6.841 Advanced Complexity Theory Mar 14, 2007

Lecture 11

Lecturer: Madhu Sudan *Scribe: Tim Abbott*

1 Overview

This lecture describes BPP amplification, shows BPP $\subset$ P$/_{\text{poly}}$, and shows that BPP is contained within the polynomial hierarchy. Finally, we have an introductory discussion of randomized rounding for unique SAT.

2 BPP Amplification

Definition 1 *A language L is in* Strong BPP *if for any polynomial $q(n)$, there exists a polynomial time machine $M(\cdot,\cdot)$ such that for any $x \in \{0,1\}^n$, we have that either $x \in L$ and*

$$Pr_y\left[M(x,y) accepts\right] \geq 1 - 2^{-q(n)}$$

or $x \notin L$ and

$$Pr_y\left[M(x,y) accepts\right] \leq 2^{-q(n)}$$

Remark In this discussion, we will consider a machine polynomial time if its time is polynomial in the length of the first argument. The part of the second argument (the random bits) that the machine can ever examine will consequently be polynomial in the length of the first argument, so we can assume that it is polynomial in the size of the first argument.

Definition 2 *A language L is in* Weak BPP *if there exists a polynomial time machine $M(\cdot,\cdot)$, a polynomial $p(n)$, and a polynomial time computeable function $s(n)$, such that for any $x \in \{0,1\}^n$, we have that either $x \in L$ and*

$$Pr_y\left[M(x,y) accepts\right] \geq s(n) + \frac{1}{p(n)}$$

or $x \notin L$ and

$$Pr_y\left[M(x,y) accepts\right] \leq s(n)$$

Theorem 3 (Amplification Theorem) *Strong BPP = Weak BPP*

Proof Clearly, Strong BPP is contained in Weak BPP, so we must show that Weak BPP is contained in Strong BPP. Suppose L is in Weak BPP, and fix a polynomial $q(n)$. We have a polynomial $p(n)$, machine M, and polynomial time

computeable function $s(n)$ satisfying the weak BPP property. We will define an algorithm M' as follows. Pick $y_1, \ldots, y_t$ independently and identically from the distribution of possible random inputs for M, and for each y_i, compute $z_i = M(x, y_i)$. Let $\overline{z} = \frac{\sum_{i=1}^{n} z_i}{t}$. Then M' accepts if and only if $\overline{z} \geq s(n) + \frac{1}{2p(n)}$. We will seperate out the two cases $x \in L$ and $x \notin L$.

We will need to use the Chernoff Bound:

Theorem 4 (Chernoff Bound = Tail Inequality = Hoeffding Bound) *Suppose $z_1, \ldots, z_t$ are bounded (say in $[0, 1]$), and are independently and identically distributed, with mean μ. Then*

$$Pr\left[|\overline{z} - \mu| \geq \epsilon\right] \leq e^{-\Omega(\epsilon^2 t)}$$

Remark There are cases in which one can strengthen this bound, but we won't be needing the stronger versions.

Suppose now that $x \in L$. The y_i's are independent and identically distributed with $\mu \geq s(n) + \frac{1}{p(n)}$. By the Chernoff Bound, with $\epsilon = \frac{1}{2p(n)}$ and $\mu \geq s(n) + \frac{1}{p(n)}$, we have that

$$Pr\left[\text{M' accepts } x \in L\right] = Pr\left[\overline{z} \geq \frac{1}{2p(n)} + s(n)\right] \leq Pr\left[|\overline{z} - \mu| \geq \frac{1}{2p(n)}\right] \leq e^{-\Omega(t/p(n)^2)}$$

For $t = \Omega(p(n)^2 q(n))$, we see that for $x \in L$, the Strong BPP condition holds.

The argument is similar for $x \notin L$, so the Strong BPP conditions holds for $x \notin L$ as well. Thus Weak BPP equals Strong BPP, as desired.

■

Remark Note the quadratic dependence on $p(n)$. This procedure of converting a Weak BPP algorithm to a Strong BPP algorithm costs a lot of randomness, and time, though of course they both remain polynomial in the input size.

