{0,1\}^h : (x,y,k) \in \Omega\right\}\right|$$

holds.

From the last two lemmas we have that:

- If (x,k,c) is a Yes-instance of *p-approx*(χ), then

$$\Pr_{r\in\mathcal{F}_{n,m}}\left[\exists y_1,\ldots,y_6\in\{0,1\}^h\left((x,y_1,\ldots,y_6,m,r,k,)\in\Omega^*\right)\right]\geq\frac{3}{4}.$$

- If (x,k,c) is a Not-instance of *p-approx*(χ), then

$$\Pr_{r\in\mathcal{F}_{n,m}}\left[\exists y_1,\ldots,y_6\in\{0,1\}^h\left((x,y_1,\ldots,y_6,m,r,k,)\in\Omega^*\right)\right]\leq\frac{1}{4}.$$

Note that the number of random bits used to specify the random choice r, where $r\in\mathcal{F}_{n,m}$, is $O\left(h\left(k\right)\log\left(|x|\right)\right)$. Thus, we have proven that *p-approx*(χ) belongs to the parameterized class $BP\cdot\exists\cdot FPT$. □

5. A parameterized Lautemann-Sipser theorem: approximate counting belongs to the $PH\left[P\right]$ hierarchy

We are trying to prove that for all $\chi\in\#W\left[P\right]$ the gap problem *p-approx*(χ) belongs to $PH[P]$. We already know that given $\chi\in\#W\left[P\right]$ the gap problem *p-approx*(χ) belongs to $BP\cdot\exists\cdot FPT$. In this section we prove that, under certain complexity theoretic hypothesis, the class $BP\cdot\exists\cdot FPT$ is included in $\forall\cdot\exists\cdot FPT$. Specifically we prove that

(1) $BP\cdot FPT\subset\exists\cdot\forall\cdot FPT\cap\forall\cdot\exists\cdot FPT$.
(2) If $\exists\cdot FPT$ is $\wedge$-closed, then $BP\cdot\exists\cdot FPT$ is included in $\forall\cdot\exists\cdot FPT$.

As a corollary we get our parameterized analogue of Stockmeyer's theorem: if $\exists\cdot FPT$ is $\wedge$-closed, parameterized approximate counting belongs to $\forall\cdot\exists\cdot FPT$.

Note 5.1. Remember that $\exists\cdot FPT$ is equal to $W\left[P\right]$.

Our proof is very close to Lautemann's proof [7], though we have to use the pseudorandom generator of Ajtai, Komlos and Szemeredi (see Ref. [6]) (*AKS* algorithm, for short) to save random bits and we have to take into account some technical details.

5.1. Majority reductions and probability amplification

A probabilistic parameterized class $BP\cdot\mathcal{C}$ is well behaved if $BP\cdot\mathcal{C}$ has some type of probability amplification, *i.e.* $BP\cdot\mathcal{C}$ is well behaved if given $L\in BP\cdot\mathcal{C}$ we can decrease the error probability associated to L by reducing L to some other problem in the class $BP\cdot\mathcal{C}$. Here we introduce a formal notion of well behaveness that we call the *pam* property.

Definition 5.2. Given $\mathcal{C}$ a parameterized class, $BP\cdot\mathcal{C}$ has the *pam* property if and only if for all $L\in BP\cdot\mathcal{C}$ and for all computable function g there exist $\Omega\in\mathcal{C}$ and a computable function f such that

- $(x,k)\in L\Rightarrow\Pr_{y\in\{0,1\}^f}\left[(x,y,k)\in\Omega\right]\geq 1-2^{-g(k)\log(|x|)}$.

- $(x,k) \notin L \Rightarrow \Pr_{y\in\{0,1\}^f}\left[(x,y,k)\in\Omega\right] \leq 2^{-g(k)\log(|x|)}$.

In this section we study the relation between the closure of $\mathcal{C}$ under majority reductions and the probability-amplification properties of $BP\cdot\mathcal{C}$. We prove that if $\mathcal{C}$ is *maj*-closed, then $BP\cdot\mathcal{C}$ has the *pam* property. The proof of this theorem is very similar to the classical analogue, but in addition we have to use in the proof the AKS algorithm in order to save random bits.

