Toda's Theorem.
Part 2: $BPP^{\oplus P} \subseteq P^{PP}$

Dmitry Shiryaev

JASS 2006

Abstract

Remember, Toda's Theorem states that $PH \subseteq P^{PP}$, where PH is the polynomial hierarchy and PP is the class of sets accepted by polynomial-time-bounded probabilistic Turing machines with two-sided unbounded error probability. Statement of the theorem can be proved by obtaining the chain of inclusions:

$$PH \subseteq BPP^{\oplus P} \subseteq PP^{\oplus P} \subseteq P^{\#P} = P^{PP}$$

First inclusion was proved in the previous talk, here we will prove two rest inclusions and an equality.
So, proof will be divided in 3 steps: first, we remember some basic definitions, such as BPP, PP, $\oplus P$ and define some new class $\#P$. In that part we will also see, that first inclusion is evident, as $BPP \subseteq PP$. Second step is to show that $\#P$-oracle isn't more powerful than PP-oracle - and we get an equality $P^{\#P} = P^{PP}$. After all, the last and the main part of the paper is about the inclusion between $PP^{\oplus P}$ and $P^{\#P}$.

1 Remindings and Definitions

Let's remember the definitions of some classes, that will appear in our proof. PP is the class of sets, accepted by polynomial-time-bounded probabilistic Turing machines with two-sided *unbounded* error probability. BPP - the subclass of PP with two-sided *bounded* error probability. These definitions can be written as follows:

Definition 1. L is in PP iff there exists polynomial-time bounded NTM M, such that

$$x \in L \iff P\{M(x) = 1\} > \frac{1}{2},$$

Definition 2. L is in BPP iff there exists polynomial-time bounded NTM

M, such that

$$\begin{cases} x \in L \Rightarrow P\{M(x) = 1\} > \dfrac{3}{4} \\ x \notin L \Rightarrow P\{M(x) = 0\} > \dfrac{3}{4}. \end{cases}$$

Definition 3. For the NTM M we define $acc_M(x)$ as the number of accepting computation paths of M on input x. Then we can define the class $\oplus P$ $(parity - P)$.

Definition 4. L is in $\oplus P$ iff there exists polynomial-time bounded NTM M, such that

$$x \in L \iff 2 \mid acc_M(x).$$

PP, BPP, $\oplus P$, defined above, are classes of languages. Now we'll define the $\#P$, and it won't be a class of languages, but a class of functions.

Definition 5. Here's the definition of $sharp - P$ function class:

$\#P = \{f \colon \Sigma^* \to \mathbb{N} \cup \{0\} \mid \exists$ polynomial-time NTM M_f, such that $\forall x\; f(x) = acc_M(x)\}$.

Remark, $\#P$ is the function class. Nevertheless, we can use it as an oracle. Asking an oracle from the class $\#P$ means asking NTM of number of it's accepting computation paths.

Proposition 1. $BPP^{\oplus P} \subseteq PP^{\oplus P}$
Proof. It's evident that $BPP \subseteq PP$. It simply leads to our conclusion. □

2 The proof of $P^{\#P} = P^{PP}$

Proposition 2. $P^{\#P} = P^{PP}$
Proof. 1)$P^{\#P} \subseteq P^{PP}$. This part is quite evident. Without loss of generality, we can say that lengths of all branches in computation tree of our NTM are equal. We can ask oracle about the number of accepting computation paths and we want to be able to ask oracle whether it is more or less than a half of all paths. We know the number of all paths and the number of all accepting paths - so we can find whether it is more or less than a half.

2) $P^{\#P} \supseteq P^{PP}$. We have an oracle from $\#P$, it is realized by NTM M. Let's define the language $L = \{(x \in \Sigma^*, y \in \mathbb{N}) : acc_M(x) > y\}$. Imagine we'll show that $L \in PP$. Then, with access to L as an oracle, using binary search, we'll find $acc_M(x)$ in logarifmic time of $2^{poly(n)}$, i.e. in the polynomial time. So, it remains only to show that $L \in PP$.

To show that $L \in PP$ we'll construct NTM M' (with inputs (x, y) where $x \in \Sigma^*$, $y \in \mathbb{N}$), such that number of accepting paths of it is more than $\frac{1}{2}$ whenever $acc_M(x) > y$. Existance of NTM with this property shows us exactly that $L \in PP$, due to the definition of L. M' is defined as follows: on it's first step it makes non-deterministic choise of two branches. While one branch runs

M, another branch makes fictious computations, such that number of paths is the same as M has, and the number of rejecting paths is exactly y. It's easy to construct this branch: let machine simply look at the binary representation of y.

