CS 221 Computational Complexity, Lecture 9 Feb 20, 2018

Lecture 9

Instructor: Madhu Sudan *Scribe: Marcus Comiter*

1 Today

1. BPP $\subseteq$ P/poly
2. BPP $\subseteq$ PH
3. One hope is that maybe BPP=P?

2 Definition of $\text{BPP}_{C,S}$

We define Promise-$\text{BPP}_{C,S}$ as the following:

$$\text{Promise-BPP}_{C,S} = L : (L_Y, L_N)$$

We say that a language $L \in \text{BPP}_{C,S}$ if there exists a polynomial time $M(.,.)$ such that $\forall x$:

$$x \in L_Y \Rightarrow Pr_y[M(x,y) = 1] \geq C$$

$$x \in L_N \Rightarrow Pr_y[M(x,y) = 1] \leq S$$

where C is completeness and S is soundness.

We can define different variants of BPP, namely "Strong BPP", "BPP", and "Weak BPP", in terms of C and S. We define them as follows:

$$\text{Strong BPP}_{C,S} : C = 1 - \exp(-\text{poly}(n)); s = \exp(-\text{poly}(n))$$

$$\text{Regular BPP}_{C,S} : C = \frac{2}{3}; s = \frac{1}{3}$$

$$\text{Weak BPP}_{C,S} : C - S = \frac{1}{\text{poly}(n)}$$

3 Amplification Theorem

The amplification theorem states that Strong-BPP = (Regular) BPP = Weak BPP. The first application that we will look at of the amplification theorem is to show that BPP $\subseteq$ P/poly. The idea of the proof is that we hope to be able to fix an n and $x \in \{0,1\}^n$ such that $\exists\ y$ for which $M(x,y) = L(x) \forall x \in \{0,1\}^n$.

To show this proof for this, we will work with "Strong BPP" with the following choices for C and S:

$$C = 1 - \frac{1}{2^{n+1}}$$

$$S = \frac{1}{2^{n+1}}$$

Using this, we have $\forall x$:

$$Pr_y[M(x,y) \neq L(x)] \leq \frac{1}{2^{n+1}} \triangleq \delta$$

$$Pr_y[\exists x \in \{0,1\}^n \text{s.t.} M(x,y) \neq L] \leq \Sigma_{x \in \{0,1\}^n} Pr[M(x,y) \neq L(x)]$$
$$\leq 2^n * \delta \leq \frac{1}{2}$$
$$\Rightarrow \exists y \text{ s.t. } \forall x \quad M(x,y) = L(x)$$

where the second to last inequality results from the Union Bound, which is:

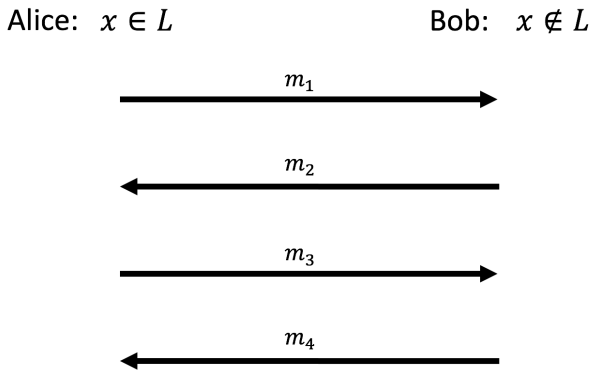
$$Pr[E_1 \cup E_2] \leq Pr[E_1] + Pr[E_2]$$

The second application of the Amplificaiton Theorem we will explore is showing that BPP $\subseteq$ PH. Recall that the PH complexity class involves a debate between parties "Alice" and "Bob", where Alice says $x \in L$ and Bob says $x \notin L$.

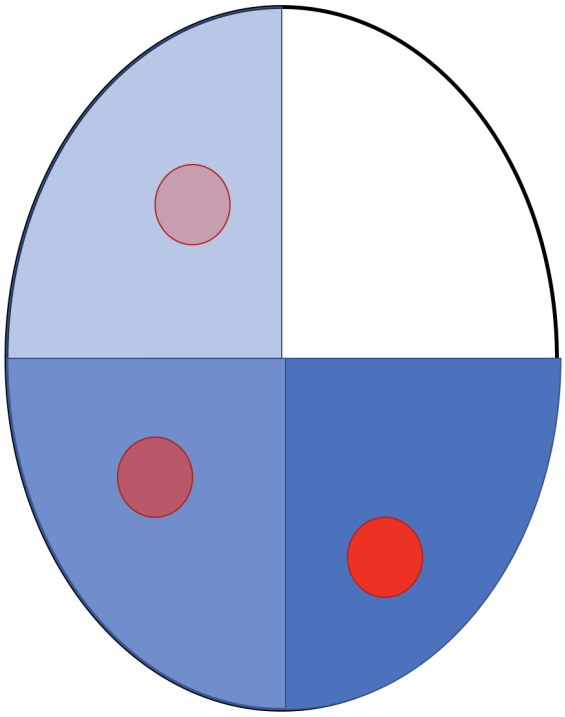
Given Alice sends message m_1, Bob responds m_2, Alice responds m_3, and Bob responds m_4, we have the verifier V such that:

