18.405J/6.841J Advanced Complexity Theory February 13, 2000

Lecture 3

Lecturer: Shanghua Teng *Scribe: Hoeteck Wee*

Polynomial Hierarchy

In this lecture, we introduce the polynomial hierarchy, which generalizes the notions of P and NP. In particular, we provide 3 definitions of the polynomial hierarchy, which we claim are equivalent.

Definition 1 *Let $\Sigma_k\,TIME(f(n))$ be the class of languages L accepted by an Alternating Turing Machine (ATM) that begins in an existential ($\exists$) state, alternates between existential and universal states at most $k-1$ times, and halts (on all branches) within time $O(f(n))$. $\Pi_k\,TIME(f(n))$ is defined similarly, except that the ATM begins in a universal ($\forall$) state.*

We also define

$$\Sigma_k P = \bigcup_{i=1}^{\infty} \Sigma_k\,TIME(n^i)$$
$$\Pi_k P = \bigcup_{i=1}^{\infty} \Pi_k\,TIME(n^i)$$

The first level of this hierarchy makes up the familiar complexity classes, with $\Sigma_1 P = \text{NP}$ and $\Pi_1 P = \text{coNP}$. On the other hand, $\textit{MIN-FORMULA} = \{\langle\phi\rangle \mid \phi \text{ is a minimal Boolean formula}\} \in \Pi_2 P$.

Definition 2 $PH = \bigcup_{k=0}^{\infty} \Sigma_k P$.

Note that $PH = \bigcup_{k=0}^{\infty} \Pi_k P$, since $\Pi_k P \subset \Sigma_{k+1} P$ and $\Pi_k P \subset \Sigma_{k+1} P$. In addition, we have $\text{NP} \cup \text{coNP} \subset PH \subset AP = PSPACE$.

Definition 3 *First,* $\Sigma_0 P = \Pi_0 P = P$*; and for all* $k \geq 1$,

$$\begin{aligned}
\Sigma_k P &= \{A \mid \exists\, B \in \Pi_{k-1}P,\ c > 0 \text{ such that } x \in A \iff \exists\, y \text{ such that } (x,y) \in B \text{ and } |y| \leq |x|^c\} \\
\Pi_k P &= \{A \mid \exists\, B \in \Sigma_{k-1}P,\ c > 0 \text{ such that } x \in A \iff \forall\, y \text{ such that } (x,y) \in B \text{ and } |y| \leq |x|^c\}
\end{aligned}$$

Definition 4 *An oracle is a language A. An oracle Turing machine M^A is an ordinary Turing machine with an extra tape called the oracle tape. Whenever M writes a string on the oracle tape, it is informed whether that string is a member of A in a single computation step. Define P^A to be the class of languages decidable with a polynomial time oracle Turing machine that uses oracle A. Define NP^A and $coNP^A$ similarly.*

Definition 5 *First,* $\Sigma_0 P = \Pi_0 P = P$*; and for all* $k \geq 1$,

$$\Sigma_k P = NP^{\Sigma_{k-1}}$$
$$\Pi_k P = coNP^{\Sigma_{k-1}}$$
$$\Delta_k P = P^{\Sigma_{k-1}}.$$

Note that $\Sigma_k P = \text{co}\Pi_k P$. It follows that $\text{NP}^{\Sigma_{k-1}} = \text{NP}^{\Pi_{k-1}}$, since we may check whether a string is in $A \in \Pi_{k-1}P$ by asking the oracle tape whether it is in $\text{co}A \in \Sigma_{k-1}P$ and then negating the outcome. By the same argument, we have $\text{coNP}^{\Sigma_{k-1}} = \text{coNP}^{\Pi_{k-1}}$.

Next, observe that $MIN\text{-}COLOR = \{\langle G, k\rangle \mid k$ is the minimum number of colors needed to color the graph $G\} \in \Delta_2 P$. Checking whether a k-coloring of G exists can be done in NP, so we could for each $k = 1, 2, \ldots, n$ (where n is the number of vertices in G) query an oracle to check whether a k-coloring of G exists. On the other hand, there is no known NP or coNP algorithm for *MIN-COLOR*.

Theorem 6 *The definitions of* $\Sigma_k P$ *in 1 and 3 are equivalent.*

(The proof for this theorem is omitted; refer to the notes for the next lecture.)

Theorem 7 *The definitions of* $\Sigma_k P$ *and* $\Pi_k P$ *in 3 and 5 are equivalent.*

That 3 implies 5 is easy. For the converse, our oracle Turing machine may make polynomially many queries to the oracle tape nondeterministically, and intuitively, 3 allows us to make only one such query, but note that we could use a different $\Pi_k P$-language here. What we then do is to existentially guess the computation history made by the oracle TM as well as the certificates used by the queries to the oracle that returned "yes".

Proof By induction on k. $k = 0$ and $k = 1$ are straight-forward. Note that for each k, once we show that the 2 definitions of $\Sigma_k P$ equivalent, it follows that the 2 definitions of $\Pi_k P$ are equivalent, because $\Pi_k P = \text{co}\Sigma_k P$ in both definitions.

