6.841/18.405J: Advanced Complexity	**Wednesday, Feburary 19th, 2003**
Lecture 5	
Instructor: Madhu Sudan	*Scribe: Steven Stern*

Problem Set 2 available today. Due in 2 weeks, on March 5th.

1. Polynomial Hierarchy
2. PH does not collapse?
3. Circuit Complexity
4. Karp-Lipton Theorem

1 Polynomial Hierarchy

Definitions

Σ_i^P =Languages accepted by polynomial time bounded ATM starting in existential state with i alternating quantifiers.
Π_i^P =Languages accepted by polynomial time bounded ATM starting in universal state with i alternating quantifiers.

$$PH = \bigcup_{i \geq 0} \Sigma_i^P$$

$\Sigma_1^P = NP$, $\Pi_1^P = coNP$ and by convention, $\Sigma_0^P = \Pi_0^P = P$. $MINDNF \in \Sigma_2^P$ (this was also shown to be complete for the class by Umans[U98]).

Since we can always add "null" quantifiers, we know that $\Sigma_{i-1}^P \subseteq \Pi_i^P \subseteq \Sigma_{i+1}^P$. We can also define this class as $\Pi_i^P = \{\bar{L} | L \in \Sigma_i^P\}$. Since NP allows for one existential state, we can say that $\Sigma_i^P = NP^{\Pi_{i-1}^P}$.

Furthermore, since PH is an infinite union, we can also define it as $PH = \bigcup_{i \geq 0} \Pi_i^P$

Complete Problems

There exists a complete problem for each of these classes, defined as follows:

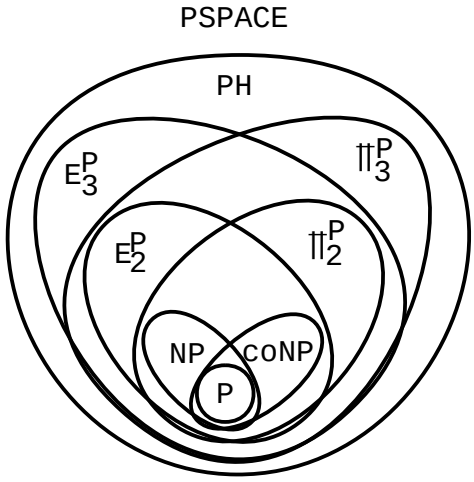
$$i\exists TQBF = \{\phi | \exists x_1 \forall x_2 \exists x_3 ... Q_i x_i \phi(x_1, x_2, ..., x_i)\}$$

$$i\forall TQBF = \{\phi | \forall x_1 \exists x_2 \forall x_3 ... Q_i x_i \phi(x_1, x_2, ..., x_i)\}$$

where ϕ is a $3CNF$ formula, and $x_1, x_2, ..., x_i$ are blocks of variables. $i\exists TQBF$ is Σ_i^P complete, and $i\forall TQBF$ is Π_i^P complete, for polynomial time reductions ($\forall i \geq 1$)

2 PH does not collapse?

We believe (but haven't proven) that $\Sigma_i^P \neq \Pi_i^P$, $\forall i \geq 1$.



Theorem 1 *If* $\Sigma_i^P = \Pi_i^P$, $\forall j \geq i$, $\Sigma_j^P = \Pi_j^P = \Sigma_i^P$, *and thus,* $PH = \Sigma_i^P$.

Proof: By induction on j. True (by assumption) for $j = i$. Let $j > i$ and assume true for $j - 1$.

We know that Σ_j^P contains every TM M, where M makes j alternating steps. We can rewrite this as:

$$L = \{x | \exists y \text{ s. t. } N(x, y) \text{ accepts, where } N \text{ makes } j-1 \text{ alternating steps starting in a } \forall \text{ state}\}$$

But, since $\Sigma_{j-1}^P = \Pi_{j-1}^P$, there must exist $N'(x, y)$ which accepts the same language as N, but makes $j - 1$ alternations and starts in an $\exists$ state.

$$L(N') = \{(x, y) | \exists z \text{ s. t. } N''(x, y, z) \text{ accepts, } N'' \text{ makes } j-2 \text{ alternations and starts in a } \forall \text{ state}\}$$

$$L = \{x | \exists y, z \text{ such that } N''(x, y, z) \text{ accepts}\}$$

$$L \in \Sigma_{j-1}^P$$

3 Circuit Complexity / Non-Uniform Computation

Circuit Complexity

How small a circuit can we build to decide SAT?

We define circuits which take boolean inputs, of size n, and produce a boolean output, of size m. That is, $\{0,1\}^n \Rightarrow \{0,1\}^m$. A circuit is represented by a DAG (Directed Acyclic Graph), with the following properties:

1. n "input" vertices (have in-degree=0), labelled $x_1, x_2, ..., x_n$.
2. Many intermediate nodes (gates), of in-degree=1 (the NOT function) or in-degree=2 (the AND, OR functions). If these have out-degree= 1, it is called a formula. If out-degree> 1, it is a circuit.
3. m designated outputs.
4. Size is defined as the number of gates.

