| 18.405J/6.841J Advanced Complexity Theory | February 23, 1998 |
|---|---|
| Lecture 6 | |
| *Lecturer: Daniel A. Spielman* | *Scribe: Matthew Secor* |

# NP and P/poly

The major result that we will show here is that if NP is contained within P/poly, then the polynomial hierarchy collapses. We will prove that if NP $\subseteq$ P/poly then $\Sigma_2 P = \Pi_2 P$. Before we prove this, let's review what this all means. Recall the definition of P/poly:

$$\begin{aligned} \text{P/poly} = \{L \quad : \quad & \exists k > 0, A \in P \text{ and a sequence of strings } \{S_n\}_{n=1}^{\infty} \text{ such that} \\ & |S_n| = \mathrm{O}(n^k) \text{ and } \forall w, w \in L \Longleftrightarrow (w, S_{|w|}) \in A\} \end{aligned}$$

Using this definition, we know the following, which will be useful later:

**Proposition 1**

$$\begin{aligned} \text{NP} \subseteq \text{P/poly} \Longrightarrow \text{SAT} \in \text{P/poly} \quad \Longrightarrow \quad & \exists k > 0, A \in P, \{S_n\}_{n=1}^{\infty} \textit{ such that} \\ & |S_n| \leq (kn)^k \textit{ and } \forall \phi, \phi \in \text{SAT} \Longleftrightarrow (\phi, S_{|\phi|}) \in A\ . \end{aligned}$$

We can think of the sequence of strings $\{S_n\}_{n=1}^{\infty}$ as advice strings which, in conjunction with the original input to SAT, allow us to decide SAT in polynomial time.

For the following definitions, we will assume that NP $\subseteq$ P/poly and thus SAT $\in$ P/poly. We begin with a definition:

**Definition 2** *A sequence of strings $(a_1, \ldots, a_\ell)$ is* good *if $|a_i| \leq (ki)^k$ and for all $|\phi| \leq \ell$,*

$$\phi \in \text{SAT} \Longleftrightarrow (\phi, a_{|\phi|}) \in A,$$

*where $A$ is defined as the machine in Proposition 1.*

Note that these good strings are not necessarily the same as the sequence of strings $\{S_n\}_{n=1}^{\infty}$ in Proposition 1. There could be many different good sequences of length $\ell$, although the sequence $\{S_n\}_{n=1}^{\ell}$ is, by definition, good.

**Definition 3** GOOD *is the language consisting of all of the good sequences of strings.*

**Proposition 4** GOOD $\in \Pi_2^{\mathrm{P}}$

**Proof** Given a candidate sequence, $(a_1, \ldots, a_\ell)$, we must show two things. First, we need to check that the size constraint is met ($|a_i| \leq (ki)^k$ for some $k$). This is easy. Second, we must show that the check to see if some sequence of strings is good can be done with a $\Pi_2^{\mathrm{P}}$ machine. In order to see this, consider the verification that needs to be done for the input $(a_1, \ldots, a_\ell)$. The following property must hold:

$$\forall \phi, |\phi| \leq \ell : \phi \in \text{SAT} \Longleftrightarrow (\phi, a_{|\phi|}) \in A\ .$$

If we can show that GOOD is in coNP$^{\text{SAT}}$, we are done because coNP$^{\text{SAT}} \subseteq \Pi_2^{\mathrm{P}}$. We can show this by noting that checking for $\phi \in$ SAT takes a single call to a SAT oracle, and checking $(\phi, a_{|\phi|}) \in A$ requires polynomial time. So, for each $\phi$ that we check, we require polynomial time with a call to a

SAT oracle, and we can use a $\forall$ state to perform the computation over each possible $\phi$, giving rise to a $\mathrm{coNP}^{\mathrm{SAT}}$ machine. ■

Now, we will introduce the concept of self-reducibility, needed to prove a stricter containment of GOOD.

**Definition 5** *A language $L$ is* self-reducible *if there exists a polynomial time oracle Turing machine $M^L$ such that*

*(i) $M^L$ accepts $L$, and*

*(ii) $M^L$ can only ask oracle about strings of length strictly less than the length of its input.*

According to this definition, almost all of the languages we have seen are self-reducible.[1] In particular, SAT is self-reducible.

