18.405J/6.841J Advanced Complexity Theory	February 15, 2001

Lecture 4

Lecturer: Daniel A. Spielman — *Scribe: Matthew Lepinski*

Polynomial Hierarchy

Recall from last lecture the three definitions of the Polynomial Time Hierarchy.

Definition 1 *Σ_k^P is the class of all languages which are accepted by a polynomial time alternating turing machine which begins in a existential state and has at most $k-1$ alternations. Similarly, Π_k^P is the class of all languages which are accepted by a polynomial time alternating turning machine which begins in a universal state and has at most $k-1$ alternations.*

Definition 2

$$\begin{aligned}\Sigma_0^P &= \Pi_0^P = P \\ \Sigma_k^P &= NP^{\Sigma_{k-1}^P} \\ \Pi_k^P &= co-NP^{\Sigma_{k-1}^P}\end{aligned}$$

Definition 3 $\Sigma_0^P = \Pi_0^P = P$.

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

Definition 4 *The polynomial hierarchy $PH = \bigcup_k \Sigma_k^P = \bigcup_k \Pi_k^P$.*

In the previous lecture, we proved the equivalence of Definition 1 and Definition 2. In this lecture, we prove the equivalence of Definition 1 and Definition 3. In this proof we will make use of the following fact:

Fact 5 *Definition 3 is equivalent to $L \in \Sigma_k^P$ if, and only if $\exists c > 0$ and $A \in P$ such that $x \in L \Leftrightarrow \exists y_1 \forall y_2 \exists y_3 \ldots (\exists/\forall) y_k\ (x, y_1, y_2, \ldots, y_k) \in A$ and $|y_i| < |x|^c$.*

To see that this fact is true, first unroll the recursion in Definition 3. Then observe that if $|y_{i+1}|$ is polynomial in $(|y_i| + |y_{i-1} + \ldots + |x|)$ and $|y_j|$ is polynomial in $|x|$ for $j \leq i$, then $|y_{i+1}|$ is polynomial in $|x|$.

Theorem 6 *Definition 1 is equivalent to Definition 3.*

Proof The main difference between Definition 3 and Definition 1 is that in Definition 3 the machine is required to so through all of its Existential and Universal states before it does any other computation. Whereas in Definition 1 the machine is allowed to interleave computation, Universal states and Existential states however it sees fit. (As long as there are at most $k-1$ alternations.)

With this in mind, it is clear that Definition 1 includes Definition 3. This is because the alternating machine in Definition 1 is allowed to do it's alternation before it's other computation and thus can recognize any language of the form specified by Fact 1.

To see that Definition 3 includes Definition 1, consider a language L which is accepted by a polynomial time alternating turing machine M which starts in an existential state and alternates at most $k-1$ times. We will construct a constant c and a language $A \in P$ which satisfies Definition 3.

Since M runs in polynomial time, there is a polynomial $p(n)$ which bounds it's running time. Let c be a constant such that for all sufficiently large n, $p(n) < n^c$.

We now construct a turing machine M' (we think of M' as simulating M without the non-determinism). We will define A to be the language accepted by M'. Therefore, we construct M' to take inputs of the form $(x, y_1, y_2, \ldots, y_k)$. M' begins simulating M on x and uses the following rules to handle the existential and universal states of M.

- When M would enter a existential state, M' reads a bit from auxiliary input y_1 and follows the branch zero if the bit is 0 and follows the branch one if the bit is 1.
- Continue using y_1 to determine which branch to follow up until the point where M enters it's first universal state.
- When M would enter a universal state, M' reads a bit from auxiliary input y_2 and follows the branch zero if the bit is 0 and follows the branch one if the bit is 1.
- Continue using y_2 to determine which branch to follow up until the point where M enters it's next existential state.
- Continue in this fashion advancing to the next auxiliary input each time M alternates between existential and universal states (or vice versa). Note: Since M is guaranteed to have at most $k-1$ alternations, M' will not run out of auxiliary inputs.

By the construction of M', if M' accepts $(x, y_1, y_2, \ldots, y_k)$ for some y_1 and all y_2 and some y_3 $\ldots$, then M would accept x. ■

The Polynomial Time Hierarchy Collapses

Proposition 7 *If $\Sigma_k^P = \Pi_k^P$ then $\Sigma_k^P = \Sigma_j^P$ for all $j > k$.*

Proof In this proof we use Definition 3. (Since we've proved all three definitions equivalent, we can use whichever is most convenient.) Additionally, we prove only the case where $j = k+1$, all other $j's$ follow easily by induction.

