18.405J/6.841J Advanced Complexity Theory March 5, 1998

Lecture 8

Lecturer: Madhu Sudan *Scribe: Anna Lysyanskaya*

1 Amplification of BPP

Recall that in the last lecture we viewed **BPP** as the class of languages for which there exists a probabilistic polynomial time Turing machine such that:

- for all strings in the language, the machine will accept with probability at least $\frac{2}{3}$;
- for all strings not in the language, the machine will accept with probability at most $\frac{1}{3}$.

Now we are going to present a more broad definition of **BPP**. Instead of fixed constants $\frac{2}{3}$ and $\frac{1}{3}$ we will use parameters c and s.

Definition 1 *A language L is in $\mathbf{BPP}_{c,s}$ $(0 \leq s(n) < c(n) \leq 1, \forall n \in \mathbb{N})$ if there exists a probabilistic polynomial time Turing Machine M such that:*

1. $\forall w \in L, Pr[M \text{ accepts } w] \geq c(|w|)$
2. $\forall w \notin L, Pr[M \text{ accepts } w] \leq s(|w|)$

The function $c(n)$, where n is the length of the input to M, is called the *completeness* and the function $s(n)$ is called the *soundness* of the probabilistic polynomial time Turing machine. It turns out that the exact values of c and s don't matter because of the following theorem:

Theorem 2 (Amplification of BPP) *For all choices of polynomially computable functions $c(n)$ and $s(n)$, $\{0,1\}^n \mapsto [0,1]$, such that there exists a polynomial $Q(n)$ such that $\forall n$ $c(n) - s(n) \geq \frac{1}{Q(n)}$, and $m = O(1)$,*

$$\mathbf{BPP}_{c,s} = \mathbf{BPP}_{1-\frac{1}{2^{n^m}}, \frac{1}{2^{n^m}}} .$$

Proof We will show that, given a **BPP** machine M with completeness $c(n)$ and soundness $s(n)$, we can construct a **BPP** machine for the same language, with completeness $1 - \frac{1}{2^{n^m}}$, soundness $\frac{1}{2^{n^m}}$, for any $m = O(1)$.

Consider a machine M' that does the following:

1. Run M on k independently chosen random tapes $R_1, ..., R_k$.
2. Accept if the number of times M accepted is greater than or equal to $k(\frac{c(n)+s(n)}{2})$.

Let X_i be the indicator random variable for the event that M accepts w on random string R_i. By the definition of $\mathbf{BPP}_{c,s}$, we know that $w \in L \Rightarrow \mathrm{E}[X_i] \geq c(n)$, $w \notin L \Rightarrow \mathrm{E}[X_i] \leq s(n)$. Let us estimate the completeness and soundness values of M':

$$c_{M'}(n) = 1 - \Pr\left[\sum_{i=1}^{k} X_i < k\left(\frac{c(n)+s(n)}{2}\right) \mid w \in L\right]$$

$$s_{M'}(n) = \Pr\left[\sum_{i=1}^{k} X_i > k\left(\frac{c(n)+s(n)}{2}\right) \mid w \in L\right]$$

Here Chernoff bounds give us a powerful tool. Chernoff bounds guarantee that for any k independent identically distributed random variables $X_1, ..., X_k$, taking on values in the interval $[0,1]$, with expected value $\mathrm{E}[X_i] = p$, for any value of $\epsilon \in [0,1]$,

$$\Pr\left[\sum_{i=1}^{k} X_i < kp(1-\epsilon)\right] \leq e^{-\frac{\epsilon^2}{3}pk}$$

$$\Pr\left[\sum_{i=1}^{k} X_i > kp(1+\epsilon)\right] \leq e^{-\frac{\epsilon^2}{3}pk}$$

