18.405J/6.841J Advanced Complexity Theory		February 20, 1997
	Lecture 5	
Lecturer: Dan Spielman		*Scribe: Russell Schwartz*

1 Introduction

In this lecture we will prove a relationship between the polynomial time hierarchy and another class we will study: P/poly. First, we will define the class P/poly. We will then prove several propositions useful to the proof. Finally, we will be able to prove the theorem.

The central theorem we will be proving states that if NP can be decided by a sequence of polynomial size circuits then the polynomial hierarchy collapses.

2 $\mathcal{C}/\mathcal{S}$

We begin by defining the notation $\mathcal{C}/\mathcal{S}$.

Definition 1 *Let $\mathcal{C}$ be a class of languages and $\mathcal{S}$ be a family of sequences of strings. Then define $\mathcal{C}/\mathcal{S} = \{\mathcal{L} | \exists \mathcal{A} \in \mathcal{C}$ and $s \in \mathcal{S}$ s.t. $x \in L \Leftrightarrow (x, s_{|x|}) \in A\}$.*

In this lecture, we will be particularly concerned with one example of the notation C/S: the class $P/poly$.

Example 2 *$P/poly = \{L | \exists A \in P$ and a sequence s and a polynomial p s.t. $|s_i| \leq p(i)$ and $x \in L \Leftrightarrow (x, s_{|x|}) \in A\}$, where by s we mean the sequence of strings $s_1, s_2, \ldots$.*

Proposition 3 *$P/poly$ = the languages accepted by some sequence of polynomial size circuits (i.e. we have a sequence of circuits $c_1, c_2, \ldots$ and a polynomial p s.t. $|c_i| \leq p(i)$ and $x \in L \Leftrightarrow c_{|x|}$ accepts x.)*

Proof Let C = the set of languages accepted by a sequence of circuits of polynomial size.
$C \subseteq P/poly$: Assume L is a language accepted by a sequence of polynomial size circuits. Then for each input $x \in L$, we can encode the circuit accepting strings of size $|x|$ in a string of size polynomial in $|x|$. Then we can define the set A to be $\{(x, s_{|x|}) | s_{|x|}$ describes a circuit accepting $x\}$. Then there is some sequence s s.t./ $A \in P$ and $x \in L \Leftrightarrow (x, s_{|x|}) \in A$. Therefore $L \in P/poly$.
$C \supseteq P/poly$: Assume $L \in P/poly$. Since the advice string s_i is constant for each input size i of the elements of L and a separate circuit is used for each input size, we can hardcode the appropriate advice string for each i into the circuit for input size i. Each circuit i can then implement the polynomial time algorithm needed to determine membership in L using its hardcoded advice string and a polynomial number of gates. Thus, we can construct a sequence of polynomial size circuits deciding L. Therefore, $L \in C$. ■

3 NP and $P/poly$

We will now state the main theorem of this lecture.

Theorem 4 (Karp-Lipton + Sipser) *If $NP \subseteq P/poly$ then $PH \subseteq \Sigma_2^P$.*

To begin a proof of this theorem, we can first note that $NP \subseteq P/poly$ iff $SAT \in P/poly$. Therefore $NP \subseteq P/poly$ is equivalent to: $\exists A \in P$, a polynomial p, and a sequence g s.t. $|g_i| \leq p(i)$ and $\phi \in SAT \Leftrightarrow (\phi, g|\phi|) \in A$.

We can further define an advice string α to be *good* if it is of the form $\alpha = (\alpha_1, \ldots, \alpha_l)$ s.t. $\forall \phi : |\phi| \leq l, \phi \in SAT \Leftrightarrow (\phi, \alpha_{|\phi|}) \in A$ and $|\alpha| \leq p(i)$.

Example 5 *$(g_1, g_2, \ldots, g_l)$ is a good string for any l.*

For the proof of Theorem 4 we will need to show how to decide *good.* We will first show that good is decidable in Π_2^P.