Now, consider $k \geq 2$. That definition 5 includes definition 3 is easy: the nondeterministic machine on input x guesses an appropriate y and makes a single query to check whether $(x, y) \in B$, where $B \in \Sigma_{k-1}P$.

Conversely, suppose $L \in \Sigma_k P$ according to definition 5, so L can be decided by a polynomial-time NTM M^K using as an oracle a language $K \in \Sigma_{k-1}$, and by the induction hypothesis, there exists a TM $C \in \Sigma_{k-2}$ and an integer c with the following property: $z \in K$ if and only if there exists w with $(z, w) \in C$ and $|w| \leq |z|^c$.

We may construct the TM B in definition 3 as follows. First, we know that $x \in L$ if and only if there exists a short accepting computation of M^K on x. Let y be a string recording such a computation of M^K, and $|y|$ is bounded by a polynomial in $|x|$. Recall that M^K is an oracle machine, so some of its steps will be queries to K, and some of these queries will have "yes" answers and the remaining ones "no" answers. For each "yes" query z_i, our certificate y also includes z_i's own certificate w_i such that $(z_i, w_i) \in C$. This forms our definition of B: $(x, y) \in B$ if and only if y records an accepting computation of M^K on x, together with a certificate w_i for each "yes" query z_i in the computation.

Next, we claim that checking whether $(x, y) \in B$ can be done in $\Pi_{k-1}P$ (by the induction hypothesis, we may pick definition 5 here). First, we need to check whether all steps of M^K are legal, which can be done in deterministic polynomial time. Then, we must check for polynomially many pairs (z_i, w_i) whether $(z_i, w_i) \in C$, but this can be done in $\Pi_{k-2}P$, which is in $\Pi_{k-1}P$. Finally, for all those "no" queries z'_i, we must check that $z'_i \notin K$, or equivalently, $z'_i \in \text{co}K$. However, $\text{co}K \in \text{co}\Sigma_{k-1}P = \Pi_{k-1}P$. We may therefore collapse all of these checks into a single $\Pi_{k-1}P$ computation. ■

Corollary 8 *The definitions of* $\Pi_k P$ *in 3 and 5 are equivalent.*

Definition 9 $TQBF_k^{\exists} = \{\langle\phi\rangle : \phi$ *is a satisfiable TQBF whose first quantifier is a* $\exists$*, with* $k-1$ *alternations of the quantifiers*$\}$. $TQBF_k^{\forall}$ *is defined similarly, except that the first quantifier is a* $\forall$.

Theorem 10 *For all $k \geq 1$, $TQBF_k^{\exists}$ is $\Sigma_k P$-complete.*

Proof We can inductively "unwind" definition 3:

$$\begin{aligned}\Sigma_k P \quad = \quad & \{A \mid \exists\, B \in P, c_1, \ldots, c_k \geq 1 \text{ such that } x \in A \iff \\ & \exists\, y_1 \quad |y_1| \leq |x|^{c_1} \\ & \forall\, y_2 \quad |y_2| \leq (|x| + |y_1|)^{c_2} \\ & \ldots \\ & \exists/\forall\, y_k \quad |y_k| \leq \left(|x| + \sum_{i=1}^{k-1} |y_i|\right)^{c_k} \\ & \text{ such that } (x, y_1, y_2, \ldots, y_k) \in B\}\end{aligned}$$

(where the last quantifier depends on the parity of k). It is then clear that $\text{TQBF}_k^{\exists} \in \Sigma_k P$ since we may evaluate a Boolean formula on a given input string in polynomial time.

Conversely, given any $L \in \Sigma_k P$ in this form, we may in polynomial time construct a Boolean formula that is equivalent to $B \in P$ via Cook's Theorem, with variables that correspond to the strings $y_1, y_2, \ldots, y_k$ so that we have an equivalent TQBF with the same $k - 1$ alternations of quantifiers. Hence, every language $L \in \Sigma_k P$ is reducible to $\text{TQBF}_k^{\exists}$ in polynomial time. ∎

By the same argument, we may show that $\text{TQBF}_k^{\forall}$ is $\Pi_k P$-complete for $k \geq 1$.

Theorem 11 *If $\Sigma_k P = \Pi_k P$ for some $k \geq 1$, then $\Sigma_j P = \Sigma_k P\ \forall\, j > k$, in which case we say that "the PH collapses."*

Proof It suffices to show that $\Sigma_k P = \Pi_k P$ implies $\Sigma_{k+1} P = \Pi_{k+1} P$, from which the result follows by induction on j. We will use definition 3. Let $A \in \Sigma_{k+1} P \Rightarrow \exists\, c > 0, B \in \Pi_k P$ satisfying: $x \in A \iff \exists\, y : |y| \leq |x|^c$ such that $(x, y) \in B$. If $\Pi_k P = \Sigma_k P, \exists\, c' > 0, C \in \Pi_{k-1} P$ satisfying: $x \in A \iff \exists\, y : |y| \leq |x|^c$ and $\exists\, y' : |y'| \leq (|x| + |y|)^{c'}$ such that $(x, y, y') \in C$. But then we can just combine y and y' into one string, which implies that $A \in \Sigma_k P$, and the proof is complete. ∎