18.405J/6.841J Advanced Complexity Theory February 18, 1999

Lecture 5

Lecturer: Daniel A. Spielman *Scribe: Prahladh Harsha*

1 Review of previous lecture

In the last lecture, we defined the classes $\Sigma_{\mathrm{k}}^{\mathrm{P}}$, $\Pi_{\mathrm{k}}^{\mathrm{P}}$ and PH. $\Sigma_{\mathrm{k}}^{\mathrm{P}}$ is the class of languages L accepted by a polynomial time ATM that begins in an existential ($\exists$) state and alternates between existential ($\exists$) and universal ($\forall$) states atmost $k-1$ times. $\Pi_{\mathrm{k}}^{\mathrm{P}}$ is defined similarly except that it begins with an universal ($\forall$) state. We then gave another characterisation of $\Sigma_{\mathrm{k}}^{\mathrm{P}}$ and $\Pi_{\mathrm{k}}^{\mathrm{P}}$ as follows:

$$L \in \Sigma_{\mathrm{k}}^{\mathrm{P}} \text{ iff } \exists A \in \Pi_{\mathrm{k-1}}^{\mathrm{P}} \text{ and } c > 0 \text{ such that } w \in L \iff \exists_{(|y| \leq |x|^c)} y \text{ such that } (w, y) \in A$$

$\Pi_{\mathrm{k}}^{\mathrm{P}}$ was defined similarly, where $\Sigma_0^{\mathrm{P}} = \Pi_0^{\mathrm{P}} = \mathrm{P}$. We also proved the equivalence of both these definitions. In this lecture, we shall give yet another characterisation of the polynomial hierarchy. Recall, the $\Sigma_{\mathrm{k}}^{\mathrm{P}}$-complete and $\Pi_{\mathrm{k}}^{\mathrm{P}}$-complete languages $\exists\mathrm{QBF_k}$ and $\forall\mathrm{QBF_k}$ respectively, which we introduced in the last class.

$$\begin{aligned}
\exists\mathrm{QBF_k} = \{\phi : \phi = \exists\bar{x}_1\forall\bar{x}_2\exists\ldots Q_k\bar{x}_k f(\bar{x}_1, \bar{x}_2, \ldots \bar{x}_k) &\text{ where } Q_k \text{ is } \exists \text{ if } k \text{ is odd and } \forall \\
&\text{if } k \text{ is even; for some unquantified boolean formula } f \text{ and } \phi \text{ is true } \} \\
\forall\mathrm{QBF_k} = \{\phi : \phi = \forall\bar{x}_1\exists\bar{x}_2\forall\ldots Q_k\bar{x}_k f(\bar{x}_1, \bar{x}_2, \ldots \bar{x}_k) &\text{ where } Q_k \text{ is } \forall \text{ if } k \text{ is odd and } \exists \\
&\text{if } k \text{ is even; for some unquantified boolean formula } f \text{ and } \phi \text{ is true } \}
\end{aligned}$$

Finally, we proved the first of our PH theorems that $\Sigma_{\mathrm{k}}^{\mathrm{P}} = \Pi_{\mathrm{k}}^{\mathrm{P}} \Rightarrow \Sigma_{\mathrm{k+1}}^{\mathrm{P}} = \Sigma_{\mathrm{k}}^{\mathrm{P}}$.

2 Polynomial Hierarchy in terms of OTMs

In the first half of today's lecture, we shall see an alternate characterisation of PH in terms of OTMs. Recall the definition of OTMs and the corresponding class of languages accepted by them from last lecture.

$$\mathrm{NP}^A = \{L : \exists \text{ a nondeterministic polynomial time OTM } M^? \text{ such that } M^A \text{ accepts } L\}$$

The classes $\Sigma_{\mathrm{k}}^{\mathrm{P}}$, $\Pi_{\mathrm{k}}^{\mathrm{P}}$ can be defined using OTM and the languages $\forall\mathrm{QBF_k}$ as seen from the theorem below.