Proposition 6 *good* $\in \Pi_2^P$

Proof Assume we are given some α. Then we can construct an ATM that performs the following steps:

1. Use $\forall$ to guess all formulas ϕ s.t. $|\phi| \leq l$.
2. Test whether α encodes a circuit for deciding inputs of size $|\phi|$ and reject if not.
3. If $(\phi, \alpha_{|\phi|}) \in A$ then use $\exists$ to guess a satisfying assignment and accept if one exists.
4. If $(\phi, \alpha_{|\phi|}) \notin A$ then use $\forall$ to try out all truth assignments and accept if none cause ϕ to be true.

This ATM will accept if and only if α encodes a circuit that correctly decides all formulas of length at most l. Since it uses at most one reversal starting with a $\forall$, this proves *good* $\in \Pi_2^P$. ■

Before we can prove Theorem 4, we will need to prove a slightly stronger result than the one above. We will require one additional concept, *self-reducibility*, to accomplish this.

4 Self-reducibility

We define a language L to be *self-reducible* if there exists an oracle polynomial time TM M that on inputs of length n only asks its oracle questions of length less than n and that accepts a string w iff $w \in L$.

Proposition 7 *SAT is self-reducible.*

Proof To prove this, we first note that any instance of SAT can be written out more fully as the problem of deciding the truth of the statement $\exists x_1 \exists x_2 \ldots \exists x_n : \phi(x_1, x_2, \ldots, x_n)$ for some ϕ. By separately fixing one variable into each possible value, we can express this statement in terms of two smaller expressions equivalent to: $\exists x_2 \ldots \exists x_n : \phi(true, x_2, \ldots, x_n)$ or $\exists x_2 \ldots \exists x_n : \phi(false, x_2, \ldots, x_n)$. An oracle TM M could therefore solve an instance of SAT in polynomial time by using the oracle to decide these two smaller problems, thus proving the proposition. ■

Using the self-reducibility of SAT, we can now prove a stronger result for *good* than we proved above. Specifically:

Proposition 8 *good* $\in coNP$

Proof The proof involves using self-reducibility to replace the use of $\exists$ in the proof that *good* $\in \Pi_2^P$. Specifically, given a string α, we decide whether α is good as follows:

1. Use $\forall$ to pick all formulas ϕ s.t. $|\phi| \leq l$.

2. Test if $(\phi, \alpha_{|\phi|}) \in A$.

3. If yes, then use α and the self-reducibility of SAT to find a witness for ϕ and accept if a witness is found.

4. If no, use $\forall$ to verify that there is no assignment that makes ϕ true and accept if none is found.

This will decide whether α is good without using $\exists$, proving the proposition. ■

We can note that it is possible to move all of the $\forall$ statements to the beginning of the algorithm by replacing line 4. with:

4. Use self-reducibility to reduce ϕ to two formulas, ϕ_1 and ϕ_2, of one less variable and accept if $(\phi_i, \alpha_{|\phi|}) \notin A$ for both ϕ_1 and ϕ_2.

This approach will work because we are already testing all smaller formulas through the first $\forall$. Therefore if α is incorrect for either ϕ_1 or ϕ_2 that will be caught on the tests of those formulas or their sub-expressions.

5 Proving the theorem

We can now prove Theorem 4. First, we restate the theorem as follows:

Theorem 9 *If $NP \subseteq P/poly$ then $\Sigma_3^P \subseteq \Sigma_2^P$.*

Proof Consider an instance of QBF_3, which we will remember is complete in Σ_3^P. The instance will have the form $\exists x \forall y \exists z : \phi(x, y, z)$. If we assume $NP \subseteq P/poly$ then we can solve the instance of QBP_3 in Σ_2^P by the following steps:

1. Use $\exists$ to guess each possible advice string α.

2. Use $\exists$ to guess each x.

3. Use $\forall$ to guess every y.

4. Test if α is *good*.

5. If so, use α to decide $(\exists z : \phi(x, y, z), \alpha_{|\exists z : \phi(x,y,z)|}) \in A$.

As we showed above, *good* $\in coNP$. We can therefore decide whether α is good using only $\forall$. Furthermore, $(\exists z : \phi(x, y, z), \alpha_{|\exists z : \phi(x,y,z)|}) \in A$ can be decided in P. Therefore, an ATM requires only one alternation starting from $\exists$ to decide an instance of QBF_3, showing $\Sigma_3^P \subseteq \Sigma_2^P$. This completes the proof of Theorem 4. ■