Notation 5.3. *From here on, we will use the symbol $\otimes$ to denote the boolean operator Majority.*

Definition 5.4. L is majority reducible to L^* if and only if there exist an fpt algorithm $\mathcal{M}$ and two computable functions f,g such that, on input (x,k), algorithm $\mathcal{M}$ computes a sequence

$$(x_1,k_1),\ldots,\left(x_{f(k)\log(|x|)},k_{f(k)\log(|x|)}\right)$$

that satisfies:

(1) $(x,k)\in L \Leftrightarrow \bigotimes_{j\leq f(k)\log(|x|)} (x_j,k_j)\in L^*$.
(2) For all $i\leq f(k)\log(|x|)$ we have $k_i\leq g(k)$.

We will say that $\mathcal{C}$ is maj-closed if and only if $\mathcal{C}$ is closed under majority reductions. Next theorem says us that in order to amplify the success probability (equivalently, to decrease the error probability), we can make a big saving of random bits if we use a suitable *pseudorandom generator*.

Theorem 5.5 (AKS theorem)**.** *There exist an algorithm, namely AKS, and constants $N_1,N_2\in\mathbb{N}$ such that for all $m,i\in\mathbb{N}$ and for all $a\in\{0,1\}^{N_1(m+i)}$ algorithm AKS computes, on input (a,i,m), a sequence $a_1,\ldots,a_{iN_2}\in\{0,1\}^m$ such that for all $A\subset\{0,1\}^m$*

(1) $|A|\geq\frac{3}{4}2^m \Rightarrow \Pr_{a\in\{0,1\}^{N_1(m+i)}}\left[\bigotimes_{j\leq iN_2} a_j\in A\right]\geq 1-2^{-i}$.
(2) $|A|\leq\frac{1}{4}2^m \Rightarrow \Pr_{a\in\{0,1\}^{N_1(m+i)}}\left[\bigotimes_{j\leq iN_2} a_j\in A\right]\leq 2^{-i}$.
(3) *The running time of AKS is bounded by a polynomial $p(m,i)$.*

Note 5.6. The algorithm AKS is the pseudorandom generator of Ajtai, Komlos and Szemeredi which is based on expander graphs (see Ref. [6]).

Notation 5.7. *From now on we will use the symbols N_1 and N_2 to denote the constants mentioned in the statement of theorem 5.5 (i.e. N_1 and N_2 denote the parameters of the AKS algorithm).*

Using AKS theorem we can easily prove the following theorem which says us that there exists a deep relation between the closure under majority reductions and probability amplification.

Theorem 5.8. *If $\mathcal{C}$ is maj-closed, then $BP\cdot\mathcal{C}$ has the pam property.*

Proof. Let L, Ω be languages such that $L \in BP \cdot \mathcal{C}$, $\Omega \in \mathcal{C}$ and

- $(x,k) \in L \Rightarrow \Pr_{y \in \{0,1\}^f}\left[(x,y,k) \in \Omega\right] \geq \frac{3}{4}$,
- $(x,k) \notin L \Rightarrow \Pr_{y \in \{0,1\}^f}\left[(x,y,k) \in \Omega\right] \leq \frac{1}{4}$

where f is some suitable computable function.

Given g a computable function we define Ω^g in the following way

$$\Omega^g := \left\{ (x,y,k) : y \in \{0,1\}^{N_1(f+g)} \ \& \bigotimes_{j \leq N_2 g(k)\log(|x|)} (x, z_j, k) \in \Omega \right\}$$

where $z_1, \ldots, z_{N_2 g(k)\log(|x|)}$ is the output-sequence of the algorithm AKS on input $(y, g(k)\log(|x|), f(k)\log(|x|))$. The problem Ω^g belongs to $\mathcal{C}$ because $\mathcal{C}$ is maj-closed and it follows from the AKS-theorem that

- $(x,k) \in L \Rightarrow \Pr_{y \in \{0,1\}^{N_1(f+g)}}\left[(x,y,k) \in \Omega^g\right] \geq 1 - 2^{-g(k)\log(|x|)}$.
- $(x,k) \notin L \Rightarrow \Pr_{y \in \{0,1\}^{N_1(f+g)}}\left[(x,y,k) \in \Omega^g\right] \leq 2^{-g(k)\log(|x|)}$.