Theorem 1 $\mathrm{NP}^{\forall\mathrm{QBF_k}} = \Sigma_{\mathrm{k+1}}^{\mathrm{P}}$

Proof
(i) $\Sigma_{\mathrm{k+1}}^{\mathrm{P}} \subseteq \mathrm{NP}^{\forall\mathrm{QBF_k}}$: This is the easy direction of the proof. Since $\exists\mathrm{QBF_{k+1}}$ is $\Sigma_{\mathrm{k+1}}^{\mathrm{P}}$-complete, it is sufficient if we show that $\exists\mathrm{QBF_{k+1}} \in \mathrm{NP}^{\forall\mathrm{QBF_k}}$. The non-deterministic OTM $M^?$ that accepts $\exists\mathrm{QBF_{k+1}}$ (with access to oracle $\forall\mathrm{QBF_k}$) works as mentioned below. Let ϕ be any formula in $\exists\mathrm{QBF_{k+1}}$. Let it be of the form $\phi = \exists x_{11}x_{12}\ldots x_{1i_1}\psi = \exists\bar{x}_1\psi$ for some $\psi \in \forall\mathrm{QBF_k}$[1]. $M^?$ non-deterministically guesses the assignments, $\bar{t} = t_1t_2\ldots t_{i_1}$for the variables $x_{11}, x_{12}, \ldots, x_{1i_1}$ where

[1]Apologies for the abuse of notation: ψ in the strict sense does not belong to $\forall\mathrm{QBF_k}$ as it has a few unquantified variables in it.

$t_k \in \{true, false\}$ and then queries the oracle $\forall\mathrm{QBF_k}$ with the formula $\psi[\bar{x}_1 = \bar{t}]$ (ie., with the guessed assignments $t_1, t_2, \ldots, t_{i_1}$ substituted for the occurrences of the variables $x_{11}, x_{12}, \ldots, x_{1i_1}$ in ψ). $M^?$ accepts ϕ iff the oracle accepts $\psi[\bar{x}_1 = \bar{t}]$. Clearly, $L(M^{\forall\mathrm{QBF_k}}) = \Sigma^{\mathrm{P}}_{\mathrm{k+1}}$.

(ii) And now to the tougher direction, $\mathrm{NP}^{\forall\mathrm{QBF_k}} \subseteq \Sigma^{\mathrm{P}}_{\mathrm{k+1}}$: Consider any language $L \in \mathrm{NP}^{\forall\mathrm{QBF_k}}$. Let $M^?$ be a non-deterministic OTM with access to oracle $\forall\mathrm{QBF_k}$ that accepts L. To begin with, we shall assume that $M^?$ queries the oracle atmost once in each of its computation paths and then later deal with the general case. As $\forall\mathrm{QBF_k} \in \Pi^{\mathrm{P}}_{\mathrm{k}}$, there exists an ATM $\bar{N}$ such that $L(\bar{N}) = \forall\mathrm{QBF_k}$[2] and $\bar{N}$ begins in an universal ($\forall$) state and alternates between existential ($\exists$) and universal ($\forall$) states atmost $k-1$ times. To show, $L \in \Sigma^{\mathrm{P}}_{\mathrm{k+1}}$, we shall construct an ATM $\bar{M}$ using $M^?$ and $\bar{N}$ such that $L(\bar{M}) = L$ and $\bar{M}$ begins in an existential ($\exists$) state and alternates between existential ($\exists$) and universal ($\forall$) states atmost k times.