Now let's find the number of accepting computation paths of $M^{'}$. There are $acc_M(x)+(all_M - y)$ of them, where all_M is the number of all paths of M. Then the probability of $M^{'}$ accepting is $\frac{acc_M(x)+(all_M-y)}{2\cdot all_M} = \frac{1}{2} + \frac{acc_M(x)-y}{2\cdot all_M}$ which is greater then $\frac{1}{2}$ whenever $acc_M(x) > y$. This ends our proof. □

3 The proof of $PP^{\oplus P} \subseteq P^{\#P}$

Preparing to the proof, we need the following number theory lemma, which can be easily prooved by induction.

Lemma 1.
Define polynoms $s_i(z)$ recurrently:

$$s_0(z) = z,$$
$$s_i(z) = 3s_{i-1}(z)^4 + 4s_{i-1}(z)^3.$$

Then the following property holds for all natural numbers i and z:
if z is odd, then $2^{2^i} | s_i(z)$,
if z is even, then $2^{2^i} | (s_i(z) + 1)$.

In our main proof we won't need the construction of $s_i(z)$, we'll need only the property, that the lemma states. Now we come to the main part of this paper.

Proposition 3. $PP^{\oplus P} \subseteq P^{\#P}$
Proof. Let's take the language $L \in PP^{\oplus P}$. L can be defined as follows:

$$L = \{x \mid \exists A \in P^{\oplus P} : \Big|\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in A\}\Big| > \frac{1}{2} \cdot 2^{p(|x|)}\},$$

where p is a polynom. Define $l(x) := \lceil log(p(|x|)) + 1 \rceil$. First of all, remember, that $\oplus P^{\oplus P} = \oplus P$ was proved in the previous talk, when discussing the first part of Toda's Theorem. It immediately implies $P^{\oplus P} = \oplus P$ and so we can change the class of A in the definition of L:

$$L = \{x \mid \exists A \in \oplus P : \Big|\{y \in \{0,1\}^{p(|x|)} \mid (x,y) \in A\}\Big| > \frac{1}{2} \cdot 2^{p(|x|)}\}.$$

Let $\widetilde{A}$ be the NTM, corresponding to the language A. Our goal is to show that $L \in P^{\#P}$. We'll construct a machine N such that using it only once as an $\#P$-oracle will help us to find whether x belongs to L or not. The construction of N will be divided in 2 steps. First, with the help of Lemma 1, we'll construct

NTM M, such that number of it's accepting computation paths will be strongly related to the belonging of input to the A. Then we'll construct N as NTM, guessing y and then running M on (x, y) - and ask an oracle to the number of accepting paths of N. We'll see, that this number $mod(2^p)$ will be exactly the number of proper y's such that $(x, y) \in A$. This will end our proof.

Here comes the construction of the NTM M. Define polynom $q(z) = (s_{l(x)}(z))^2$, where s is the polynom defined in Lemma 1. Let $q(z) = q_1 \cdot z^{c_1} + \ldots + q_m \cdot z^{c_m}$, where $q_1, \ldots, q_m$ are all non-zero coefficients of $q(z)$. Then machine M will make the non-deterministic choice between m branches - call them $1, 2, \ldots, m$. In each i's branch machine makes it's second choice between z_i subbranches. In each of that subbranches machine several times runs $\widetilde{A}$, one after another, number of times is c_i. Every next run of $\widetilde{A}$ starts iff the previous accepted.

Why do we need such a strange construction of M? Let's find the number of it's accepting computation paths. Machine M takes x as an input, and accepts iff all $\widetilde{A}$'s on it's path of computation accepted. If $acc_{\widetilde{A}}(x) = z$, then $acc_M(x) = q(z) = (s_{l(x)}(z))^2$ due to the construction of M. Now we remember the statement of Lemma 1, and immediately get that if machine $\widetilde{A}$ has odd number of accepting paths on fixed input x, then M has 1 modulo $2^{2^{l(x)}}$ accepting computation paths, else 0 the same modulo. That is the only property of M, that we'll need in future.

Now the time has come to construct the machine N. First, it makes non-deterministic choice between 2^p branches, in other words it guesses y - the computation path for machine $\widetilde{A}$. Then it simply runs $M(x, y)$. Each choice of y, such that $(x, y) \notin A$ brings us 0 modulo $2^{2^{l(x)}}$ accepting computation paths of N, and for every y such that $(x, y) \in A$ number of corresponding accepting computation paths of N will be 1 modulo $2^{2^{l(x)}}$. This implies, that the number of accepting computation paths of N (which will be found by question to an oracle) modulo $2^{2^{l(x)}}$ is exactly the number of proper y's - such y's, that $(x, y) \in A$. Remember that $l(x) := \lceil log(p(|x|)) + 1 \rceil$ and so $2^{l(x)} > p$. It implies that the number of proper y's can be found simply by asking oracle of the number of accepting computation paths of N and taking this number modulo 2^p. The only remaining thing is to compare it with $\frac{1}{2} \cdot 2^p$. $\square$