**Proposition 6** SAT *is self-reducible.*

**Proof** In order to show this, we need to find out how to tell whether $\phi(x_1, \ldots, x_n)$ is satisfiable, if we only have access to an oracle that can tell us about formulas shorter than $\phi$. Consider the following two formulas:

$$\begin{aligned} \phi_0(x_2, \ldots, x_n) &= \phi(0, x_2, \ldots, x_n) \text{ , and} \\ \phi_1(x_2, \ldots, x_n) &= \phi(1, x_2, \ldots, x_n) \ . \end{aligned}$$

Given $\phi$, we can build $\phi_0$ and $\phi_1$ in polynomial time, and we also know that $|\phi_0|$ and $|\phi_1|$ are less than $|\phi|$. So, in order to check whether $\phi \in \mathrm{SAT}$, we only need to check if either $\phi_0$ or $\phi_1$ are in SAT, and accept if either is. Thus, we can construct a machine $M^{\mathrm{SAT}}$ which computes $\phi_0$ and $\phi_1$ and accepts only if one of them is satisfiable (determined with two calls to the SAT oracle). ■

We now use this definition of self-reducibility to show that $\mathrm{GOOD} \in \Pi_1^{\mathrm{P}}$, which will allow us to construct a $\Sigma_2^{\mathrm{P}}$ machine that decides $\exists\mathrm{QBF}_3$ if $\mathrm{SAT} \in \mathrm{P/poly}$.

**Lemma 7** $\mathrm{GOOD} \in \Pi_1^{\mathrm{P}}$

**Proof** We start by defining a Turing machine $M_{\mathrm{GOOD}}$, then we show that it accepts GOOD and that it is in $\Pi_1^{\mathrm{P}}$. Essentially, $M_{\mathrm{GOOD}}$ is trying to verify the following property for every $\phi$ such that $|\phi| \leq \ell$:

$$\phi \in \mathrm{SAT} \iff (\phi, a_{|\phi|}) \in A \ . \tag{1}$$

$M_{\mathrm{GOOD}}$ performs the following algorithm on each input $(a_1, \ldots, a_n)$:

**Algorithm:**

I **for all** $\phi$ such that $|\phi| \leq \ell$ (using a $\forall$ state)

II  check if $(\phi, a_{|\phi|}) \in A$

III  **if** $|\phi| = 1$, verify Property (1) directly,
  **accept** if the property is satisfied
  **reject** otherwise

IV  **if** $(\phi, a_{|\phi|}) \notin A$,
  **accept** if no assignment satisfies $\phi$ (using $\forall$ state to try all possible assignments)
  **reject** otherwise

V  **if** $(\phi, a_{|\phi|}) \in A$, self-reduce $\phi$ to get $\phi_0$ and $\phi_1$ and
  **accept** if $(\phi_0, a_{|\phi_0|}) \in A$ or $(\phi_1, a_{|\phi_1|}) \in A$
  **reject** otherwise

[1] There are languages in P/poly that are not self-reducible. Also, it is possible to choose languages in NP which are not self-reducible.

Step V requires some explanation. In order to show that $\phi \in \mathrm{SAT}$, by self-reducibility, it suffices to show that $\phi_0 \in \mathrm{SAT}$ or $\phi_1 \in \mathrm{SAT}$. If we know the algorithm works for shorter formulas, then checking that $(\phi_0, a_{|\phi_0|}) \in A$ is equivalent to checking whether $\phi_0 \in \mathrm{SAT}$. So, by an inductive argument, we can show that this will work for a formula of a certain length once we know it works for all shorter lengths.

Now, we need to prove this algorithm accepts GOOD. (i) and (ii) will establish this.

(i) If the input $(a_1, \ldots, a_\ell)$ is good, then the program accepts. This is clear from the definition of the algorithm.

(ii) If the program accepts, then the input is good. Assume (for contradiction) that it accepts, but the input is not good. Let $\phi$ have minimal length such that

(a) $(\phi, a_{|\phi|}) \in A$ but $\phi \notin \mathrm{SAT}$, or

(b) $(\phi, a_{|\phi|}) \notin A$ but $\phi \in \mathrm{SAT}$.

Then, if (b) were the case, we would reach step IV, but we would reject because a satisfying assignment would be found. If (a) were the case, we would reach step V, and produce $\phi_0$ and $\phi_1$ that must both not be satisfiable because $\phi$ has minimal length such that the program accepts incorrectly, so we would reject. The final case is for $|\phi| = 1$. In this case, step III provides the correct behavior.