3 BPP $\subset$ P /poly

Corollary 5 (Adleman) BPP $\subset$ P /poly

Proof It suffices to show that Strong BPP is contained in P /poly. Let $L \in$ BPP. Pick as our polynomial $q(n) = 2n$. Then M accepts L with error probability 2^{-2n}. Suppose $M(x, y)$ is a machine with input x and random bits y. We say that y is *wrong advice* for x if $M(x, y) \neq L(x)$. Then

$$Pr_y\left[y \text{ is wrong for } x\right] \leq 2^{-2n}$$

Using the fact that there are at most 2^n values of x of length n, we have by the Union Bound that

$$Pr_y\left[\text{There exists an } x \text{ such that } y \text{ is wrong for } x\right] \leq 2^n * 2^{-2n} = 2^{-n}$$

Since this probability is less than 1, there exists some y such that y is wrong for no x, i.e. there exists a y such that $M(x, y) = L(x)$ for all x of length n. Using this y as the advice, we have that $L \in \mathrm{P}/_{\mathrm{poly}}$, as desired.

■

Remark Since BPP $\subset \mathrm{P}/_{\mathrm{poly}}$, if we had that NP $\subset$ BPP, we would then have that NP $\subset \mathrm{P}/_{\mathrm{poly}}$. But NP $\subset \mathrm{P}/_{\mathrm{poly}}$ implies that the polynomial hierarchy collapses, which we think is quite unlikely. Thus, this corollary suggests that $np \not\subset$ BPP, or that nondeterministic polynomial time is more powerful than randomized polynomial time.

4 BPP $\subset \Sigma_2^P$

Theorem 6 (Sipser, Lautemann) BPP *is contained in the polynomial hierarchy. In particular,* BPP $\subset \sum_2^P$

Proof In Σ_2^P, we have a short interaction between the prosecution, who is trying to show that $x \in L$, and the defense, who is trying to show that $x \notin L$. First, the prosecution sends a message to the defense, and then the defense sends a reply. The jury, who sees both messages, must then decide in polynomial time whether or not x is in L.

Suppose $L \in$ BPP. Then we have M, which uses l random bits, and has error probability $2^{-q(n)}$ for some polynomial $q(n)$. Suppose x is an input for L. Let us iterate through a few ideas for how to do this proof.

Idea 1. The prosecution sends $y \in \{0,1\}^l$, and the jury accepts if $M(x, y)$ does. But since BPP has false positives, the prosecution could do this for $x \notin L$, and this idea doesn't work.

Idea 2. The defense sends $y \in \{0,1\}^l$, and the jury rejects if $M(x, y)$ does. But since BPP has false negatives, the prosecution could do this for $x \in L$, and this idea doesn't work.

Idea 3. The defense sends most of the bits of y, and the prosecution picks the rest. Since if $x \in L$, most values of y cause M to accept, the prosecution should be able to succeed, while if $x \notin L$, most values of y cause M to reject, so the defense should be able to pick his bits of y such that no choice on the prosecution's part causes M to accept. This is on the right track, but requires the defense to pick first, and isn't yet a protocol.

Idea 4. The prosecution picks a list $y_1, \ldots, y_k$, and then the defense picks some y. The jury then considers $y \oplus y_i$, for each y_i, and if $M(y \oplus y_i)$ accepts for any i, then the jury accepts, otherwise it rejects.

We will use the algorithm of Idea 4. First, suppose $x \in L$. We say that y_i is *wrong* for y if $M(x, y \oplus y_i)$ rejects. Then using the Strong BPP property, for any y,

$$Pr_{y_i}[y_i \text{ is wrong for } y] \leq 2^{-q(n)} \leq \frac{1}{2}.$$

If the y_i's are picked independently at random, then

$$Pr_{y_1,\ldots,y_k}[y_i \text{ is wrong for } y \text{ for all } i] \leq 2^{-k}$$

Thus, by the union bound, we have that

$$Pr_{y_1,\ldots,y_k}[\text{There exists } y \text{ such that for all } i,\ y_i \text{ is wrong for } y] \leq 2^{l-k}$$

Thus if we pick $k > l$, then for any y, there must exist some set of y_i's that are not all wrong for y. The prosecution can then pick these y_i's, and regardless of what y the defense picks, the jury will decide that $x \in L$, as desired.