Using Chernoff bounds, setting ϵ so that $\epsilon \leq \frac{c(n)-s(n)}{2c(n)}$ and $\epsilon \leq \frac{c(n)-s(n)}{2s(n)}$ (so that $\frac{c(n)-s(n)}{2} \leq c(n)(1-\epsilon)$ and $\frac{c(n)-s(n)}{2} \geq s(n)(1+\epsilon)$), and setting k such that $k \geq \frac{3n^m}{c(n)\epsilon^2}$ and $k \geq \frac{3n^m}{s(n)\epsilon^2}$ (so that $\frac{\epsilon^2}{3}c(n)k \geq n^m$ and $\frac{\epsilon^2}{3}s(n)k \geq n^m$), we get that

$$c_{M'}(n) = 1 - \Pr\left[\sum_{i=1}^{k} X_i < k\left(\frac{c(n)+s(n)}{2}\right) \mid w \in L\right] \geq 1 - e^{-\frac{\epsilon^2}{3}c(n)k} \geq 1 - 2^{-n^m}$$

$$s_{M'}(n) = \Pr\left[\sum_{i=1}^{k} X_i > k\left(\frac{c(n)+s(n)}{2}\right) \mid w \notin L\right] \leq e^{-\frac{\epsilon^2}{3}s(n)k} \leq 2^{-n^m}$$

Since $c(n)$ and $s(n)$ differ at least by the inverse of polynomial $Q(n)$, k is polynomial in n. Therefore M' is a probabilistic polynomial time Turing machine with the desired properties. ■

2 BPP is in P/poly

Theorem 3 **BPP** $\subseteq$ **P/poly**.

Proof Consider an arbitrary language $L \in$ **BPP**. By definition of **BPP**, we know that for any fixed string w of length n, there is a random string R of length polynomial in n such that a **BPP** machine gives the right answer if it is given R on its random tape. The idea is to find a string R that is good for all input strings of size n. By aplification of **BPP**, we know that there exists a Turing machine $M \in \mathbf{BPP}_{1-\frac{1}{2^{n^2}}, \frac{1}{2^{n^2}}}$ that decides L. Note that the probability of making an error is much smaller than the number of possible inputs.

Classify all possible random strings R as follows:

- R is *bad* for a specific input w if $M(x, R)$ gives the wrong answer;
- R is *bad* if there exists an input string w for which R is *bad*.
- R is *good* otherwise.

For a fixed input w, $\Pr[R \text{ is } \textit{bad} \text{ for } w] \leq \frac{1}{2^{n^2}}$. Then, overall,

$$\Pr[R \text{ is } \textit{bad}] \leq \sum_{w \in \{0,1\}^n} \Pr[r \text{ is } \textit{bad} \text{ for } w] \leq 2^n \frac{1}{2^{n^2}} < 1 \ .$$

Therefore,

$$\Pr[R \text{ is } \textit{good}] = 1 - \Pr[R \text{ is } \textit{bad}] > 0 \ .$$

Therefore, there exists a polynomial size advice string for any input of length n. ■

