CS 6810: Theory of Computing — Fall 2021

Lecture 11: Sep 30, 2021

Lecturer: Eshan Chattopadhyay — *Scribe: James Meyers*

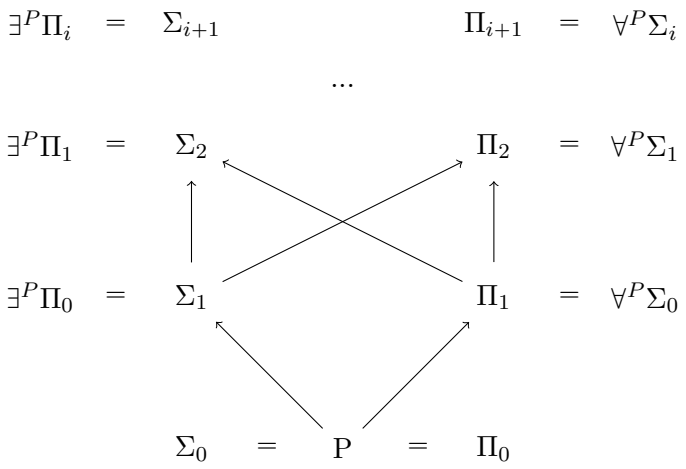
1 More on the Polynomial Hierarchy

In previous class, we introduced two distinct unary operators on complexity classes, $\exists^P$ and $\forall^P$. For any complexity class C, and for any language $L \subseteq \{0,1\}^*$:

Definition 1.1. $L \in \exists^P C \iff (\exists L' \in C(\forall x(x \in L \iff \exists y \in \{0,1\}^{poly(|x|)} \langle x, y\rangle \in L')))$.

Definition 1.2. $L \in \forall^P C \iff (\exists L' \in C(\forall x(x \in L \iff \forall y \in \{0,1\}^{poly(|x|)} \langle x, y\rangle \in L')))$.

Recall the visual representation of the polynomial hierarchy. Below, directed arrows represent subsethood.



Definition 1.3. *The polynomial hierarchy (PH) collapses to its i^{th} level if $\exists i \in \mathbb{N}$ such that $PH = \Sigma_i$.*

We believe that PH does not collapse, and this is often used as evidence towards other conjectures in complexity theory. For instance, we will show that if all languages in NP has polynomial sized circuits, then PH collapses (to the second level).

We begin by proving some easy claims that show properties of PH.

Claim 1.4. *$P = NP$ implies that PH collapses to its 0^{th} level.*

Proof. Assume P = NP. This can be proven by induction. Let the inductive hypothesis, $P(i)$ be that $\Sigma_i, \Pi_i \subseteq$ P. For $P(1)$, we have already that NP $= \Sigma_1 \subseteq$ P. Given some $L \in \Pi_1$. We have $\bar{L} \in$ NP $\therefore \bar{L} \in$ P $\therefore L \in$ coP $=$ P. Now given some $i \geqslant 1$, we show $P(i) \implies P(i+1)$. Given some $L \in \Sigma_{i+1}$. By definition, $L \in \exists^P \Pi_i$. Since $\Pi_i \subseteq$ P by the IH, and P $\subseteq \Pi_i$ for all i, we have $L \in \exists^P$P $\therefore L \in$ NP $\therefore L \in$ P. Given some $L \in \Pi_{i+1}$, by definition $L \in \forall^P \Sigma_i \therefore L \in \forall^P$P (by the IH) $\therefore L \in$ coNP $\therefore L \in$ P (by the IH, coNP $\subseteq$ P). So, since $\forall i \geqslant 1$, $\Sigma_i \subseteq$ P, clearly PH $= \bigcup_{i \in \mathbb{N}} \Sigma_i =$ P. □

