18.405J/6.841J Advanced Complexity Theory March 1, 2001

Lecture 7

Lecturer: Dan Spielman *Scribe: Abhinav Kumar*

Today we will talk about Randomized Complexity classes.

Last time we showed BPP $\subset$ P/poly. Today we will show BPP $\subset \Sigma_2 P \cap \Pi_2 P$

BPP error amplification

Error amplification means decreasing the probability of error. The key tool in this regard is the use of *Chernoff bounds*.

Theorem 1 *Let $X_1, X_2, ..., X_n$ be independent, identically distributed random variables taking values in [0,1]. Let $Y = \sum_{i=0}^{n} X_i$, μ= E[Y].*

Then $Pr[|Y - \mu| > \epsilon\mu] < e^{-\mu\epsilon^2/3}$, for $\epsilon \in [0, 1]$.

Proof: See handout on probability.

Amplification Lemma: Let L be a language such that there is a randomized polynomial-time TM M such that,

$$x \in L \Rightarrow Pr[M(x) accepts] \geq C(|x|)$$
$$x \notin L \Rightarrow Pr[M(x) accepts \leq S(|x|)$$
$$\text{where } C(n) - S(n) \geq \frac{1}{p(n)} \text{ for some poly } p()$$

Then $\forall b > 0, \exists$ a random ptime TM M' such that

$$x \in L \Rightarrow Pr[M'(x) accepts] \geq 1 - 2^{-|x|^b}$$
$$x \notin L \Rightarrow Pr[M'(x) accepts \leq 2^{-|x|^b}$$

(Remark : This is reasonably tight)

Proof: M' runs M $k = 12(p(n)^3 + n^b$ times, $n = |x|$. Accepts if M accepted more than $k \cdot \frac{s(n)+c(n)}{2}$ times. Applying Chernoff's bound, we get the desired behaviour of M'.

Theorem 2 *(Sipser) BPP $\subset \Sigma_2 P$.*

Note: This implies co-BPP $\subset \Pi_2 P$, but co-BPP = BPP since definition of BPP is symmetric.

Proof:

Let L be a language in BPP. We know $\exists A \in$P and a function $f(n) = n^{O(1)}$ such that

If $w \in L$ then $\Pr_{r:|r|=f(n)}[(r, w) \in A] > 1 - 2^{-n}$, and

if $w \notin L$ then $\Pr_{r:|r|=f(n)}[(r, w) \in A] < 2^{-n}$.

where $n = |w|$.

Def. Define $R_w = \{r : |r| = f(n)$ such that $(r, w) \in A\}$. Correspondingly,

If $w \in L$ then $|R_w| > (1 - 2^{-n})2^{f(n)}$ and

if $w \notin L$ then $|R_w| < 2^{-n}2^{f(n)}$.

The idea here is to take a number of translations of R_w, and see if they cover the entire space $\{0,1\}^{f(n)}$. Each translation of R_w has the same size as R_w, and if R_w is most of the space (ie, $w \in L$) then this collection of translations would be likely to cover the space. However, if R_w is very small (ie, $w \notin L$) then this collection could never cover the space. More formally,
Def. (translation) Let $S \subset \{0,1\}^{f(n)}$. For $t \in \{0,1\}^{f(n)}$ let the translation $S \oplus t$ be defined as

$$\{x : x \oplus t \in S\}$$

where $x \oplus t$ is defined as the XOR of the two strings (or, the bitwise sum modulo 2).
Claim. (1) If $|S| > (1 - 2^{-n})2^{f(n)}$ then $\exists \tau = \{t_1, \dots t_{f(n)}\}$ such that

$$\bigcup_{i=1}^{f(n)} (S \oplus t_i) = \{0,1\}^{f(n)}$$

(2) If $|S| < 2^{-n}2^{f(n)}$ then $\forall \tau = \{t_1, \dots t_{f(n)}\}$,

$$\bigcup_{i=1}^{f(n)} (S \oplus t_i) \neq \{0,1\}^{f(n)}$$

First, we show that the claim proves the theorem. If the claim is true, we can design a $\Sigma_2 P$ machine M to solve L as follows:
1. Use $\exists$ states to generate τ.
2. Use $\forall$ states to generate $r \in \{0,1\}^{f(n)}$.
3. Check if $r \in \bigcup_i R_x \oplus t_i$ and accept if so, otherwise, reject.

