18.405J/6.841J Advanced Complexity Theory	February 8, 2000
Lecture 3	
Lecturer: Daniel A. Spielman	*Scribe: Reid Andersen*

The Polynomial Time Hierarchy

Last lecture we saw that AP=PSPACE, since an Alternating Turing Machine (ATM) can solve TQBF in polynomial time using many alternations. By restricting the number of alternations, we can define complexity classes that might not collapse as easily into PSPACE.

Definition 1 *Let $\Sigma_k TIME(f(n))$ be the class of languages L accepted by an 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 $O(f(n))$. We also define*

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

Note that $NP = \Sigma_1 P$.

Definition 2 $\Pi_k TIME(f(n))$ *and* $\Pi_k P$ *are defined similarly, but the ATM must begin in a universal ($\forall$) state.*

Alternatively we may define $\Pi_k P = \text{co-}\Sigma_k P$. These definitions are equivalent. For consistency, we define the special cases $\Pi_0 P = \Sigma_0 P = P$. Note that co-$NP = \Pi_1 P$

By definition, $\Sigma_k P \subseteq \Pi_{k+1} P$ and $\Pi_k P \subseteq \Sigma_{k+1} P$, since a machine can alternate quantifiers once at the beginning of its computation. This prompts the following definition:

Definition 3 *The polynomial hiearchy $PH = \bigcup_k \Sigma_k P = \bigcup_k \Pi_k P$.*

Clearly PH $\subseteq$ AP = PSPACE, but it's an open question whether PSPACE $\subseteq$ PH.

As an example, consider MIN-FORMULA = {f: f is a boolean formula such that for any smaller boolean formula g, there exists an assignment of variables x such that f(x) is not equal to g(x)}

MIN-FORMULA $\in \Pi_2 P$, since we can make a turing machine that universally choosesa boolean formula g, and then switches to an existential state to determine if there is an assignment for which f(x) and g(x) are not equal. It's not immediately clear that MIN-FORMULA $\in NP$, and it probably isn't.

In contrast, consider the proof that PSPACE $\subseteq$ AP. It won't work to show PSPACE $\subseteq$ PH, since we needed an unbounded number of alternations to solve TQBF.

We now present an alternate set of definitions, inspired by the definition of NP as the class of languages with polynomial time verifiers:

Definition 4 *First, $\Sigma_0 P = \Pi_0 P = P$. Next,*

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

Theorem 5 *The two definitions of $\Sigma_k P$ are equivalent.*

Proof

We can inductively "unwind" the second definition, if k is even:

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

If k is odd, we get a similar definition.

clearly $\Sigma_k^{def2} P \subseteq \Sigma_k^{def1} P$, since a $\Sigma_k P$ machine (from definition 1) can guess the strings y_i in polynomial time, with just the right number of alternations.

to prove the reverse insclusion, we have the following lemma:

Lemma 6 *for any TM $M \in \Sigma_k TIME(f(n))$, there exists a TM $M' \in \Sigma_k TIME(f(n))$ that accepts the same language as M, and goes through all its alternations before consulting the work tape. (before doing anything else)*

Proof if M runs in $TIME(T(n))$, M' writes T(n) bits on the tape for each of its alternations, then simulates M deterministically, sequentially reading bits written during the alternation corresponding to the one M would be using to determine which branch of M's computation to follow. ∎

From this, it follows trivially that $\Sigma_k^{def1} P \subseteq \Sigma_k^{def2} P$, since a def2 machine can use the advice strings y_i to simulate M in the same manner that M' does.

∎

Let $TQBF_k^{\exists(\forall)} = \{\phi$: ϕ is a satisfiable TQBF whose first quantifier is $\exists(\forall)$, and the quantifiers switch between $\exists$ and $\forall$ at most k-1 times$\}$

Then $TQBF_k^{\exists(\forall)}$ is complete for $\Sigma_k P$.

Theorem 7 *If $\Sigma_k P = \Pi_k P$ then $\forall j > k, \Sigma_k P = \Sigma_j P$, and we say "the Polynomial Hierarchy collapses to the kth level". It is widely believed that the Polynomial Hierarchy does not collapse.*

Proof We will use definition 2. Let $A \in \Sigma_{k+1} P \Rightarrow \exists\, c > 0, B \in \Pi_k P$ such that $x \in A \iff \exists\, y : |y| \leq |x|^c$ such that $(x, y) \in B$. If $\Pi_k P = \Sigma_k P, \exists\, c' > 0, C \in \Pi_{k-1} P$ such that $x \in A \iff \exists\, y : |y| \leq |x|^c$ 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. ∎

Corollary 8 *if $\Sigma_k P \neq \Pi_k P$, then $NP \neq co-NP$.*

Finally, we look at another way of looking at $\Sigma_k P$ and $\Pi_k P$, based on oracle Turing Machines.

Definition 9 *An oracle TM (OTM) $M^?$ with oracle (a language) A has an oracle tape on which it can write a string w and enter an "oracle state" and, in one step, return to state "oracle yes" if $w \in A$ or "oracle no" if $w \notin A$.*

We say that A is Turing-reducible to B if there exists a Ptime oracle Turing machine $M^?$ such that M^B accepts A.

Note that $P^A = \{L \mid \exists$ a poly-time OTM $M^?$ that, on oracle A, accepts L$\}$. Consider that $P^{3\mathrm{SAT}} \supseteq NP$ (which is trivial to show), and also $P^{3\mathrm{SAT}} \supseteq$ co-NP since a machine can reject if the oracle replies "yes", and accept otherwise.

Similarly, $NP^A = \{L \mid \exists$ a nondet poly-time OTM $M^?$ that, on oracle A, accepts L$\}$.

Here's a definition of the PH involving oracle TMs:

Definition 10 $\Sigma_2 P = NP^{3SAT}$

we can also define NP^C for a class of langages C, by $NP^C = \{L$ s.t. $\exists B \in C$ and an NPtime OTM $M^?$ such that $L \in M^B\}$

using this, we define $\Sigma_k P = NP^{\Sigma_{k-1} P}$