Claim 1.5. *NP = coNP $\implies$ PH = NP.*

Proof. Again, we use induction. First, assume NP = coNP. $\forall i \geqslant 1, P(i)$ is that $\Sigma_i, \Pi_i \subseteq \Sigma_1$. For $P(1)$, this is immediate. Given some i, we show $P(i) \implies P(i+1)$. Assume $P(i)$. Given some $L \in \Sigma_{i+1} \therefore L \in \exists^P \Pi_i$. By the induction hypothesis, then, $L \in \exists^P \text{NP} = \text{NP}$. Given some $L \in \Pi_{i+1} \therefore L \in \forall^P \Sigma_i$. Again, the IH gives us $L \in \forall^P \text{NP} = \forall^P \text{coNP} = \text{coNP} = \text{NP}$. So, $\forall i \geqslant 1$, we have $\Sigma_i, \Pi_i \subseteq \Sigma_1$. Then, clearly $PH = \bigcup_{i \in \mathbb{N}} \Sigma_i = \Sigma_1$. □

Corollary 1.6. *For all positive i, $(\Sigma_i = \Pi_i \implies PH = \Sigma_i = \Pi_i)$.*

This can be seen by simply repeating the induction performed above, and changing the base case based on any particular given i (showing $P(n)$ for all $n \geqslant i$). For any $o < i$, $\Sigma_o \subseteq \Sigma_i$ follows from the definition of PH.

Claim 1.7. *If PH has a complete problem, then it collapses.*

Proof. Suppose that L is PH-complete. Then for some i, $L \in \Sigma_i \vee L \in \Pi_i$. Assume $L \in \Sigma_i$. Given any $L' \in \text{PH}$, we have $L' \leqslant_P L$. So, $L' \in \Sigma_i$ and thus $\text{PH} \subseteq \Sigma_i \therefore \text{PH} = \Sigma_i$. An exactly analogous argument can be given for $L \in \Pi_i$. In either case, PH collapses to either Σ_i or Π_i for some i. □

Corollary 1.8. *PH = PSPACE $\implies$ PH collapses.*

This follows from the fact that we know a PSPACE-complete problem, namely $TQBF$.

We are now ready to prove the belief (based on PH does not collapse) that there is a language in NP with super-polynomial circuit lower bounds.

Theorem 1.9 (Karp-Lipton)**.** *NP $\subseteq$ P/poly $\implies$ PH = Σ_2.*

Proof. If we can show that $\Pi_2\text{SAT} \in \Sigma_2$, we will have $\Pi_2 \subseteq \Sigma_2$ (as it is complete for Π_2 under poly-time reductions). Then, for any $L \in \Sigma_2$, since $\overline{L} \in \Pi_2$, we have $\overline{L} \in \Sigma_2$, which implies $L \in \Pi_2$ and subsequently, then $\Sigma_2 \subseteq \Pi_2 \therefore \Sigma_2 = \Pi_2$. Then by corollary 1.5, $\text{PH} = \Sigma_2$.

Recall that $\phi \in \Pi_2\text{SAT} \iff \forall y \exists z (\phi(y, z) = 1)$. Given any fixed y, let L_y be defined by $\phi \in L_y \iff \exists z (\phi(y, z) = 1)$. It is clear to see that $(\forall y \phi \in L_y) \iff \phi \in \Pi_2\text{SAT}$. For any arbitrary y, clearly $L_y \in \text{NP}$. So, because $\text{NP} \subseteq \text{P/poly}$, we know there is some poly-size circuit family $\{C_n(y, \cdot)\}$ such that $\forall \phi \forall n$, $C_n(y, \phi) = 1 \iff \exists z \phi(y, z) = 1$.

We will use $\{C_n(y, \cdot)\}$ to build a new poly-size circuit family, $\{C'_n(y, \cdot)\}$ where $\forall \phi$, $C'_n(y, \phi) = z$ such that $\phi(y, z) = 1$, if such a z exists. Build $\{C'_n(y, \cdot)\}$ using $\{C_n(y, \cdot)\}$ based on the search-to-decision reduction (see Theorem 2.19 in Arora & Barak). To do this, we can use C_n as an 'oracle' for C'_n and compute $C_n(y, \phi(y, 0 z_2 z_3 ...))$. If this yields 1, we set $z_1 = 0$ and do the same for z_2. Else, we set $z_1 = 1$ and continue.

