University of California, Los Angeles
CS 281 Computability and Complexity

Instructor: Alexander Sherstov
Scribe: Daniel Israel
Date: February 14, 2024

LECTURE
11

Completeness in the Polynomial Hierarchy

In the last lecture, we prove that totally quantified Boolean formulas (TQBF) are PSPACE-complete. TQBF allows a polynomial number of quantifiers, and toward the end of lecture, we consider the implications of restricting the number of quantifiers to a constant. It turns out fixing the number of quantifiers to a constant $k = 1, 2, 3, ...$ reveals a rich structure of complexity classes, where classes corresponding to larger k subsume the preceding classes (these containments are believed to be proper). The lecture ends by defining the Polynomial Hierarchy as the union of these classes.

In this lecture, we continue our discussion of the Polynomial Hierarchy, consider conditions under which it "collapses", and prove which problems are complete with respect to a particular level of the hierarchy. In particular, to show these results we build on top off the completeness of SAT and make good use of Cook-Levin. Many results also have dual "co" class result that can be proven in a very straightforward manner with understanding of the "co" operator. Finally, we show that each level of the Polynomial Hierarchy can be characterized using oracles, and oracles in general present a powerful method to build upon existing complexity classes.

11.1 Polynomial Hierarchy

We first recall the definition of the Polynomial Hierarchy.

Definition 11.1. The Polynomial Hierarchy (PH) is given by

$$\mathrm{PH} = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup ... = \Pi_1 \cup \Pi_2 \cup \Pi_3 \cup ... \tag{11.1}$$

where we define

$$\Sigma_i = \{\text{Languages } L \text{ such that there exists a polytime machine } M \text{ where} \tag{11.2}$$
$$x \in L \iff \exists u_1 \in \{0,1\}^{|x|^c}, \forall u_2 \in \{0,1\}^{|x|^c}, ..., M(x, u_1, ..., u_k) = 1\} \tag{11.3}$$

and its complement

$$\Pi_i = \{\text{Languages } L \text{ such that there exists a polytime machine } M \text{ where} \tag{11.4}$$
$$x \in L \iff \forall u_1 \in \{0,1\}^{|x|^c}, \exists u_2 \in \{0,1\}^{|x|^c}, ..., M(x, u_1, ..., u_k) = 1\} \tag{11.5}$$

In the Polynomial Hierarchy we can refer to a specific level for $k \geq 0$.

Definition 11.2. The k-th level of the PH is $\Sigma_k \cup \Pi_k$.

Note that the zeroth level of the Polynomial Hierarchy coincides with P and the first level is NP∪coNP. The purpose of defining levels in this way is similar to the way we define complexity classes such as P and NP. It is generally believed that P = NP is improbable, so proofs take the form: "Algorithm A is not efficient unless P = NP". In an analogous way, we can show that under certain circumstances, the PH will collapse to a particular level. Showing P = NP is merely a specific case of a more general structure.

Theorem 11.3. *If* $\Sigma_k = \Pi_k$ *for some* k*, then* PH $= \Sigma_k$.

Proof. We already know that each level is a subset of the higher levels. We must now show that under these circumstances the higher levels will become a subset of the lower levels. More concretely, we want

$$\Sigma_l, \Pi_l \subseteq \Sigma_k = \Pi_k \text{ for } l = k, k+1, k+2, ... \tag{11.6}$$

We can show the above statement through induction. In the base case $k = l$, the above statement holds through equality. As our inductive step, assume that $\Sigma_{l-1}, \Pi_{l-1} \subseteq \Sigma_k$, and we must show that $\Sigma_l, \Pi_l \subseteq \Sigma_k$. Fix an $L \in \Sigma_l$ where

$$x \in L \iff \exists u_1 \forall u_2 ... M(x, u_1, ..., u_l) \tag{11.7}$$

The above expression has l quantifiers. Now consider

$$L' = \{(x, u_1) : \forall u_2 \exists u_3 ... M'(x, u_1, ..., u_l)\} \tag{11.8}$$

Using the definition of Π_{l-1} and our inductive hypothesis, we have $L' \in \Pi_{l-1} \subseteq \Sigma_k$. Thus, there exists a polytime TM M' such that

$$(x, u_1) \in L' \iff \exists w_1 \forall w_2 ... M'(xu_1, w_1, ..., w_k)\} \tag{11.9}$$

We plug the right hand side expression into our original expression over L and utilize M' in the following manner to obtain

$$x \in L \iff \exists u_1 \exists w_1 \forall w_2 ... M'(x, u_1, w_1, w_2, ..., w_k) \tag{11.10}$$

But $\exists u_1 \exists w_1$ can be merged into a single existential quantifier over $u_1 w_1$. We therefore have an expression over k quantifiers, so $L \in \Sigma_k$.

