**18.405J/6.841J Advanced Complexity Theory** February 10, 2000

Lecture 4

*Lecturer: Dan Spielman* *Scribe: David Manz*

# 1 Review

We are discussing the definitions of the polynomial hierarchy. The first thing to do in this lecture is to show that Definition 3 is equivalent to Definition 1. To review, the definitions are:

**Definition 1** $\Sigma_k P$ = a language L that is recognized by an ATM, which begins in an $\exists$ state and alternates at most $k-1$ times.

**Definition 3** $\Sigma_k P = NP^{TQBF^{\forall}_{k-1}} = NP^{TQBF^{\exists}_{k-1}}$

# 2 Equivalence of Definitions

**Terminology Note:**
$QBF$ is a set of things, $TQBF$ is the language of true $QBF$s.
$QBF^{\exists}_k$ is a set of things, $TQBF^{\exists}_k$ is the language of true $QBF^{\exists}_k$s.

**Theorem 1** *Definition 1 $\subseteq$ Definition 3*

**Claim 2** $TQBF^{\exists}_k \subseteq NP^{TQBF^{\forall}_{k-1}}$

**Proof** Consider a $QBF^{\exists}_k : \exists x_1 \forall x_2 \ldots \exists x_k A(x_1, x_2, \ldots x_k)$. The NP machine can guess an assignment for $x_1$ and accept if $A(\underline{x_1}, x_2, \ldots x_k) \in TQBF^{\forall}_{k-1}$, which is determined by a call to the oracle. ■

**Theorem 3** *Definition 1 $\supseteq$ Definition 3*

The problem is that a NOTM $M^?$ can ask the oracle many questions. We need to condense this down to just one question. First, we will prove the special case, $NP^{TQBF^{\forall}_1} \subseteq \Sigma_2 P$ (this is co-SAT).

**Claim 4** *We can combine instances of calls to a $TQBF^{\forall}_1$ oracle.*

We want to know if $\forall \overline{x} \Phi(x)$ is true ($\Phi(x)$ is some boolean formula). However, we also want to find out if $\Theta(y)$ is true. We can ask both as: $\forall \overline{x}, \overline{y}(\Phi(x) \bigwedge \Theta(y))$. This is not enough to prove the theorem. There are two problems with this approach:

1. **Problem:** It could be that $M^?$ accepts *iff* first is true and second is false ($\neg \forall y \Theta(y) \rightarrow \exists y \neg \Theta(y)$)
   **Solution:** We want to know $\forall x \Phi(x) \bigwedge \exists y(\neg \Theta(y))$. The NP machine can guess y.
2. **Problem:** We don't know the questions in advance, and one can depend on another.
   **Solution:** Guess what all questions to the oracle will be, and guess their answers; simulate the original machine, reject if you guessed the wrong questions, else use your answers guessed. At the end, we need to verify the answers and accept if they were correct and the simulated machine accepted. So, we get a polynomial number of questions and answers to verify. We need to verify $\forall x \Phi_1(x) \bigwedge \forall x \Phi_3(x) \bigwedge \ldots \forall x \Phi_k(x) \bigwedge \neg(\forall x_2 \Theta(x)) \bigwedge \ldots \neg(\forall x_j \Theta)$

. The formulae subject to the universal quantifiers can be lumped together with ANDs, and a negated universal formula translates into the equivalent existential expression with the formula negated (as described in the solution to the preceding problem).

Now for the general case.

**Claim 5** *$NP^{TQBF^{\forall}_{k-1}} \subseteq \Sigma_k P$.*

We did $k = 2$. In the general case, the first step of combining "yes" instances is fine, so we just have to get answers to $QBF^{\exists}_{k-1}$. We will use the NP OTM to guess first quantifier, and then the rest gets globbed into the one question–we do a lot of pre-guessing and then reject where guesses were incorrect.

## 3 Circuit Complexity

**Definition 6** *For a function $f : \{0,1\}^n \to \{0,1\}^b$ the circuit complexity of $f$, $C(f)$, is the least $m$ such that there exists a circuit (boolean, binary, $\bigvee$, $\bigwedge$, $\neg$) with $m$ gates that computes $f$.*

**Definition 7** *A language $L$ has polynomial circuit complexity if there exists $k$ such that the functions $f_n : \{0,1\}^n \to \{0,1\}$ such that*

$$f_n(x) = \begin{cases} 1 \ |x| = n \text{ and } x \in L \\ 0 \text{ otherwise} \end{cases}$$

$$C(f_n) = O(n^k)$$

**Theorem 8** *(Karp-Lipton-Sipser)*
*If an NP-complete language has polynomial circuits, then $\Sigma_2 P = \Pi_2 P$.*

**Definition 9** *$L \in P/poly$ if there exists a constant $k > 0$ and an infinite sequence of strings $\{S_i\}_{i \in N}$ and $A \in P$ and $|S_i| = O(i^k)$ such that $w \in L \leftrightarrow (w, S_{|w|}) \in A$. ($S_i$ is an "advice string")*

**Proposition 10** *$L \in P/poly \leftrightarrow P$ has polynomial circuit complexity.*

**Proof** $\leftarrow$
Let $S_i$ be an encoding of a circuit that computes $f_n$ of size $O(n^k)$ and $A = \{(w, S)$ such that the circuit encoded by $S$ accepts string $w\}$.■

**Proof** $\rightarrow$
Consider M that accepts A. Consider the computation tableau of M accepting A, convert it into a circuit (a different circuit for each different size input), and hardwire in the advice string, S.■

**Theorem 11** *If $NP \subseteq P/poly$, then $\Sigma_2 P = \Pi_2 P$.*

**Proof** $NP \subseteq P/poly \rightarrow SAT \subseteq P/poly \rightarrow \exists k, A \in P, \{S_n\}_{n \in N}$ such that $|S_n| \leq (kn)^k$ and $w \in SAT \leftrightarrow (w, S_{|w|}) \in A$. ■

**Definition 12** *Let $(g_1, \ldots g_l)$ be a sequence of strings. $(g_1, \ldots g_l)$ is good if:*

1. $|g_i| \leq (ki)^k$
2. $\forall w : |w| \leq l$,
   $w \in SAT$ iff $(w, g_w) \in A$

## 4 For Next Time

Define $GOOD$ = the language of *good* strings.

What is the complexity of $GOOD$?

$GOOD \in \Pi_1 P = coNP$

First, prove $GOOD \in \Pi_2 P$ (easy)