CS 6810: Theory of Computing **Fall 2023**

Lecture 18: Oct 24, 2023

Lecturer: Eshan Chattopadhyay *Scribe: Haripriya Pulyassary*

1 Introduction

Randomness is useful in designing algorithms. An excellent example of this is the problem of testing if a polynomial polynomial $p \in \mathbb{F}_q[x_1, \ldots, x_n]$ is the zero polynomial. While it is difficult to design a deterministic algorithm for this problem (as we would have to check the coefficient of each monomial), the Schwartz-Zippel Lemma asserts that a randomly selected x will not be a root of p with high probability. This observation yields the following simple randomized algorithm for the problem: randomly choose x and evaluate $p(x)$; if $p(x) = 0$, return "Yes", otherwise return "No".

In this module, we will study the setting where Turing machines have access to randomized bits.

2 Probabilistic Turing Machines

There are two equivalent definitions for probabilistic Turing machines.

I. Let M be a multitape Turing machine with two transition functions δ_0, δ_1. We also assume that M has access to random coins. At each step M chooses between one of δ_0, δ_1 using the random bits. When M requires a random bit, it goes to a special state; at this point the random coin is flipped and its value is reported to M.

   We assume that the coin is always fair. It is important to note here that we cannot access the history of coin flip realizations at a later step. That is, if the machine requires access to the realized value of the 10th coin flip at a later step, then this value must be stored on the tape. Hence, while we could think of a tape with random bits written on it (instead of a coin being flipped each time), we must ensure that M cannot see past history on this tape. One workaround that ensures this is that we only allow the pointer to move to the right on the random bits tape.

II. M is a Turing Machine which takes in two inputs: (x, r) where x is the input it is working on, and r is the randomness supplied to it. We assume that $r \sim_{Uniform} \{0,1\}^{poly(|x|)}$.

Both definitions are equivalent as in, given one, it is possible to simulate the other. However, we will often find it easier to work with the second definition.

3 Complexity classes

In this section, we define some (time) complexity classes for this setting.

Definition 3.1 (Probabilistic Polynomial Time (PP)). *$L \in PP$ if there exists M that runs in polynomial time (for all random choices), and*

- *If $x \in L$, then $\Pr[M(x,r) = 1] > \frac{1}{2}$*
- *If $x \notin L$, then $\Pr[M(x,r) = 1] \leq \frac{1}{2}$.*

Notice that this definition just requires M to perform better than the naive algorithm of accepting with a probability of $\frac{1}{2}$.

Definition 3.2 (Bounded Probabilistic Polynomial Time (BPP)). *$L \in BPP$ if there exists M that runs in polytime for any random choice, and*

- *If $x \in L$, then $\Pr[M(x,r) = 1] > \frac{2}{3}$.*
- *If $x \notin L$, then $\Pr[M(x,r) = 1] < \frac{1}{3}$.*

Definition 3.3 (Randomized Polynomial (RP)). *$L \in RP$ if there exists a probabilistic Turing Machine M that runs in polynomial time for any random choice, and*

- *If $x \in L$, then $\Pr[M(x,r) = 1] \geq \frac{1}{2}$.*
- *If $x \notin L$, then $\Pr[M(x,r) = 1] = 0$.*

Definition 3.4 (co-RP). *$L \in RP$ if there exists a probabilistic Turing Machine M that runs in polynomial time for any random choice, and if $x \in L$, then $\Pr[M(x,r) = 1] = 1$. If $x \notin L$, $\Pr[M(x,r) = 1] \leq \frac{1}{2}$.*

A helpful mnemonic is *RP* does not allow *false positives*, and *co-RP* does not allow *false negatives*.

There are two classes of randomized algorithms: Monte-Carlo (these algorithms can be wrong with some probability bound) and Las Vegas (such algorithms cannot output a wrong answer, but allowed to take very long time on some inputs, as long as the expected time (over r) is polynomial). In a sense, one can view the following complexity class, ZPP, as pertaining to languages with "Las Vegas"-esque probabilistic Turing machines.

Definition 3.5 (Zero-error Probabilistic Polynomial (ZPP)). *$L \in ZPP$ if M is a probabilistic TM such that for all x, $M(x) = L(x)$ with probability 1 and $\max_{x \in \{0,1\}^n} \mathbb{E}_r[T(x)] = poly(n)$.*

Proposition 3.6. *$ZPP = RP \cap co{-}RP$.*

Proof. We first show that $RP \cap co{-}RP \subseteq ZPP$. Let $L \in RP \cap co - RP$. Then, there exists Turing machines M_{RP} and M_{coRP} such that both decide L and satisfy the requirements of RP and $co - RP$ respectively. Consider the following algorithm:

```
while True do
    Run M_RP(x, r1) and M_co-RP(x, r2) where r1, r2 ~ {0,1}^poly(|x|).
    if M_RP(x, r1) = 1 then
        break and return x ∈ L
    end
    if M_co-RP(x, r2) = 0 then
        break and return x ∉ L
    else
        Repeat
    end
end
```

Note that this algorithm will always output $L(x)$ (as $M_{RP}(x,r) = 1$ only if $x \in L$ and $M_{co-RP}(x,r) = 0$ only if $x \notin L$). It remains to analyze the expected runtime. The expected number of times we repeat the while loop is at most $\frac{1}{1-\frac{1}{2}} = 2$; this is the expected value of a geometric random variable with $p = 1 - \frac{1}{2}$, which is a lower bound on the probability that $x \in L$ and $M_R(x, r_1) = 1$ or $x \notin L$ and $M_{co-RP}(x, r_2) = 0$. So the expected runtime is at most $2 \cdot (T_{RP} + T_{co-RP})$. Thus, $L \in ZPP$.