This is polynomial time, since we can check whether $r \in R_x$ in polynomial time, and $f(n)$ is polynomial in n. By the claim, if $x \in L$ then on any correct τ we accept. If $x \notin L$ we reject, since there is no such τ. Therefore, we only have to prove the claim.

First, we prove part (2) of the claim. If $|S| < 2^{-n}2^{f(n)}$ then

$$\left|\bigcup_i (S \oplus t_i)\right| \leq f(n)2^{-n}2^{f(n)}.$$

Since $f(n) = n^{O(1)}, f(n)2^n < 1$ for sufficiently large n. Therefore, this union doesn't cover $\{0,1\}^{f(n)}$.

Note that the "sufficiently large n" clause here doesn't cause a problem. If we take this into account, we need only hard-code the correct answer for all words smaller than this bound into our $\Sigma_2 P$ machine.

Next, we prove part (1). Let

$$\begin{aligned} p &= \Pr_\tau[\forall_{|r|=f(n)} r \in \bigcup_{i=1}^{f(n)} (t_i \oplus S)] \\ &= 1 - \Pr_\tau[\exists_{|r|=f(n)} r \notin \bigcup_{i=1}^{f(n)} (t_i \oplus S)] \\ &\geq 1 - \sum_{|r|=f(n)} \Pr_\tau[r \notin \bigcup_{i=1}^{f(n)} (t_i \oplus S)]. \end{aligned}$$

We choose the t_i's independently, so we can consider them independently. Therefore,

$$\begin{aligned} p &\geq 1 - \sum_{|r|=f(n)} \prod_{i=1}^{f(n)} Pr_{t_i}[r \notin t_i \oplus S)] \\ &= 1 - \sum_{|r|=f(n)} \prod_{i=1}^{f(n)} 2^{-n} \end{aligned}$$

since $r \in t_i \oplus S$ if and only if $t_i \in r \oplus S$,

$$= 1 - 2^{f(n)}(2^{-n})^{f(n)} = 1 - 2^{-f(n)(n-1)} > 0.$$

Since this probability is nonzero, there must be at least some τ for which the union of the translations determined by τ covers $\{0,1\}^{f(n)}$. This completes the proof. ■

Verifying Polynomial identities

Let p be a given polynomial in k variables, $q_1, ..., q_k$ given polnomials in m variable. The equation

$$p(q_1(y_1, ..., y_m), ..., q_k(y_1, ..., y_m)) = 0$$

can be difficult to check deterministically. Expanding the polynomial out would give us an exponential number of terms. However, if we use randomization there is an easy test. Choose $x_1, ..., x_m$ at random and see if we get zero. If we choose $x_i's$ in a large enough range, the probability that the test is passed but p is not identically zero becomes exponentially small. So we can check polynomial identities in co-RP.

Lemma 3 (Schwartz's Lemma) *Let $P(x_1, x_2, \ldots, x_n)$ be a polynomial of degree d. Then if $P \not\equiv 0$, then*

$$\Pr_{x_1, x_2, \ldots x_n \in S}[P(x_1, x_2, \ldots, x_n) = 0] \leq \frac{dn}{|S|}.$$

Proof: We use induction on n.
Base case $(n = 1)$: If $P \not\equiv 0$, then there are at most d zeroes in p. At most $d = d \cdot 1$ of them are in S.
Inductive step: Write

$$P(x_1, x_2, \ldots, x_n) = \sum_{i=0}^{d_1} x_1^i P_i(x_2, \ldots, x_n).$$

By hypothesis,

$$\Pr_{x_2, \ldots, x_n}[P_{d_1}(x_2, \ldots, x_n) = 0] \leq \frac{(n-1)d}{|S|}.$$

If $P_{d_1}(x_2, \ldots, x_n) \neq 0$, then $\Pr_{x_1}[P(x_1, \ldots, x_n)] \leq \frac{d_1}{|S|}$. ■