Construction of $\bar{M}$: On input w, $\bar{M}$ existentially gueses the non-deterministic choices (say, $\bar{y}$) of $M^?$ on input w. $\bar{M}$ then proceeds exactly as $M^?$ works, using $\bar{y}$ whenever $M^?$ has to make a non-deterministic guess. When $M^?$ has to query the oracle $\forall\mathrm{QBF_k}$ with the formula ϕ, $\bar{M}$ existentially (non-deterministically) guesses an answer (say, a), writes down both the formula ϕ and its guess "a" on its tape and continues with the simulation of $M^?$ as if the oracles's response to its query were "a". If $M^?$ rejects w, $\bar{M}$ also rejects w. Otherwise if $M^?$ accepts, $\bar{M}$ before accepting has to verify that the answer it guessed was right. Here $\bar{M}$ uses $\bar{N}$ to verify if $\phi \in \forall\mathrm{QBF_k}$. In case $a =$ "yes" (ie., it had guessed "yes"), $\bar{M}$ then works exactly as $\bar{N}$ would have worked on input ϕ. In other words, it accepts along a computation path iff $\bar{N}$ accepts along the corresponding computation path. In case $a =$ "no", $\bar{M}$ has to verify that $\phi \notin \forall\mathrm{QBF_k}$ or equivalently $\neg\phi \in \exists\mathrm{QBF_k}$. Suppose $\neg\phi = \exists\bar{x}\tilde{\phi}$. $\bar{M}$ existentially (non-deteministically) guesses the assignments (say, $\bar{t}$) for the variables corresponding to the first existential quantifier of $\neg\phi$ (ie., the variables in $\bar{x}$). Let $\phi' = \tilde{\phi}[\bar{x} = \bar{t}]$. $\bar{M}$ then checks that $\phi' \in \forall\mathrm{QBF_{k-1}}$ by working exactly as $\bar{N}$ would have worked on input ϕ'. Here, we are using the fact that $\forall\mathrm{QBF_{k-1}} \subset \forall\mathrm{QBF_k}$. Clearly by construction of $\bar{M}$, $L = L(\bar{M})$. Also $\bar{M}$ is an ATM that begins in the existential ($\exists$) and alternates between $\exists$ and $\forall$ states atmost once more than $\bar{N}$ alternates. (ie., $\bar{M}$ altenates atmost k times) Hence $L \in \Sigma^{\mathrm{P}}_{\mathrm{k+1}}$.

And now to the general case. The hurdle here is that the OTM $M^{\forall\mathrm{QBF_k}}$ can query the oracle several times along each of its computation paths. We shall get over this problem by simulating an OTM $M^?$ that can query $\forall\mathrm{QBF_k}$ several times by another OTM $N^?$ which queries the oracle atmost once along each of its computation paths and then the earlier proof works for this OTM $N^?$.

The rest of the proof will involve the construction of $N^?$ given $M^?$. $N^?$ works exactly as $M^?$ except for querying the oracle. Whenever $M^?$ queries the oracle $\forall\mathrm{QBF_k}$, $N^?$ instead guesses the answer of the oracle non-deterministically, writes down both the formula queried and its assumed answer on the tape and continues with the simulation of $M^?$. If $M^?$ rejects, $N^?$ too rejects. However, if $M^?$ accepts, $N^?$ verifies its guesses by querying the oracle in one shot and accepts only if its guesses were right. It is to be noted that the simulation so far has been independent of the oracle $\forall\mathrm{QBF_k}$, whereas the verification stage in which we have to verify all the guesses in one go is highly dependent on the oracle $\forall\mathrm{QBF_k}$.

Suppose $N^?$ has to verify its answers $a_1, \ldots a_n$ to the formula queries $\phi_1, \ldots \phi_n$ respectively. (each of the a_i's is either "yes" or "no"). The easy case is when all of the a_i's are "yes". In this case, if $N^?$ could produce a formula Φ such that $\Phi \in \forall\mathrm{QBF_k}$ iff $\phi_i \in \forall\mathrm{QBF_k}$ for $i = 1, 2, \ldots n$, then $N^?$ could just query the oracle with Φ instead of querying it for each of ϕ_i. This Φ can be produced easily by merely pasting together all the ϕ_is as shown below. (see example)

$$\Phi = \bigwedge_{i=1}^{n} \phi_i$$

Example Consider $\phi_1 = \forall x_1 \exists y_1 f_1(x_1, y_1), \phi_2 = \forall x_2 \exists y_2 f_2(x_2, y_2)$, (here $\phi_1, \phi_2 \in \forall\mathrm{QBF_2}$), then

[2] $L(M) = L$ for a Turing Machine M if M accepts the language L.

