6.841: Advanced Complexity Theory Fall 2012

Lecture 15— November 1, 2012

Prof. Dana Moshkovitz *Scribe: Liz Simon*

1 Overview

In the last lecture we saw a proof presented by Madhu Sudan that $E \not\subseteq SIZE(2^{\epsilon n}) \implies BPP = P$, where $E = TIME(2^{O(n)})$. In other words, he showed that if E does not have circuit lower bounds of exponential size, we can prove $BPP = P$.

This lecture will be the last lecture on the topic of randomness. We have seem that randomness seems to give us some power in algorithms, but we can also avoid it at some price – for example, derandomizing algorithms via a pseudorandom generator. However, pseudorandom generators rely on unproven circuit lower bounds.

In this lecture we will explore the question of whether the converse of the theorem above is also valid: does derandomization imply circuit lower bounds?

2 Main Section

2.1 Intuition behind Madhu's lecture

2.1.1 Why do we use circuit lower bounds?

Q: Why are we using circuit lower bounds here, as opposed to a claim such as $E \not\subseteq P$ for example?
A: The proof of the Nisan-Wigderson pseudorandom generator relies on nonuniformity, by showing that distinguishing a pseudorandom generator implies a circuit for solving a hard problem – this reduction involves hardwiring advice into a circuit in order to solve the hard problem. A contradiction requires that the hard problem has circuit lower bounds.

In particular, the derandomization works by feeding the randomized algorithm the output of a pseudorandom generator. The randomized algorithm has input, which can serve as an advice for distinguishing between the output of the generator and truly uniform bits. So even though we want to derandomize a uniform algorithm, we end up needing bits that look random to a circuit.

2.1.2 Exponential versus polynomial time

Q: Why is our assumption about exponential time when we want a result about polynomial time?
A: This is because the generator takes a logarithmic size input and turns it into a polynomial size random string, so the generator is exponential in its input size.

2.2 The Kabanets-Impagliazzo Theorem

The Nisan-Wigderson pseudorandom generator shows that a circuit lower bound against E implies that BPP can be derandomized. The Kabanets-Impagliazzo Theorem shows that derandomizing a *single* problem in BPP implies circuit lower bounds.

2.2.1 Statement of the theorem

Suppose that the Polynomial Identity Testing problem is in P. Then at least one of the following is true:

1. $NEXP \not\subseteq P_{/poly}$; or,
2. $PERM$ doesn't have polynomial size arithmetic circuits

2.2.2 Remarks about the theorem

First, notice that this is of the same flavor as what we aimed to show, because we have a derandomization result implying a hardness result.

However, we should be careful about the fact that this theorem implies an "or" result, as these types of statements may be trivial. For example, the statement "$P \neq NP$ or $NP \neq EXP$" is trivial, even though either of the two statements on their own would be an amazing result. Fortunately, the result we are interested in turns out to not be trivial at all.

2.2.3 Polynomial Identity Testing

The Polynomial Identity Testing problem, or PIT, is the following: given as input an arithmetic circuit C over a field $\mathbb{F}$, where the size of the field is much larger than the degree of C, decide whether or not C always computes 0.

2.2.4 Lemma: $PIT \in coRP$

This is proven by the algorithm given in Madhu's lecture. The algorithm works as follows:

1. Pick $\vec{x} \in \mathbb{F}^n$
2. Accept if $C(\vec{x}) = 0$

If $C(\vec{x}) = 0$, the algorithm never makes a mistake, which is why it shows that $PIT \in coRP$. However, if $C(\vec{x})$ is not the identically zero polynomial, then the following Lemma shows that this algorithm succeeds with high probability.

2.3 Schwartz-Zippel Lemma

This lemma was proved by Reed and Muller, and a stronger version of it was proved by Schwartz [1] and Zippel [2]. The proof presented here is from [3]:

Lemma: $C \neq 0 \implies \Pr(C(\vec{x}) = 0) \leq \frac{\deg(C)}{|\mathbb{F}|}$

Proof: Without loss of generality, assume $\deg C > 0$, because otherwise the proof is trivial. We can break C up into two parts, one homogeneous of degree $\deg C$ and the other with degree less than $\deg C$: $C = C^{\deg C} + C^{<\deg C}$.

Notice that $C^{\deg C} \neq 0$. Therefore $\exists y \in \mathbb{F}^n$ such that $C^{\deg C}(y) \neq 0$. Note: this uses the assumption that $\deg C < |\mathbb{F}|$. We claim that $y \neq \vec{0}$, because if it were then $C^{\deg C}(y) = 0$, but we chose y so that this would not be the case. Now, we can partition $\mathbb{F}^n$ into lines $\{z + t\vec{y} | t \in \mathbb{F}\}$ for all $z \in \mathbb{F}$.

Fact: $C^{\deg C}(z + ty) = t^{\deg C} \cdot C^{\deg C}(y) + \cdots$.

This can be verified easily, since this is a polynomial in t of degree $\deg C$, and the highest part has coefficient $C^{\deg C}$. But we know $C^{\deg C}(y) \neq 0$, by how we chose it. So on each line we get a nonzero polynomial in one variable. We know that a polynomial in one variable of degree n has at most n zeroes. So the number of zeroes on each line is at most $\deg C$, and so the total number of zeroes is at most $\deg C$ times the number of lines. i.e. $|\{\vec{x} | C(\vec{x}) = 0\}| \leq \deg C \cdot |\mathbb{F}|^{n-1}$.