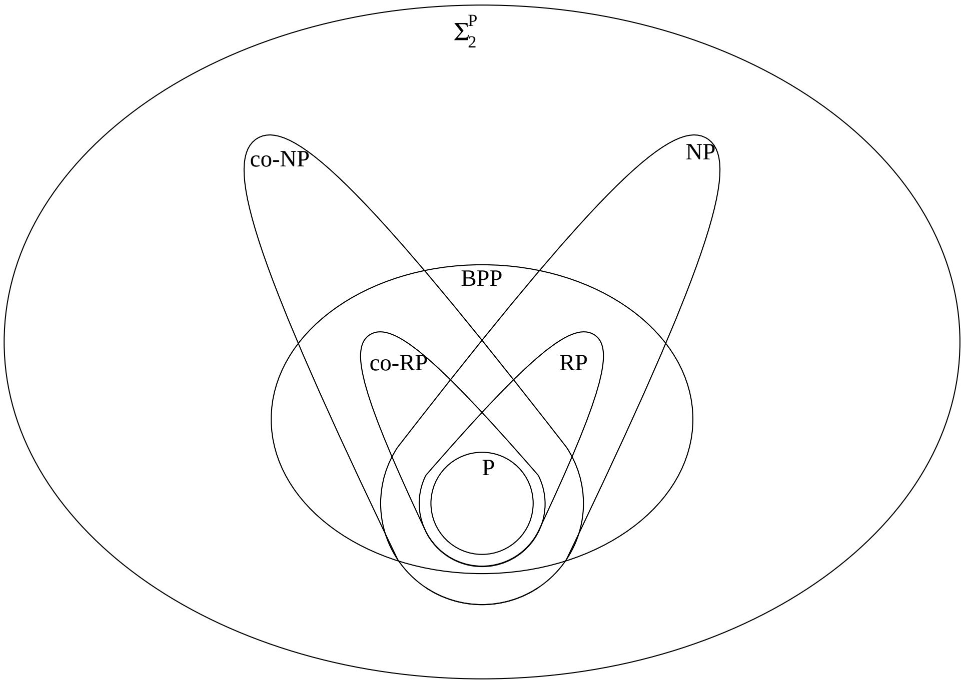

Figure 1: The containments of some complexity classes

3 BPP is in Σ_2^P

It will also be useful to look at how **BPP** fits into the polynomial hierarchy. Figure 1 illustrates what is known about the relationship between **P**, **RP**, **coRP**, **BPP**, **NP** and **coNP**. It also illustrates the following theorem:

Theorem 4 **BPP** $\subseteq \Sigma_2^P$

Proof Suppose $L \in$ **BPP**. We wish to show that there is a Σ_2^P machine that decides L. More specifically, we will show that there is a deterministic polynomial time Turing machine $M(x, y, z)$ such that

$$x \in L \Leftrightarrow \exists y \text{ such that } \forall z \ M(x, y, z) = 1$$

$$x \notin L \Leftrightarrow \forall y \exists z \text{ such that } M(x, y, z) = 0$$

This proof is originally due to Michael Sipser, later it was impoved by Clemens Lautemann.

Say A is a **BPP**machine that uses $Q(n)$ random bits with completeness $c(n) = \frac{1}{2}$ and soundness $s(n) = \frac{1}{3Q(n)}$, where n denotes the length of the input string and $Q(n)$ is a polynomial. A can also be viewed as a deterministic machine that takes a random tape as an input and behaves deterministically ever after. Consider the space R of all random strings that can be on A's random tape on input of size n. There are $2^{Q(n)}$ strings in R.

Define the shift operation $F_s(y)$ of $y \in R$ by $s \in R$ as $F_s(y) = y \oplus s$. A shift operation is random if its operator s is chosen uniformly at random.

Imagine a new machine, $A'(x, y, S)$, where S is a sequence of random shift operators $(s_1, \ldots, s_k)$, $y \in R$, and x is the original input whose membership in L_A is being tested. A' is a deterministic machine such that:

$$A'(x, y, S) = 1 \Leftrightarrow \exists s_i \in S, A \text{ accepts } x \text{ with } y \oplus s_i \text{ on its random tape.}$$

If $x \notin L_A$, and a specific S is chosen at random, then

$$\begin{aligned}
\Pr_{y\in R}[A'(x,y,S)=1] &= \Pr_{y\in R}[\exists s_i \in S \text{ such that } A(x, y \oplus s_i)] && (1)\\
&\le \sum_{i=1}^{k} \Pr_{y\in R}[A(x, y\oplus s_i)] && (2)\\
&\le k \Pr_{y\in R}[A(x,y)] && (3)\\
&\le \frac{k}{3Q(n)} && (4)
\end{aligned}$$

(1) follows from the way A' was defined. (2) is a fact from probability. (3) follows because $y \oplus s_i$ is distributed uniformly over R, just as y. (4) follows from the way we defined A.