Therefore, $BP \cdot \mathcal{C}$ has the *pam* property □

Corollary 5.9. *$BP \cdot FPT$ has the pam property.*

Proof. FPT is closed under majority reductions. □

5.2. A parameterized Lautemann-Sipser theorem

Let $\mathcal{C}$ be a parameterized class such that $BP \cdot \mathcal{C}$ has the *pam* property. Given $L \in BP \cdot \mathcal{C}$ we can suppose that there exist $\Omega \in \mathcal{C}$ and a computable function f such that

- $(x,k) \in L \Longrightarrow |S_{x,k}| \geq 2^{f(k)\log(|x|)}\left(1 - 2^{-k\log(|x|)}\right)$,
- $(x,k) \notin L \Longrightarrow |S_{x,k}| \leq 2^{f(k)\log(|x|)}2^{-k\log(|x|)}$,

where $S_{x,k} := \left\{ y \in \{0,1\}^f : (x,y,k) \in \Omega \right\}$.

Along this section we fix a parameterized class $\mathcal{C}$ such that the *pam* property holds for $BP \cdot \mathcal{C}$.

Notation 5.10. *In the following if $v \in \{0,1\}^{N_1 f}$, the symbol $AKS(v)$ will denote the set $\left\{v_1, \ldots, v_{N_2 f(k)\log(|x|)}\right\}$, where $v_1, \ldots, v_{N_2 f(k)\log(|x|)}$ is the output-sequence of the algorithm AKS, on input $(v, 2f(k)\log(|x|), f(k)\log(|x|))$.*

Note that if $(x,k) \in L$, the inequality

$$\Pr_{v \in \{0,1\}^{N_1 f}}\left[\left\{v_1, \ldots, v_{N_2 f(k)\log(|x|)}\right\} \cap S_{x,k} \neq \varnothing\right] \geq 1 - 2^{-2f(k)\log(|x|)}$$

holds.

Notation 5.11. *If it is clear from the context we will use the symbol n to denote the number $f(k)\log(|x|)$.*

Notation 5.12. *Given $S \subset \{0,1\}^m$ and given $v \in \{0,1\}^m$ we use the symbol $S+v$ to denote the set*

$$\{s + v : s \in S\}$$

where $+$ denotes the addition operation over the vector space $GF(2)^m$.

The next two lemmas are our parameterized version of the core of Lautemann's probabilistic argument.

Lemma 5.13. *If $(x,k) \in L$, then $\exists v \in \{0,1\}^{N_1 \cdot f} \forall z \in \{0,1\}^f((AKS(v)+z) \cap S_{x,k} \neq \emptyset)$.*

Proof. Let (x,k) be a member of L. We prove that

$$Pr_{v \in \{0,1\}^{N_1 f}}[\forall z \in \{0,1\}^f ((AKS(v)+z) \cap S_{x,k} \neq \emptyset)] > 0.$$

If we fix $z \in \{0,1\}^f$, we have

$$Pr_{v \in \{0,1\}^{N_1 f}}[AKS(v)+z \subset (S_{x,k})^c] \leq 2^{-2n}.$$

Therefore

$$\begin{aligned} &Pr_{v \in \{0,1\}^{N_1 f}}[\exists z \in \{0,1\}^f (AKS(v)+z \subset (S_{x,k})^c)] \\ &\qquad \leq \sum_{z \in \{0,1\}^f} Pr_{v \in \{0,1\}^{N_1 f}}[AKS(v)+z \subset (S_{x,k})^c] \leq 2^n 2^{-2n} \lneqq 1. \end{aligned}$$

Thus, we have

$$Pr_{v \in \{0,1\}^{N_1 f}}[\forall z \in \{0,1\}^f (AKS(v)+z \not\subseteq (S_{x,k})^c)] \gneqq 0.$$

So, we have proven that if $(x,k) \in L$, then

$$\exists v \in \{0,1\}^{N_1 \cdot f} \forall z \in \{0,1\}^f((AKS(v)+z) \cap S_{x,k} \neq \emptyset). \qquad \square$$