2.4 The Permanent

The permanent of an $n \times n$ matrix is defined by the following formula:

$$Perm(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i,\sigma(i)},$$

where S_n is the group of all permutations of n elements.

2.4.1 Application of the permanent

If A is a matrix of zeroes and ones, where for some bipartite graph $a_{ij} = 1$ if and only if there is an edge between i and j in the graph, then the permanent of A counts how many perfect matchings the graph has.

2.4.2 Valiant's Theorem

This theorem proved by Valiant [4] states that the problem of finding the permanent of a matrix, known as $PERM$ is $\#P - complete$.

Definition: $\#P$ counts how many witnesses an NP relation has. For example, $\#SAT$ counts how many satisfying assignments the given formula has. $\#P$ is the class of all problems that can be formulated in this way.

Note: $\#P$ is not a class of languages, but a class of functions.

2.5 Proof of Kabanets-Impagliazzo

We will assume three things towards contradiction

1. $PIT \in NSUBEXP$
2. $NEXP \subseteq P_{/poly}$
3. $PERM$ has polysize arithmetic circuits

We aim to get a contradiction, in order to prove that (1) implies either (2) or (3) must be false.

2.5.1 Proof by Aaronson and van Melkebeek

The proof we will present is not the original proof, but one written by Aaronson and van Melkebeek[5]. The contradiction we aim to show is $\Sigma_2 \subseteq SIZE(n^c)$ for some constant c. This is a contradiction because we showed that it is false in assignment 1.

2.5.2 Main Lemma

The main lemma states that statements (1) and (3) above imply that $\Sigma_2 \subseteq NSUBEXP$, where $NSUBEXP = \cap_{\epsilon>0} NTIME(2^{n^\epsilon})$. This will be proved later in the lecture.

2.5.3 Proof, assuming the main lemma

We know that $\Sigma_2 \subseteq NSUBEXP$ by the main lemma. So we know that $\Sigma_2 \subseteq NSUBEXP \subseteq NE$, from (2). NE is the set of all problems reducible to

$$L = \{< M, x, t > \mid \text{nondeterministic TM } M \text{ accepts } x \text{ within } t \text{ steps}\}.$$

Note that the natural algorithm for deciding L runs in time t (where the input size is $\log t$). Clearly $NE \subseteq NEXP$, and since $NEXP \subseteq P/poly$, there must be a constant c, such that L can by solved by a circuit of size n^c. This is the contradiction we desired to show, so assuming the main lemma, we're done.

2.6 Proof of Main Lemma

Assuming (1) and (3), we want to show $coNP \subseteq NSUBEXP$, because then that would imply that $\Sigma_2 \subseteq NSUBEXP$ – since obviously $NP \subseteq NSUBEXP$. We know that $L \in coNP \iff$

($x \in L \iff$ there are no witnesses for x). Counting the number of witnesses is a $\#P$ problem, so perhaps if we could reduce $coNP$ to computing PERM, and show that PERM is computable in $NSUBEXP$, we would be done. By Valiant's Theorem, we can construct a matrix $A_{L,x}$ such that the number of witnesses for x is zero $\iff \text{Perm}A_{L,x} = 0$. We now show how to compute the Permanent in $NSUBEXP$ time, using our assumptions.

2.6.1 Downward Self-reducibility of the Permanent

The permanent has a property known as downward self-reducibility, which means that it satisfies the following formula:

$$Perm(A) = \sum_{i=1}^{n} a_{1,i} Perm(A_{1,i})$$

where $A_{1,i}$ denotes the minor of A where the first row and ith column of A is removed. This gives us a recursive formula for the permanent in terms of its minors.

By (3), we know that the permanent has polynomial size circuits, but we don't know what they are. To deal with this, we should:

1. Guess the polynomial size circuits for computing permanent $\{C_n\}$
2. Check $\{C_n\}$ by checking if $C_n(A) \equiv \sum_i a_{1,i} C_{n-1}(A_{1,i})$, or equivalently, if the polynomial $P_n(A) = C_n(A) - \sum_i a_{1,i} C_{n-1}(A_{1,i})$ is identically zero. This is a polynomial in n^2 variables, and there n such polynomials.

Since both sides of the equations in item 2 above are arithmetic circuits, this is a Polynomial Identity Testing problem, which has a $NSUBEXP$-time algorithm by (1). We can use this method to guess and verify the circuits that compute the Permanent, and then use it to solve any problem in $\#P$, and in particular, solve any $coNP$ problem, and so we're done!

2.7 Conclusion

In this lecture, we proved the Kabanets-Impagliazzo Theorem, which tells us that derandomization gives us some hardness result. This is the last lecture on the topic of randomness. Next time we will begin exploring the topic of Probabilistically Checkable Proofs.

References

[1] Schwartz, Jack. *Fast probabilistic algorithms for verification of polynomial identities*, Journal of the ACM, 27:701717. 1980.

[2] Zippel, Richard. *An Explicit Separation of Relativised Random Polynomial Time and Relativised Deterministic Polynomial Time*, 1989.

[3] Moshkovitz, Dana. *An Alternative Proof of the Schwartz-Zippel Lemma*, Electronic Colloquium on Computational Complexity, 10-096. 2010.

[4] Valiant, Leslie G. *The Complexity of Computing the Permanent*, Theoretical Computer Science (Elsevier) 8 (2): 189201. 1979.

[5] Aaronson, Scott and D. van Melkebeek. *On Circuit Lower Bounds from Derandomization*, Theory of Computing 7(1):177-184, 2011.