However, the circuit of interest to us is one that decides SAT. That is, let $|\phi| = n$, and $SAT_n : \{0,1\}^n \Rightarrow \{0,1\}$, can we design a small hardware circuit to solve this problem?

We define the complexity class corresponding to circuits as follows:

$$\text{SIZE}(s(n)) = \{L| \quad \text{There exists an infinite family of circuits } C_1, C_2, \ldots, C_n, \ldots \text{ such that } |C_i| \leq s(i), \forall i \text{ and } x \in L \iff C_{|x|} = 1\}$$

In other words, SIZE($s(n)$) is the set of all languages that can be decided by a family of circuits of size $s(n)$.

Non-Uniform Complexity

Can a little "advice" help solve hard problems?

For certain problems, this advice certainly can help. The problem of deciding if a TM halts on the input 0^n can be decided with non-uniformity. However, it is believed that no "nice" problems, such as SAT, can be solved more efficiently with non-uniformity.

We define a TM that uses non-uniformity as: $M(\bullet, \bullet)$, where the first argument passed is the "advice", represented as a string, and the second argument is the input. There is a different "advice" string for each input size.

Using a Circuit as the "Advice"

The interesting problem to examine is determining if there exist advice strings, $a(1), a(2), ..., a(n)$, such that $M(a(n), \phi) = 1$ iff $\phi \in SAT$. The running time of M must be polynomial time, and the size of $a(n)$ must be polynomial in n, for this problem to be interesting. If either were allowed to be exponential, then the problem becomes trivial.

Definition: $L \in P/poly$ if there exists a polynomial time bounded Turning Machine, M, polynomial p, and advice strings $a_1, a_2, ..., a_n, ...$ with $|a_n| \leq p(n)$ such that for every $x \in \{0,1\}^n$, $x \in L \Leftrightarrow M(x, a_{|x|}) = 1$.

Note that $P/poly$ is exactly the class of languages that are computable by a family of circuits of polynomial size. This equivalence is noted by observing the following facts. (i) The circuit can serve as the advice for each n. (ii) The advice for each n can be hardwired into the corresponding circuit. In other words, $P/poly = \cup_{d \geq 0} \text{SIZE}(n^d)$.

4 Karp-Lipton Theorem

Clearly, if $P = NP$, the SAT has polynomial sized cicuits, we'll show a weak converse, namely that if SAT has polynomial sized circuits, then the polynomial hierarchy collapses.

Theorem 2 *$NP \not\subseteq P/poly \Rightarrow NP \neq P$, and $NP \subseteq P/poly \Rightarrow PH$ collapses to some finite level. That is, $PH = \Sigma_k^P$ for some finite k.*

First for some definition,

$$M\text{-}GOOD = \{a_n | \text{if } M(a_n, \bullet) \text{ decides } SAT \text{ on } n\text{-length inputs}\}$$

The following two lemmata prove the above theorem.

Lemma 1 *Deciding if $a_n \in M$-$GOOD$ is in Π_i^P for some i*

Lemma 2 *if $NP \subseteq P/poly$ and $M\text{-}GOOD \in \Pi_i^P$ for some i, then $\Sigma_{i+2}^P = \Sigma_{i+1}^P$*

Proof of Lemma 1: Observe that

$$a_n \in M\text{-}GOOD \iff \forall\phi, \big(M(a_{|\phi|}, \phi) = 1 \iff \exists\alpha, \phi(\alpha) = 1\big).$$

Equivalently,

$$a_n \in M\text{-}GOOD \iff \forall\phi, \Big[\big((M(a_{|\phi|}, \phi) = 1) \wedge (\exists\alpha, \phi(\alpha) = 1)\big) \vee \big((M(a_{|\phi|}, \phi) = 0) \wedge (\forall\rho, \phi(\rho) = 0)\big)\Big]$$

Or equivalently,

$$a_n \in M\text{-}GOOD \iff \forall\phi, \rho\exists\alpha \Big[\big((M(a_{|\phi|}, \phi) = 1) \wedge (\phi(\alpha) = 1)\big) \vee \big((M(a_{|\phi|}, \phi) = 0) \wedge (\phi(\rho) = 0)\big)\Big]$$

and the above computation can be done in Π^P

2

mplicity, that i is odd. By definition, $(i+2)\exists TQBF = \{\phi | \exists x_1 \forall x_2 ... \exists x_{i+2} \phi(x_1, x_2, ..., x_{i+2})\}$.
we can examine $\psi(x_{i+2}) = \phi(x_1, x_2, ..., x_{i+2})$. We will also use a *M-GOOD*
nine if $M(\psi, a_{n+1}) = 1$. We are going to guess an *M-GOOD* string, and
e formally:

for ϕ:

$\in$ *M-GOOD*
$\exists x_3 ... \forall x_{i+1} M(\psi, a_n) = 1$, where $\psi(\bullet) = \phi(x_1, x_2, ..., x_{i+2})$

inimum Equivalant DNF Problem and Shortest Implicants. In *Proceedings*
IEEE Symposium on Foundations of Computer Science (FOCS). pages