Fact 5.14. *For all $k \in \mathbb{N}$ there exists $R_k \in \mathbb{N}$ such that if $|x| \geq R_k$, then*

$$2^{k \log(|x|)} \geq N_2 f(k) \log(|x|) = N_2 n.$$

Proof. We can suppose, without loss of generality, that $k \geq 2$ and that $|x| \geq \log(|x|)$. If we fix $k \geq 2$ we can take $R_k \geq N_2 f(k)$, note that

$$2^{k \log(|x|)} = |x|^k \geq |x|^2 \geq N_2 f(k) |x| \geq N_2 f(k) \log(|x|). \qquad \square$$

Lemma 5.15. *If $(x,k) \in L$ and $|x| \geq R_k$, then for all $S \subset \{0,1\}^f$ such that $|S| \leq N_2 n$, there exists $z \in \{0,1\}^f$ such that $S + z \subset S_{x,k}$.*

Proof. Suppose that for all $z \in \{0,1\}^f$ we have that $S + z \not\subseteq S_{x,k}$, (*i.e.* for all $z \in \{0,1\}^f$ we have that $S \not\subseteq S_{x,k} + z$). Then, for all $z \in \{0,1\}^f$ there exists $s_z \in S$ such that $s_z \notin S_{x,k} + z$. It implies that there exists $s \in S$ such that $s \notin S_{x,k} + z$ for at least $\frac{2^n}{N_2 n}$ of the z's, *i.e.* there exists $s \in S$ such that

$$|\{u : s + u \notin S_{x,k}\}| \geq \frac{2^n}{N_2 n}.$$

Let H_s be a set with the following two properties

(1) $H_s \subset \{u : s + u \notin S_{x,k}\}$.
(2) $|H_s| \geq \frac{2^n}{N_2 n}$.

It is easy to verify that

- $H_s + s \subset (S_{x,k})^c$.
- $|H_s + s| \geq \frac{2^n}{N_2 n}$.

And it implies that

$$|(S_{x,k})^c| \geq |H_s + s| \geq \frac{2^{f(k)\log(|x|)}}{N_2 f(k) \log(|x|)}.$$

But this is a contradiction since

$$|(S_{x,k})^c| \leq \frac{2^{f(k)\log(|x|)}}{2^{k\log(|x|)}} \text{ and } N_2 f(k) \log(|x|) \leq 2^{k\log(|x|)}. \qquad \square$$

Notation 5.16. *Given $v \in \{0,1\}^{N_1 f}$ and $v_1, \ldots, v_{N_2 n}$ the output sequence of AKS on input $(v, 2f(k)\log(|x|), f(k)\log(|x|))$ the symbol $AKS(v,i)$ denotes the string v_i.*

Let Ω_1, Ω_2 be the following pair of parameterized languages

$$\Omega_1 = \left\{ (x,v,z,k) : v \in \{0,1\}^{N_1 f} \& z \in \{0,1\}^f \& \bigwedge_{i \leq N_2 n} (x, AKS(v,i) + z, k) \in \Omega \right\}$$

&

$$\Omega_2 = \left\{ (x,v,z,k) : v \in \{0,1\}^{N_1 f} \& z \in \{0,1\}^f \& \bigvee_{i \leq N_2 n} (x, AKS(v,i) + z, k) \in \Omega \right\}.$$

It is easy to obtain, from the last two lemmas, the following corollary.

Corollary 5.17. *If $(x,k) \in L$ and $|x| \geq R_k$, then*

(1) $\forall v \in \{0,1\}^{N_1 f} \exists\, z \in \{0,1\}^f \left((x,v,z,k) \in \Omega_1\right)$.
(2) $\exists v \in \{0,1\}^{N_1 f} \forall\, z \in \{0,1\}^f \left((x,v,z,k) \in \Omega_2\right)$.

We have almost obtained representations of L as $\forall \cdot \exists \cdot \mathcal{C}$ and $\exists \cdot \forall \cdot \mathcal{C}$ languages. It remains to be verified that the languages Ω_1 and Ω_2 are elements of the class $\mathcal{C}$. Unfortunately, if we want to prove this fact we have to suppose that $\mathcal{C}$ is closed under a specific type of parameterized Turing reductions.

Definition 5.18. L is conjunctive reducible to L^* if and only if there exist an fpt algorithm $\mathcal{M}$ and two computable functions f, g such that, on input (x,k), algorithm $\mathcal{M}$ computes a sequence $(x_1,k_1),\ldots,\left(x_{f(k)\log(|x|)},k_{f(k)\log(|x|)}\right)$ that satisfies

(1) $(x,k)\in L \Leftrightarrow \bigwedge_{i\leq f(k)\log(|x|)}(x_i,k_i)\in L^*$.
(2) For all $i\leq f(k)log(|x|)$ we have that $k_i\leq g(k)$.