Picking $k \le 2Q(n)$ will make sure that

$$\Pr_{y\in R}[A'(x,y,S)=1] \le \frac{2}{3} \ ,$$

and that means that

$$\Pr_{y\in R}[A'(x,y,S)=0] > \frac{1}{3} \ ,$$

which implies that, if $x \notin L_A$, then for any $S \in R^k$ there exists a $y \in R$ such that $A'(x,y,S)=0$.

If $x \in L_A$, and a specific y is chosen at random, consider the probability that A' will reject x when given y and some random S:

$$\begin{aligned}
\Pr_{S\in R^k}[A'(x,y,S)=0] &= \Pr_{S\in R^k}[\forall s_i \in S, A(x, y\oplus s_i)=0] && (1)\\
&= \Pr_{S\in R^k}[\forall s_i \in S, A(x, s_i)=0] && (2)\\
&\le (\frac{1}{2})^k && (3)
\end{aligned}$$

(1) follows from the way A' was defined. (2) is true because s_i is distribited identically to $s_i \oplus y$. (3) follows from the fact that the $c(n) \ge \frac{1}{2}$.

Finally, consider the probability that there exists a $y \in R$ such that $A'(x,y,S)$ will reject:

$$\begin{aligned}
\Pr_{S\in R^k}[\exists y \text{ such that } A'(x,y,S)=0] &\le \sum_{y\in\{0,1\}^{Q(n)}} \Pr_{S\in R^k}[A'(x,y,S)=0] && (1)\\
&= 2^{Q(n)} \Pr_{S\in R^k}[A'(x,y,S)=0] && (2)\\
&\le \frac{2^{Q(n)}}{2^k} \ . && (3)
\end{aligned}$$

Let $k = 2Q(n)$. Then

$$\begin{aligned}
\Pr_{S\in R^k}[\forall y \in R,\ A'(x,y,S)=1] &= 1 - \Pr_{S\in R^k}[\exists y \text{ such that } A'(x,y,S)=0] && (1)\\
&\ge 1 - \frac{2^{Q(n)}}{2^{2Q(n)}} && (2)\\
&= 1 - \frac{1}{2^{Q(n)}} && (3)\\
&> 0 \ . && (4)
\end{aligned}$$

This implies that if $x \in L_A$, then there exists a sequence of shifts $S \in R^k$, such that for all $y \in R$, $A'(x,y,S)$ accepts.

To summarize, we know that

$$x \in L_A \Rightarrow \exists S \in R^k \text{ such that } \forall y \in R,\ A'(x,y,S) = 1\ ,$$

$$x \notin L_A \Rightarrow \forall S \in R^k\ \exists y \in R \text{ such that } A'(x,y,S) = 0\ .$$

Therefore,

$$x \in L_A \Leftrightarrow \exists S \in R^k \text{ such that } \forall y \in R,\ A'(x,y,S) = 1\ ,$$

and so a Σ_2^P machine can decide L_A by guessing S, guessing all y and checking that $A'(x,y,S) = 1$. ■

We can use some steps of this proof to learn more facts about **BPP**. For example, consider the following:

Proposition 5 $\mathbf{BPP} \subseteq \mathbf{RP}^{\mathbf{coRP}}$.

Note: This notation is not precise, because no **coRP**-complete language is known. What it means is that, if given an oracle for the right **coRP** language, an oracle **RP** machine can decide any **BPP** language.

Proof Idea Imagine a machine that accepts if it can find a string $S \in R^k$ such that its oracle says that $\forall y \in R,\ A'(x,y,S) = 1$. Then this oracle machine is an **RP** machine, by the definition of **RP**. Now, the claim says that the oracle it uses decides a language in **coRP**. That is so, because this oracle accepts if it cannot find a $y \in R$ such that $A'(x,y,S) \neq 1$, which, by definition of **coRP**, defines a **coRP** language. ■