Also, note that we only used $\forall$ states in the construction of $M_{\mathrm{GOOD}}$ and the computation required is polynomial in the input length, so that $M_{\mathrm{GOOD}}$ is in $\Pi_1^{\mathrm{P}}$.

■

**Theorem 8 (Karp and Lipton)** $\mathrm{SAT} \in \mathrm{P/poly} \Longrightarrow \Sigma_3^{\mathrm{P}} = \Sigma_2^{\mathrm{P}}$

**Proof** We will show that if $\mathrm{SAT} \in \mathrm{P/poly}$ then $\exists\mathrm{QBF}_3 \in \Sigma_2^{\mathrm{P}}$. If this is true, then because $\exists\mathrm{QBF}_3$ is $\Sigma_3^{\mathrm{P}}$-complete, $\Sigma_3^{\mathrm{P}} = \Sigma_2^{\mathrm{P}}$. Remember that strings in $\exists\mathrm{QBF}_3$ are true statements of the form

$$\exists x \forall y \exists z \ : \ \phi(x, y, z) \text{ is satisfiable}$$

We can determine whether a statement of this form is in $\exists\mathrm{QBF}_3$ using a Turing machine which implements the following algorithm:

**Algorithm:**

**choose** $x_0$ (using $\exists$ state)
**choose** $(a_1, \ldots, a_{|\phi|})$, $|a_i| \le (ki)^k$ (using $\exists$ state)
**for all** $y_0$ (using $\forall$ state)
  check that $(a_1, \ldots, a_{|\phi|})$ is good (using $M_{\mathrm{GOOD}}$ from above which only has $\forall$ states)
  compute $\hat{\phi}(z) = \phi(x_0, y_0, z)$
  **accept** if $(\hat{\phi}(z), a_{|\hat{\phi}(z)|}) \in A$
  **reject** otherwise

The basic idea is to choose $x$ and $y$ using $\exists$ and $\forall$ states, as well as a good sequence $(a_1, \ldots, a_{|\phi|})$. Then, we just have to check whether the simpler $\hat{\phi}(z)$ is in SAT in order to answer whether the overall statement is in $\exists\mathrm{QBF}_3$. To do this, we can use the definition of a good sequence which tells us that

$$\hat{\phi}(z) \in \mathrm{SAT} \Longleftrightarrow (\hat{\phi}(z), a_{|\hat{\phi}(z)|}) \in A$$

To show that this algorithm is correct, observe that if $\mathrm{SAT} \in \mathrm{P/poly}$, then there exists a good $(a_1, \ldots, a_{|\phi|})$; in particular, $\{S_n\}_{n=1}^{|\phi|} = (S_1, \ldots, S_{|\phi|})$ is good. In any accepting computation, you obtain such a string so the algorithm accepts if and only if the statement "$\exists x \forall y \exists z : \phi(x, y, z)$ is satisfiable" is true. We have described a Turing machine that starts with $\exists$ states, and switches only once to $\forall$ states, so the language decided by the machine is in $\Sigma_2^{\mathrm{P}}$. ■

Finally, we can prove our original statement.

**Theorem 9** $\mathrm{NP} \subseteq \mathrm{P/poly} \Longrightarrow \Sigma_2^{\mathrm{P}} = \Pi_2^{\mathrm{P}}$

**Proof** Theorem 8 tells us that $\Sigma_3^{\mathrm{P}} = \Sigma_2^{\mathrm{P}}$ if $\mathrm{NP} \subseteq \mathrm{P/poly}$. We also know then that $\Pi_3^{\mathrm{P}} = \Pi_2^{\mathrm{P}}$ because $\Pi_k^{\mathrm{P}} = \mathrm{co}\Sigma_k^{\mathrm{P}}$. From this we have that

$$\Pi_2^{\mathrm{P}} \subseteq \Sigma_3^{\mathrm{P}} = \Sigma_2^{\mathrm{P}}$$

and

$$\Sigma_2^{\mathrm{P}} \subseteq \Pi_3^{\mathrm{P}} = \Pi_2^{\mathrm{P}} \ ,$$

Which leads to $\Sigma_2^{\mathrm{P}} = \Pi_2^{\mathrm{P}}$ and a collapse of the polynomial hierarchy. ■