Definition 5.19. L is disjunctive-reducible to L^* if and only if there exist an fpt algorithm $\mathcal{M}$ and two computable functions f and g such that, on input (x,k), algorithm $\mathcal{M}$ computes a sequence $(x_1,k_1),\ldots,\left(x_{f(k)\log(|x|)},k_{f(k)\log(|x|)}\right)$ that satisfies

(1) $(x,k)\in L \Leftrightarrow \bigvee_{i\leq f(k)\log(|x|)}(x_i,k_i)\in L^*$.
(2) For all $i\leq f(k)log(|x|)$, $k_i\leq g(k)$.

We will say that $\mathcal{C}$ is $\wedge$-*closed* if and only if for all L, if there exists $L^*\in\mathcal{C}$ such that L is conjunctive-reducible to L^*, then $L\in\mathcal{C}$. We define $\vee$-closed analogously.

Fact 5.20. *If $\mathcal{C}$ is $\wedge$-closed, then $\Omega_1\in\mathcal{C}$.*

Fact 5.21. *If $\mathcal{C}$ is $\vee$-closed, then $\Omega_2\in\mathcal{C}$.*

Let g be a computable function such that for all $k\in\mathbb{N}$ the inequality $g(k)\geq N_k$ holds. Given $L\in BP\cdot\mathcal{C}$, we consider the parameterized languages:

(1) $L_{\geq}:=\{(x,k):|x|\geq g(k)\ \&\ (x,k)\in L\}$.
(2) $L_{\leq}:=\{(x,k):|x|\leq g(k)\ \&\ (x,k)\in L\}$.

Fact 5.22. $L_{\leq}\in FPT$.

Note 5.23. Given $\mathcal{C}$ a parameterized class we say that $\mathcal{C}$ is a regular class if and only if given $L\in\mathcal{C}$ there exists a computable function $h:\mathbb{N}\times\mathbb{N}\to\mathbb{N}$ such that for every $(x,k)\in\Sigma^*\times\mathbb{N}$ the query

$$(x,k)\in L?$$

can be solved in time bounded by $h(|x|,k)$. If all the problems contained in $\mathcal{C}$ are computable the class $\mathcal{C}$ is a regular class. Given $L\in\mathcal{C}$ and given $\mathbb{M}$ a Turing machine recognizing L we can define $h(n,k)$ in the following way

$$h(n,k)=\max_{x:|x|=n}\{t_{\mathbb{M}}(x,k)\}$$

where $t_{\mathbb{M}}(x,k)$ denotes the running time of $\mathbb{M}$ on input (x,k). It is important to stress that each one of the parameterized classes considered in this work and in the literature are regular classes.

Theorem 5.24 (parameterized abstract Lautemann-Sipser theorem)**.** *If $\mathcal{C}$ is $\vee$-closed, $\wedge$-closed and $BP \cdot \mathcal{C}$ has the pam property, then $L_{\geq} \in \forall \cdot \exists \cdot \mathcal{C} \cap \exists \cdot \forall \cdot \mathcal{C}$.*

Proof. We prove that $L_{\geq} \in \exists \cdot \forall \cdot \mathcal{C}$, the proof of $L_{\geq} \in \forall \cdot \exists \cdot \mathcal{C}$ is very similar.

We know that

$$(x,k) \in L_{\geq} \Rightarrow \exists v \in \{0,1\}^{N_1 f} \forall z \in \{0,1\}^{f}((x,v,z,k) \in \Omega_2).$$

If $(x,k) \notin L_{\geq}$, then

$$|\,(S_{x,k})^c\,| \geq (1 - 2^{-k\log(|x|)})2^n.$$

Then, if $(x,k) \notin L_{\geq}$ we have that

$$\forall v \in \{0,1\}^{N_1 f} \exists z \in \{0,1\}^{f}(AKS\,(v) + z \subset (S_{x,k})^c)$$

and it implies that if $(x,k) \notin L_{\geq}$, then

$$\neg \exists v \in \{0,1\}^{N_1 f} \forall z \in \{0,1\}^{f}((x,v,z,k) \in \Omega_2).$$