So, $\forall y, \phi$, we have $C'_n(y, \phi) = z \iff \exists z \phi(y, z) = 1 \therefore \phi(y, C'_n(y, \phi)) = 1 \iff \exists z \phi(y, z) = 1$. The circuit family can be guessed non-deterministically. In other words, $\forall y \exists z \phi(y, z) = 1 \iff \exists \{C'_n(y, \cdot)\} \forall y \phi(y, C'_n(y, \phi)) = 1 \iff \forall y (\phi \in L_y) \iff \phi \in \Pi_2\text{SAT}$. So, then, $\Pi_2\text{SAT} \in \Sigma_2$ and the rest follows from the argument above. □

1.1 Oracle definitions for PH

Recall the definition of $\Sigma_i SAT$, which are complete problems for Σ_i.

Definition 1.10. *For a boolean formula ϕ and i vectors of boolean variables,*
$\phi \in \Sigma_i SAT \iff \exists y_1 \forall y_2 ... y_i (\phi(y_1, y_2, ..., y_i) = 1)$.

Claim 1.11. $\forall i \geqslant 1$, $\Sigma_i = NP^{\Sigma_{i-1}SAT}$.

Proof. We prove a slightly weaker claim to exemplify this: $\Sigma_2 = \text{NP}^{\text{SAT}}$. Given some L, assume $L \in \Sigma_2$. By definition, then $\forall x, x \in L \iff \exists y \in \{0,1\}^{poly(|x|)} \forall z \in \{0,1\}^{poly(|x|)} M(x,y,z) = 1$ for some poly-time turing machine M. We show that $L \in \text{NP}^{\text{SAT}}$ by constructing a poly-time NDTM with oracle access to SAT that decides L. Given some $x \in \{0,1\}^*$. First, non-deterministically guess y. Then, given a fixed y, we define L_y by $\forall x, x \in L_y \iff \forall z \in \{0,1\}^{poly(|x|)} M(x,y,z) = 1$. Therefore $L_y \in \Pi_1 \therefore L_y \leqslant_P \overline{SAT}$. So, we poly-time transform x to a corresponding SAT query, use the oracle, and flip the answer to determine whether $x \in L_y$. If for any non-deterministically guessed y, we get $x \in L_y$, then $x \in L$. Therefore, this NDTM decides L and so $L \in \text{NP}^{\text{SAT}} \therefore \Sigma_2 \subseteq \text{NP}^{\text{SAT}}$.

Given some L, assume $L \in \text{NP}^{\text{SAT}}$. Then there is some NDTM M with oracle access to SAT that decides L. In other words, $\forall x, x \in L \iff \exists y \in \{0,1\}^{poly(|x|)} V^{\text{SAT}}(x,y) = 1$ for a poly-time verifier V that has a SAT oracle. In the execution of V, it will make some series of oracle queries $\phi_1, \phi_2, ...\phi_m$, each accompanied with a corresponding answer $a_1, a_2, ..., a_m$. For each ϕ_i, if $a_i = 1$, then we know $\exists u_i$ such that $\phi_i(u_i) = 1$ and if $a_i = 0$, then for all possible assignments v_i, $\phi_i(v_i) = 0$. For any x, if M accepts it, then there is some series of non-deterministic choices y and SAT queries with correct answers that M used. We can guess a $y \in \{0,1\}^{poly(|x|)}$, each query ϕ_i, each answer a_i, and each assignment u_i such that M accepts x using certificate (or series of non-deterministic choices) y and $\forall v_k \forall i \in [1..m](a_i = 0 \wedge \phi_i(v_k) = 0) \vee (a_i = 1 \wedge \phi_i(u_i) = 1)$. In other words, for any x, we have

$$
\begin{aligned}
x \in L \iff & \exists y \in \{0,1\}^{poly(|x|)}, \phi_1, ..., \phi_m, \\
& a_1, ..., a_m, u_1, ..., u_m \\
& \forall i \in [1..k], v, \\
& M \text{ accepts } x \text{ using the guessed answers} \wedge (a_i = 0 \wedge \phi_i(v) = 0) \vee (a_i = 1 \wedge \phi_i(u_i) = 1)
\end{aligned}
$$

Checking whether M accepts, once we have guessed non-deterministic transition function choices and queries/answers, can be done in poly-time with some small blow-up due to simulating M, and similarly verifying $\phi_i(u)$ for any assignment u can be done in polynomial time. There can only be polynomially-many queries ϕ_i, because M runs in non-deterministic polynomial time. So, $L \in \Sigma_2$. □