$$V(x, m_1, m_2, m_3, m_4) = 1 \Leftrightarrow Pr[M(x,y) = 1] \geq \frac{2}{3}$$

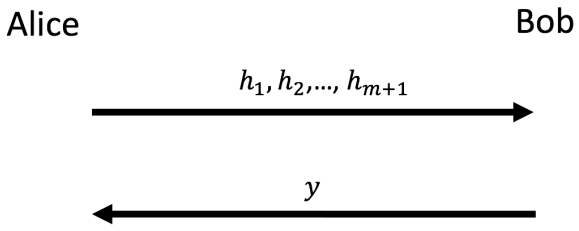
We can show this as the following picture:



Now we will prove the theorem that BPP $\subseteq$ PH [Sipser, Lautemann]. The idea of the proof is that if $S \subseteq \{0,1\}^m$ where $[m = |y|]$, then in the case when S is large, random shifts of S will cover $\{0,1\}^m$. However, when S is small, random shifts of S will leave most of $\{0,1\}^m$ uncovered. This is shown pictorially below, where subsequent shifts of S when S is large are denoted by expanding regions of blue (represented by expanded coverage areas of reduced opacity in the space of $\{0,1\}^m$). However, when S is small, the random shifts are represented by the red regions of varying opacity that leave most of $\{0,1\}^m$ uncovered.



The proof is as follows: Alice sends $h_1, h_2, ..., h_{m+1} : \{0,1\}^m \to \{0,1\}^m$. The claim Alice is making is that $\forall y \exists i \in [m+1]$ s.t. $M(x, h_i(y)) = 1$. Bob responds with y, claiming $\forall i \in [m+1] M(x, h_i(y)) = 0$. The verifier is $V : \exists i$ s.t. $M(x, h_i(y)) = 1$. We can show this as the following picture:



Now, we assume that h_i is a random 1-1 function from $\{0,1\}^m \to \{0,1\}^m$. We say that

$$x \in L : Pr_y[M(x,y) = 1] \geq \frac{1}{2}$$

Next, we fix y:, and have:

$$Pr_{h_i}[M(x, h_i(y)) = 0] \leq \frac{1}{2}$$

$$Pr_{h_1,...,h+m+1}[\forall i M(x, h_i(y)) = 0] \leq \frac{1}{2^{n+1}}$$

$$Pr_{h_1,...,h_{m+1}}[\exists y \forall i M(x, h_i(y)) = 0] \leq \frac{2^m}{2^m + 1} \leq \frac{1}{2}$$

Where the last line follows via the Union Bound. We do this to say:

$$\exists h_1, ..., h_{m+1} \text{ s.t. } \forall y \exists i M(x, h_i(y)) = 1$$

Now we have the case when:

$$x \notin L : Pr_y[M(x,y) = 1] \leq \frac{1}{2(m+1)}$$

We fix $h_1, ..., h_{m+1}$ and we fix i, and have:

$$Pr_y[M(x, h_i(y)) = 1] \leq \frac{1}{2(m+1)}$$

$$Pr_y[\exists i \in [m+1] \text{ s.t. } M(x, h_i(y)) = 1] \leq \frac{1}{2}$$

We conclude

$$\forall h_1, ..., h_{m+1} \exists y \text{ s.t. } \forall i \in [m+1] M(x, h_i(y)) = 0$$

However, there is a problem with this method. The h_i are too random and require $m * 2^m$ bits to describe. To resolve this, we use a hash for h_i given by $z_i \in \{0,1\}^m$, where:

$$h_i(y) = y \oplus z_i$$

where $\oplus$ is bitwise XOR, and note that this is a 1-1 function also.