Thus

$$(x,k) \in L_{\geq} \Leftrightarrow \exists v \in \{0,1\}^{N_1 f} \forall z \in \{0,1\}^{f}((x,v,z,k) \in \Omega_2).$$

So, we can conclude that $L_{\geq} \in \exists \cdot \forall \cdot \mathcal{C}$, (since $\Omega_2 \in \mathcal{C}$). □

Corollary 5.25. *Suppose that $BP \cdot \mathcal{C}$ has the pam property*

(1) *If $\mathcal{C}$ is $\wedge$-closed, then $BP \cdot \mathcal{C} \subseteq \forall \cdot \exists \cdot \mathcal{C}$.*
(2) *If $\mathcal{C}$ is $\vee$-closed, then $BP \cdot \mathcal{C} \subseteq \exists \cdot \forall \cdot \mathcal{C}$.*

5.3. Some specific cases

In the following we consider the cases $\mathcal{C} = FPT$ and $\mathcal{C} = W\,[P]$.

Fact 5.26. *FPT is $\wedge$-closed, $\vee$-closed and $BP \cdot FPT$ has the pam property.*

We obtain as a corollary our first parameterized analogue of the Lautemann-Sipser theorem.

Corollary 5.27 (parameterized Lautemann-Sipser theorem)**.** $BP \cdot FPT \subseteq \forall \cdot \exists \cdot FPT \cap \exists \cdot \forall \cdot FPT$.

Now we consider the more difficult case $\mathcal{C} = W\,[P]$.

Proposition 5.28. *$W\,[P]$ is $\vee$-closed.*

Proof. Given $L \in W\,[P]$ there exist a $W[P]$ Turing machine $\mathbb{M}$ and a computable function f such that for all x, k

(1) $(x,k) \in L$ if and only if $\mathbb{M}$ accepts (x,k).

(2) On every run of $\mathbb{M}$, with input (x,k), only the first $f(k)log(|x|)$ moves are nondeterministic.

Let L^* be a language such that L^* is disjunctive-reducible to L, let $\mathcal{M}, g, h$ be the algorithm and the functions in the definition of disjunctive reduction. We will define a $W[P]$ restricted Turing machine $\mathbb{M}^*$ that decides the language L^*. $\mathbb{M}^*$ is the following machine:

on input (x,k)

(1) $\mathbb{M}^*$ guesses $i \in \{1,\ldots,g(k)log(|x|)\}$.
(2) $\mathbb{M}^*$ computes (x_i,k_i) the ith element of the output sequence of $\mathcal{M}$, on input (x,k).
(3) $\mathbb{M}^*$ guesses $y \in \{0,1\}^{f(k_i)\log(|x_i|)}$.
(4) $\mathbb{M}^*$ simulates the deterministic part of the computation of $\mathbb{M}$, on input (x_i,k_i), using the nondeterministic string y.

It is clear that $(x,k) \in L^*$ if and only if $\mathbb{M}^*$ accepts (x,k). □

Proposition 5.29. *If $W[P]$ is $\wedge-$closed, then $W[P]$ is maj-closed.*

Proof. Let L be a language in $W[P]$ and let Σ be a language which is maj-reducible to L. There exist an fpt algorithm $\mathcal{M}$ and two computable functions f and g such that for all (x,k)

(1) algorithm $\mathcal{M}$ computes, on input (x,k), a sequence

$$(x_1,k_1),\ldots,\left(x_{f(k)\log(|x|)},k_{f(k)\log(|x|)}\right).$$

(2) $(x,k) \in \Sigma \Leftrightarrow \bigotimes_{i \leq f(k)\log(|x|)} (x_i,k_i) \in L$.
(3) For all $i \leq f(k)\log(|x|)$ we have that $k_i \leq g(k)$.