We now prove the other direction. Suppose $L \in ZPP$. We first show that $L \in RP$. Suppose M is a Turing machine such that $M(x) = L(x)$ with probability 1 for all x, and the expected running time of M is $T(n) = poly(n)$. We construct M' as follows: on an input x, we simulate M for $2T(x)$ steps. If an answer is obtained, we return $M(x)$. Otherwise, if no answer is obtained in $2T(x)$ steps, we output 0. If $x \notin L$, $M'(x) = 0$. If $x \in L$, the probability that $M'(x) = 0$ is precisely the probability that M does not terminate within $2T(x)$ steps; this is at most $1/2$ by Markov's inequality. So M' is a probabilistic Turing machine satisfying the conditions of RP and hence $L \in RP$.

We can use an analogous argument to show that $L \in co - RP$. We construct M'' as follows: on an input x, we simulate M for $2T(x)$ steps. If an answer is obtained, we return $M(x)$. Otherwise, if no answer is obtained in $2T(x)$ steps, we output 1. If $x \in L$, $M''(x) = 1$. If $x \notin L$, the probability that $M''(x) = 1$ is precisely the probability that M does not terminate within $2T(x)$ steps; this is at most $1/2$ by Markov's inequality. So M'' is a probabilistic Turing machine satisfying the conditions of RP and hence $L \in co - RP$. $\square$

4 Error Reduction

Definition 4.1 (RP^ε)**.** *$L \in RP^\varepsilon$ if there exists a probabilistic Turing Machine M that runs in polynomial time for any random choice, if $x \in L$, $\Pr[M(x,r) = 1] \geq 1 - \varepsilon$. If $x \notin L$, $\Pr[M(x,r) = 1] = 0$.*

It is easy to see that $RP^\varepsilon \subseteq RP$, when $\varepsilon \leq \frac{1}{2}$.

Proposition 4.2. *$RP \subseteq RP^\varepsilon$ for any $1/2 \geq \varepsilon \geq 2^{-poly(|x|)}$.*

Proof. Suppose $L \in RP$, and let M be the probabilistic Turing machine asserted by the definition of RP. We define the probabilistic Turing machine $\tilde{M}$ as follows. To run $\tilde{M}$ on input x, sample $r_1, \ldots, r_t$, run $M(x, r_1), \ldots, M(x, r_t)$, and return $\vee(M(x, r_1), \ldots, M(x, r_t))$ (we define t later). If $x \notin L$, $\tilde{M}$ outputs 0 with probability 1. If $x \in L$, the probability that $\tilde{M}$ outputs 0 is the probability that $M(x, r_i) = 0$ for all $i = 1, \ldots, t$; this probability is at most $\frac{1}{2^t}$. If we define $t := \log(1/\varepsilon)$, the probability that $\tilde{M}(x) = 0$ and $x \in L$ is at most ε. Furthermore, as long as $\varepsilon = 2^{-poly(|x|)}$, $\tilde{M}$ runs in polynomial time. $\square$

Definition 4.3 (BPP^ε)**.** *$L \in BPP^\varepsilon$ if there exists M that runs in polytime for any random choice, and if $x \in L$, then $\Pr[M(x,r) = L(x)] \geq 1 - \varepsilon$.*

Proposition 4.4. *$BPP = BPP^\varepsilon$, for any $1/3 > \varepsilon \geq 2^{-poly(|x|)}$.*

Proof. Suppose $L \in BPP$, and let M be the probabilistic Turing machine asserted by the definition of BPP. We define $\tilde{M}$ as follows. To run $\tilde{M}$ on x, sample $r_1, \ldots, r_t$, run $M(x, r_1), \ldots, M(x, r_t)$, and return $Majority(M(x, r_1), \ldots, M(x, r_t))$.

Define $X_i = M(x, r_i)$, $X = \sum_{i=1}^t X_i$. If $x \in L$, $\mathbb{E}[X] \geq \frac{2}{3}t$. Using Hoeffding's Inequality, it follows that

$$\Pr[\sum_{i=1}^t X_i < \frac{t}{2}] \leq \Pr[|X - \mathbb{E}[X]| > \frac{t}{6}] \leq 2^{-\Omega(t)} \leq \varepsilon,$$

where $t = O(\log(1/\varepsilon))$ An analogous argument shows that if $x \notin L$, $\Pr[\sum_{i=1}^t X_i > \frac{t}{2}] \leq \varepsilon$. $\square$