$\Phi(\in \forall\text{QBF}_2)$ could be got from pasting ϕ_1 and ϕ_2 together as

$$\Phi = \forall x_1 x_2 \exists y_1 y_2 \left(f_1(x_1, y_1) \wedge f_2(x_2, y_2)\right)$$

And now to the case where some of the a_is could be "no"s. Here $N^?$ has to verify that $\phi_i \in \forall\text{QBF}_\text{k}$ if a_i were "yes" and $\phi_i \notin \forall\text{QBF}_\text{k}$ if a_i were "no". Consider a formula

$$\phi_i = \forall x_{11} x_{12} \ldots x_{1i_1} \exists x_{21} \ldots x_{2i_2} \forall \ldots f_i(x_{11} x_{12} \ldots x_{1i_1} x_{21} \ldots x_{2i_2} \ldots)$$

when a_i were "no". In this case $N^?$ has to check that

$$\neg\phi_i = \exists x_{11} x_{12} \ldots x_{1i_1} \forall x_{21} \ldots x_{2i_2} \exists \ldots f_i(x_{11} x_{12} \ldots x_{1i_1} x_{21} \ldots x_{2i_2} \ldots)$$

is true. A mere pasting argument as before wouldn't suffice as seen from the example below.

Example Consider $\phi_1 = \forall x_1 \exists y_1 f_1(x_1, y_1), \phi_2 = \forall x_2 \exists y_2 f_2(x_2, y_2)$ (here $\phi_1, \phi_2 \in \forall\text{QBF}_2$)and a_1 = "no", a_2 = "yes". We have $\neg\phi_1 = \exists x_1 \forall y_1 \neg f_1(x_1, y_1)$. The Φ in this case is

$$\Phi = \forall x_2 \exists x_1 y_2 \forall y_1 \left(\neg f_1(x_1, y_1) \wedge f_2(x_2, y_2)\right)$$

Φ here belongs to $\forall\text{QBF}_3$ and not $\forall\text{QBF}_2$ as earlier.

This hurdle is got over by exploiting the non-determinism of the OTM $N^?$. $N^?$ nondeterministically guesses the assignments (say $t_1, \ldots, t_{i_1}$) of the variables (ie., $x_{11}, \ldots, x_{1i_1}$) corresponding to the first existential ($\exists$) quantifier of $\neg\phi_i$. Suppose $\neg\phi_i = \exists x_{11} x_{12} \ldots x_{1i_1} \tilde{\phi_i}$. Now, verifying $\neg\phi_i \in \exists\text{QBF}_\text{k}$ is equivalent to verifying $\phi_i' = \tilde{\phi_i}[x_{11} = t_1, \ldots, x_{1i_1} = t_{i_1}] \in \forall\text{QBF}_{\text{k}-1}$ for some $t_1, \ldots, t_{i_1}$. The queried formula Φ is obtained by pasting together the ϕ_i in the case a_i = "yes" and ϕ_i' in the case a_i = "no". Thus, Φ is given by

$$\Phi = \left(\bigwedge_{i:a_i=\text{“}yes''} \phi_i\right) \wedge \left(\bigwedge_{i:a_i=\text{“}no''} {\phi_i}'\right)$$

$N^?$ then verifies that its guesses were right, by querying the oracle with Φ. If the oracle answers "yes", then $N^?$ knows that its guesses were right and accepts and rejects otherwise. ■

3 Non-uniform Complexity

3.1 Circuit Complexity

Definition 2 *Given a boolean formula $f : \{0,1\}^a \leftarrow \{0,1\}^b$, the* **circuit complexity** *of f, given by C_f, is defined as follows:*

$$C_f = \min\{m : \exists \textit{ a circuit with } m \textit{ wires that computes } f\}$$

It is to be observed that a circuit, unlike a Turing Machine, can accept[3] only strings of a fixed size. Thus, to talk of the circuit complexity of language L, we have to consider a family of circuits.