Let $\mathbb{M}$ be the following nondeterministic $W[P]$ restricted Turing machine:
On input (x,k)

(1) $\mathbb{M}$ simulates the computation of $\mathcal{M}$ on input (x,k).
(2) Given $(x_1,k_1),\ldots,\left(x_{f(k)\log(|x|)},k_{f(k)\log(|x|)}\right)$ the output of $\mathcal{M}$, on input (x,k), machine $\mathbb{M}$ guesses $v \in \{0,1\}^{f(k)\log(|x|)}$ such that the Hamming weight of v is larger than or equal to $\frac{f(k)\log(|x|)}{2}$.
(3) $\mathbb{M}$ computes the sequence $\{(x_i,k_i) : v_i = 1\}$.
(4) $\mathbb{M}$ verifies that $\bigwedge_{i \leq f(k)\log(|x|):v_i=1} (x_i,k_i) \in L$.

$\mathbb{M}$ is a nondeterministic $W[P]$ machine since we are supposing that $W[P]$ is $\wedge$-closed, furthermore it is easy to verify that $(x,k) \in \Sigma$ if and only if $\mathbb{M}$ accepts (x,k). Then, we have that $\Sigma \in W[P]$. Thus, we have proven that $W[P]$ is maj-closed. □

Corollary 5.30. *If we suppose that $W[P]$ is $\wedge$-closed we have*

(1) *$BP \cdot \exists \cdot FPT$ has the pam property.*
(2) *(Parameterized Strong Lautemann-Sipser theorem) The class $BP \cdot \exists \cdot FPT$ is contained in the class $\forall \cdot \exists \cdot FPT$.*

(3) *(Parameterized Stockmeyer's theorem) For all* $\chi \in \#W[P]$ *the gap problem p-approx* (χ) *belongs to* $\forall \cdot \exists \cdot FPT$*, that is: approximate counting belongs to the second level of the* $PH[P]$ *hierarchy.*

Las corollary shows that if we suppose that $W[P]$ is $\wedge$-closed, then suitable parameterized analogues of Lautemann-Sipser and Stockmeyer theorems can be established. We finish this work stating a question that arises from Corollary 5.30.

Question 1. *Is* $W[P]$ $\wedge$*-closed?*

Acknowledgements. Dedicated to Mamadimitriou and Hijodimitriou, thanks to Joerg Flum. This work was developed with the financial support of VIE-UIS. The author especially thanks the anonymous referee for carefully reading the manuscript and for providing him with many helpful suggestions which have greatly improved his writing.

References

[1] M. Agrawal, N. Saxena and N. Kayal, *P*RIMES is in *P*. *Annals of Math.* **160** (2004) 781–793.
[2] V. Arvind and V. Raman, Approximation algorithms for some parameterized counting problems, in *Proceedings of the 13th Annual International Symposium on Algorithms and Computation,* edited by P. Bose and P. Morin. *Lect. Notes Comput. Sci.* **2518** (2002) 453–464.
[3] R.G. Downey and M.R. Fellows, *Parameterized Complexity.* Springer-Verlag (1999).
[4] J. Flum and M. Grohe, The parameterized complexity of counting problems. *SIAM J. Comput.* **33** (2004) 892–922.
[5] J. Flum and M. Grohe, *Parameterized Complexity Theory.* Springer-Verlag (2006).
[6] O. Goldreich, *Randomized methods in Computation.* Manuscript (2001) http://www.wisdom.weizmann.ac.il/~oded/rnd.html.
[7] C. Lautemann, BPP and the Polynomial Hierarchy. *Inform. Process. Lett.* **17** (1983) 215–217.
[8] J.A. Montoya, *On parameterized Counting.* Ph.D thesis, Freiburg University (2008).
[9] J.A. Montoya, The parameterized complexity of probability amplification. *Inform. Process. Lett.* **109** (2008) 46–53.
[10] M. Muller, *Randomized approximations of parameterized counting problems. Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC'06).* Lect. Notes Comput. Sci. **4169** (2006) 50–59.
[11] C.H. Papadimitriou, *Computational Complexity.* Addison-Wesley (1994).
[12] L. Stockmeyer, On approximation Algorithms for $\#P$. *SIAM J. Comput.* **14** (1985) 849–861.
[13] S. Toda, PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.* **20** (1991) 865–877.
[14] L.G. Valiant, The complexity of computing the permanent. *Theoret. Comput. Sci.* **8** (1979) 189–201.
[15] L.G. Valiant, The complexity of enumeration and reliability problems. *SIAM J. Comput.* **8** (1979) 410–421.

Communicated by Ch. Choffrut.
Received April 7, 2008. Accepted February 1, 2011.