$L \in \Sigma_{k+1}^P \Rightarrow \exists c,\ A \in \Pi_k^P$ such that $x \in L$ if and only if $\exists (x, y) \in A$ with $|y| < |x|^c$.

Since $\Sigma_k^P = \Pi_k^P$, we know $A \in \Sigma_k^P$. Therefore, $\exists d,\ B \in \Pi_{k-1}^P$ such that $(x, y) \in A$ if and only if $\exists (x, y, z) \in A$ with $|z| < |x, y|^d$.

Therefore, $x \in L$ if and only if $\exists y\ \exists z$ such that $(x, y, z) \in B$ and $|y| < |x|^c$ and $|z| < |x, y|^d$.

Now we combine the two existential quantifiers into a single existential quantifier to get $x \in L$ if and only if $\exists (y, z)$ such that $(x, (y, z)) \in B$ and $|(y, z)| < |x|^{cd+1}$.

Since, B is a Π_{k-1}^P language, L is in Σ_k^P by the definition of Σ_k^P. ■

If the hypothesis of the previous proposition is satisfied (that is, if $\Sigma_k^P = \Pi_k^P$) we say that the Polynomial Time Hierarchy collapses to the k^{th} level. This is the complexity theorist's way of saying "Horses Can Fly!". It has not been proven that the hierarchy does not collapse, however, complexity theorists universally believe that such a collapse does not occur.

Many theorems in complexity theory take the form "If Pigs Could Whistle Then Horses Could Fly!". That is, a complexity theorist might want to prove a result like NP is NOT in P/POLY (i.e. Pigs Can't Whistle) but finds herself unable to do so. Therefore, the complexity theorist instead proves the result "If NP in P/POLY then the Polynomial Time Hierarchy Collapses" (i.e. If Pigs Could Whistle Then Horses Could Fly). Since no one believes that the hierarchy collapses (i.e. no one believes that horses can fly), this result is taken as strong evidence that NP is NOT in P/Poly (i.e. strong evidence that Pigs Can't Whistle).

In the next lecture we will prove the first theorem of this type. However, first we must define Non-Uniform Complexity.

Non-Uniform Complexity

Definition 8 *For a function $f : \{0,1\}^n \to \{0,1\}$, the Circuit Complexity of f is the minimum m such that there exists a circuit with m AND, OR and NOT gates that computes f*

We denote the Circuit Complexity of f by $C(f)$.

Definition 9 *A language, L, has Polynomial Circuit Complexity if there exists a k such that $C(f_n) = O(n^k)$ where $f_n = 1 \Leftrightarrow x \in L$ and $|x| = n$.*

Observe that Polynomial Circuit Complexity can give you undecidable languages. For, instance the language $L = \{x : \text{Turing Machine number } |x| \text{ halts}\}$ has Polynomial Circuit Complexity.

Definition 10 *$L \in P/Poly$ if there exists $A \in P$, $k > 0$ and an infinite sequence of strings $\{S_n\}$ such that $|S_n| = O(n^k)$.*

Theorem 11 *$L \in P/Poly$ if and only if L has Polynomial Circuit Complexity.*

Proof If L has Polynomial Circuit Complexity then there exists a sequence of circuits $\{C_n\}$ and some $k > 0$ such that $|C_n| = O(n^k)$ and for all $|x| = n$, $x \in L \Leftrightarrow C_n(x) = 1$.

To show that Polynomial Circuit Complexity is contained in P/Poly, we just let S_n be a description of C_n and let A be CVAL (the Circuit Value Problem). Observe that $x \in L \Leftrightarrow (x, C_{|x|}) \in CVAL$.

To show the containment of P/Poly in Polynomial Circuit Complexity, let L be a language in P/Poly and let M be a Turing Machine that accepts A. Then $x \in L \Leftrightarrow M(x, S_{|x|})$ accepts. Since $A \in P$ we know that there exists some polynomial $p(n)$ which bounds the running time of M.

Recall the proof that CVAL is P-Complete. From that proof we know that we can transform M into a circuit, C, with input $(x, S_{|x|})$ and size $O(p(n)^2)$ such that $C(x, S_{|x|}) = 1$ if and only if $M(x, S_{|x|})$ accepts. Now we choose C_n to be the circuit C with S_n hardwired in as the second input.
■