Suppose, then, that $x \notin L$. Here we say that y is wrong for y_i if $M(x, y \oplus y_i$ accepts. Fix $q(n) = n$, so that k and l are both polynomial in n. Then we have that

$$Pr_y[y \text{ is wrong for } y_i] \leq 2^{-q(n)} \leq 2^{-n}$$

Then by the union bound,

$$Pr_y[\text{There exists } i \text{ such that } y \text{ is wrong for } y_i] \leq k2^{-n} = poly(n)2^{-n} < 1$$

Thus, there must exist some choice of y for which no i exists where y is wrong for y_i. Thus the defense can pick this y, and the jury will find that $M(x, y \oplus y_i)$ rejects for each i, and thus will reject x, as desired. We have thus shown that BPP $\subset \Sigma_2^P$. ■

5 Randomized Reductions

Randomization can help prove results in complexity theory that on their face, have nothing to do with randomness. Such reductions are called randomized reductions. Our first example will be in constructing one-way permutations. We know that if P = NP, modern cryptography doesn't work. But what if P ≠ NP? We don't know how to show that this implies modern cryptography is strong.

Definition 7 *$f : \{0,1\}^n \to \{0,1\}^n$ is a* one-way permutation *if f is a bijection and $f(x)$ is easy to compute from x, yet x is hard to compute from $f(x)$.*

For cryptography, we'd like to know that for a random x, the one-way permutation is hard to invert. Here, we will only show that there exist x for which the one-way permutation is hrad to invert.

Consider the following idea for constructing a one-way permutation, given that P ≠ NP. Given (ϕ, x) such that $\phi(x) = 1$, where ϕ is a CNF formula, we

send it to ϕ. It is easy to compute ϕ from (ϕ, x). But the opposite, computing (ϕ, x), given only ϕ, requires solving satisfiability. Unfortunately, this mapping is not one-to-one, so it is not a one-way permutation (it is instead a one-way function). So, if we had a ϕ that was an element of Unique-SAT, the problem of finding a satisfying assignment when we know that there is a unique solution, and Unique-SAT was NP-hard, then we could use such a ϕ, and we'd be done. So, we must study the Unique-SAT problem: Given a formula ϕ that has a unique satisfying assignment, does there exist an efficient algorithm to find it?

Theorem 8 (Valiant, Vazirani) *Unique-SAT is hard.*

In this lecture, we will just sketch the proof, which will be given formally next time. We will use a randomized reduction to convert an instance of SAT into an instance of a promise problem related to Unique-SAT. It will send:

- SAT $\rightarrow$ Unique-SAT
- $\phi \rightarrow \psi$, probabilistically.
- $\phi \in SAT \rightarrow \psi$ that has a unique satisfying assignment, with probability $\frac{1}{p(n)}$.
- $\phi \notin SAT \rightarrow \psi$ with not satisfying assignment, with probability 1.

Since ϕ is satisfiable, it has for some $1 \leq m \leq n$, between 2^{m-1} and 2^m satisfying assignments. We will guess m (costing at most polynomial probability of success, since there are only n distinct possibly values of m), and do the following. Let $h : \{0,1\}^n \rightarrow \{0,1\}^m$. We will define $\psi(x) = 1$ if and only if both $\phi(x) = 1$ and $h(x) = 0^m$. We hope that with probability bounded below by $\frac{1}{p(n)}$, this will give us a formula ψ that has a unique satisfying assignment, but is still hard. Thus, we need h to be both efficiently computeable, and "random". These cannot be simulataneously true, so we'll have to do some sort of compromise. In the next lecture, we will formalize this notion.