So now, in this protocol, Alice sends $z_1, ..., z_m \in \{0,1\}^m$, Bob sends y, and the reviefer checks if:

$$V : \exists i \text{ s.t. } M(x, z_i \oplus y) = 1$$

Note the crucial role of amplification: we have that $\text{BPP}_{\frac{1}{2}, 2*(m+1)}$ where m is the number of random bits and not the length of the input. Further, neither Alice nor Bob needs to be "superpowerful" to play their parts, as Alice can pick $z_1, ..., z_{m+1}$ at random, and Bob can pick y at random.

4 Fortnow's Theorem

We now turn our attention to Fortnow's Theorem, which states that if P=Promise-RP, then P=Promise-BPP [and $\Leftarrow$ Promise-BPP = Promise-$\text{RP}^{\text{Promise-RP}}$]. (Note that in the following, we often suppress the word "Promise").

We now consider the proof. We fix an L in BPP. If RP=P, then $L \in$ P. We start with

$$L'_Y = (x, z_1, ..., z_{m+1}) \text{ s.t. } Pr_y[V(x, z_1, ..., z_{m+1}, y) = 0] \geq \frac{1}{2}$$

$$L'_N = (x, z_1, ..., z_{m+1}) \text{ s.t. } Pr_y[V(x, z_1, ..., z_{m+1}, y) = 0] \leq 0$$

which is to say that there is no such string y for which this verifier accepts. Therefore we have:

$$L' = (L'_Y, L'_N) \in \text{Promise-RP} \Rightarrow L' \in P$$

Next, we have:

$$L''_Y = x \text{ s.t. } Pr_{z_1,...,z_{m+1}}[(x, z_1, ..., z_{m+1} \in L'_N] \geq \frac{1}{2}$$

and if $x \notin L$, we have:

$$L''_N = x \text{ s.t. } Pr_{z_1,...,z_{m+1}}[(x, z_1, ..., z_{m+1}) \notin L'_Y] \leq 0.$$

We claim the last line in the statement, and will soon verify it. With this, we have:

$$L'' \in \text{Promise-RP} = L'' \in \text{P}, \text{ but } L'' = L \Rightarrow L \in \text{P}$$

Now we will verify L''_N from earlier. What we want to show is that if $x \in L_N$, then $\Rightarrow x \in L''_N$, which is defined as above. To do this, we have

$$x \in L''_N \Leftrightarrow \forall z_1, ..., z_{m+1}(x, z_1, ..., z_{m+1} \in L'_Y)$$

$$\text{from the definition of } L'_N \Rightarrow \forall z_1, ..., z_{m+1}, Pr_y[V(x, z_1, ..., z_{m+1}, y) = 0] \geq \frac{1}{2}$$

5 Quantifiers

We have that:

$$\Sigma_i^P = \mathrm{NP}^{\Sigma_{i+1}^P}$$

and we believe that

$$\forall i \Sigma_i \neq \Sigma_{i+1}$$

On the one hand, this is what the world looks like with normal quantifiers, and here we have does there exists a y where something happens or does it happen for all y. When we have policy quantifiers, which are of the form:

$$Pr_y[E] \geq \frac{2}{3}$$

$$Pr_y[E] \leq \frac{1}{3}$$

then these quantifiers are starting to collapse. In particular, we get that Promise-$\mathrm{RP}^{\mathrm{RP}}$=BPP. But then we can ask what happens if we put in another RP, and we have $\mathrm{RP}^{\mathrm{RP}^{\mathrm{RP}}}$=$\mathrm{BPP}^{\mathrm{RP}} \subseteq \mathrm{BPP}^{\mathrm{BPP}}$=BPP. This does not give additional power, as a randomized algorithm can stop and ask another randomized algorithm as an oracle, and nothing will get affected. These algorithms are all very unlikely to produce the wrong answer, so if make error probability small, when you invoke the oracle you will get the right answer. So the RP hierarchy does collapse at the second level. There are schools of thought that P=BPP, and others that P $\neq$ RP $\neq$ BPP = $\mathrm{RP}^{\mathrm{RP}^{\mathrm{RP}}}$. So when working with randomness, usual belief of what qualifiers can and cannot do does not seem to work out, and we will see more dramatic effects of it. In the future, we will look at what happens if we give a randomized algorithm to an NP oracle.

6 Next Class

In future lectures, we will look at computation where we count the number of answers. Specifically, if we start with SAT, we can look at:

1. Promise Problem: $\exists$ a unique SAT assignment, find it.
2. Parity-SAT: is the number of satisfying assignments odd or even?
3. Given satisfying formula, compute the number of satisfying assignments.