Definition 3 *A language L is said to have* **polynomial circuit complexity** *if there exists a constant $k > 0$ and a sequence of circuits $\{C_n\}_{n=1}^{\infty}$ such that $|C_n| = O(n^k)$ and $\forall w \in \{0,1\}^*, w \in L \iff C_{|w|}(w) = 1$.*

Clearly, all languages in P have polynomial circuit complexity. The immediate question that arises then is whether 3SAT has polynomial circuit complexity. It is to be noted that this is not the same as asking whether P = NP, as polynomial circuits are more powerful than polynomial time Turing Machines. Though the answer to the above question is not known, it is not expected to be true as if 3SAT has polynomial circuit complexity, then the polynomial hierarchy collapses. (In fact $\Sigma_2^\text{P} = \Pi_2^\text{P}$). We shall prove this statement later.

[3] A circuit C is said to accept a string w iff $C(w) = 1$.

3.2 P/poly

Definition 4

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

P/poly can be thought of as the class of languages accepted by polynomial time Turing Machines which can use some polynomial sized (in terms of the input length) advice, dependent only on the length of the input. We now show that having polynomial sized advice is no different from being accepted by a family of polynomial circuits.

Theorem 5 $L \in \text{P/poly} \iff L$ *has polynomial circuit complexity*

Proof
(i)$\Leftarrow$: If L has polynomial circuit complexity, then there exists a family of circuits $\{C_n\}_{n=1}^{\infty}$ that accept L, where each $|C_n| = O(n^k)$ for some positive constant k. Consider a sequence of $\{S_n\}_{n=1}^{\infty}$, where each S_n is the encoding of the circuit C_n. Clearly, $|S_n|$ is polynomial in n. If CVAL is the circuit value language mentioned in lecture 1, we have $C_{|w|}(w) = 1 \iff (w, S_{|w|}) \in \text{CVAL}$. Hence $L \in \text{P/poly}$.
(ii) $\Rightarrow$: Assume $L \in \text{P/poly}$. Then there exists a sequence of strings $\{S_n\}_{n=1}^{\infty}$, where each $|S_n| = O(n^k)$ for some positive constant k and a polynomial time Turing Machine M that accepts the language $\{(w, S_{|w|}) : w \in L\}$. From the computation tableau of M on input $(w, S_{|w|})$, we can construct a circuit (as in Cook's construction) with the advice string S_n hardwired into the circuit that accepts inputs of length n. This family of circuits, so constructed, is clearly polynomial in size and accepts the language L. Thus L has polynomial circuit complexity. ■

In the rest of the lecture, we shall begin the proof of the fact we had mentioned earlier namely that if NP $\subseteq$ P/poly, then PH collapses.

Theorem 6 3SAT $\in$ P/poly $\Rightarrow \Sigma_2^{\text{P}} = \Pi_2^{\text{P}}$

Beginning of Proof If 3SAT $\in$ P/poly, then by definition there exists a constant $k > 0$, $A \in \text{P}$ and a sequence of strings $\{S_n\}_{n=1}^{\infty}$ such that $\forall n, |S_n| \leq (kn)^k$ and $\phi \in \text{3SAT} \iff (\phi, S_{|\phi|}) \in A$.

Definition 7 *A finite sequence of strings* $(a_1, a_2, \ldots, a_l)$ *is defined to be* good *if* $\forall 1 \leq i \leq l, |a_i| \leq (ki)^k$ *and* $\forall \phi$ *such that* $|\phi| \leq l$*, we have* $\phi \in \text{3SAT} \iff (\phi, S_{|\phi|}) \in A$. GOOD *is the language of all good sequence of strings.*

The first step in proving Theorem 6 will involve finding how efficiently can GOOD be decided. It can easily be shown from definition that GOOD $\in \Pi_2^{\text{P}}$. A more careful analysis will lead to the fact that GOOD $\in \Pi_1^{\text{P}}$. The proof of both these statements and the rest of the proof of Theorem 6 will be done in the next lecture.