We have shown that $\Sigma_l \subseteq \Sigma_k$, but we must also show $\Pi_l \subseteq \Sigma_k$ to complete the proof. Fortunately, this follows immediately using the "co" operator.

$$\Pi_l = \text{co}\Sigma_l = \text{co}\Sigma_k = \Pi_k = \Sigma_k \tag{11.11}$$

Thus, our induction is complete. ∎

In the above proof, the "co" operator is shown to be a powerful tool that can save effort. We see this again in the following challenge problem.

PROBLEM 11.4. Show that if SAT $\in$ coNP, then NP = coNP

Proof. Since NP reduces in polytime to SAT and SAT $\in$ coNP, NP $\subseteq$ coNP. We then have

$$\text{coNP} \subseteq \text{cocoNP} = \text{NP} \tag{11.12}$$

∎

11.2 Completeness

In the same way that we can define completeness for NP and PSPACE, we can define completeness over the Polynomial Hierarchy.

DEFINITION 11.5. A language L is Σ_k-complete if

1. $L \in \Sigma_k$
2. $L' \leq_p L$ for all $L' \in \Sigma_k$

It is worth pondering why we use polytime reductions in all of the completeness definitions. Polynomial time reductions are the "weakest" form of reduction (except for log time). Using the "weakest" reduction will give the "finest" equivalence relation on languages (meaning it will be easiest to show that two languages are not equivalent). SAT is a prototypical NP-complete problem, and we can define a more general PH version of this problem.

DEFINITION 11.6. For quantifier Q_k

$$\Sigma_k\text{SAT} = \{\text{Boolean formula } \varphi : \exists u_1 \forall u_2 ... Q_k u_k \varphi(u_1, ..., u_k) = 1\} \tag{11.13}$$

DEFINITION 11.7.

$$\Pi_k\text{SAT} = \{\text{Boolean formula } \varphi : \forall u_1 \exists u_2 ... Q_k u_k \varphi(u_1, ..., u_k) = 1\} \tag{11.14}$$

Note that boolean formulas are more general than CNF/DNFs, which are depth 2 Boolean formulas. In fact, Boolean formulas in the presence of a quantifier and new variable can be represented as depth 2. We shall now show the completeness of these languages.

THEOREM 11.8. Σ_kSAT *is* Σ_k*-complete*

Proof. First, we will reason that $\Sigma_k\text{SAT} \in \Sigma_k$. A Turing Machine M can perform the operation of a Boolean circuit in polynomial time. Once we observe $\varphi(u_1, ..., u_k) = 1 \iff M(\varphi, u_1, ..., u_k)$, by plugging in definitions we can see $\Sigma_k\text{SAT} \in \Sigma_k$. To show hardness, we begin with some language $L' \in \Sigma_k\text{SAT}$. Then we have

$$x \in L' \iff \exists u_1 \forall u_2 ... Q_k u_k M(x, u_1, ..., u_k) = 1 \tag{11.15}$$

By Cook-Levin, we can find a formula φ_M in polytime such that

$$M(x, u_1, ..., u_k) = 1 \iff Q_{k+1}u_{k+1}\varphi_M(x, u_1, ..., u_k, u_{k+1}) \tag{11.16}$$

with $Q_{k+1} = Q_k$. We will show that an arbitrary Turing Machine can be reduced to a Boolean formula. Plugging this result into our previous result yields

$$\begin{aligned} x \in L' &\iff \exists u_1 \forall u_2 ... Q_k u_k Q_{k+1} u_{k+1} \varphi_M(x, u_1, ..., u_k, u_{k+1}) && (11.17)\\ &\iff \exists u_1 \forall u_2 ... Q_k u_k u_{k+1} \varphi_M(x, u_1, ..., u_k u_{k+1}) && (11.18)\end{aligned}$$

Thus, we have produced a reduction function that takes input x and outputs $\varphi_M(x, ...)$, where k placeholders follow x such that $\varphi_M \in \Sigma_k$SAT if and only if $x \in L$. ∎

The completeness of the complement follows in a trivial manner.

Theorem 11.9. *Π_kSAT is Π_k-complete.*

Proof. In general, if a language L is complete with respect to a class C, then its complement L' will be complete with respect to co$C$. Thus, Σ_kSAT is Σ_k-complete implies Π_kSAT is Π_k-complete. ∎

Corollary 11.10. *$\Sigma_k, \Pi_k \subseteq$ PSPACE for all k, and PH $\subseteq$ PSPACE*

Proof. Σ_kSAT and Π_kSAT are special cases of TQBF, which is PSPACE-complete. ∎

Finally, we can show that PH-completeness in general implies some collapse in the PH.

Theorem 11.11. *If some language L is PH-complete, then PH $= \Sigma_k$ for some k.*

Proof. Since $L \in \text{PH} = \Sigma_1 \cup \Sigma_2 \cup ...$, we have $L \in \Sigma_k$ for some k. But then every $L' \in$ PH reduces deterministically in polytime to L, meaning PH $\subseteq \Sigma_k$. Since it is known that $\Sigma_k \subseteq$ PH, we can conclude PH $= \Sigma_k$. ∎

11.3 Characterization of PH with Oracles

The Polynomial Hierarchy can be characterized very succintly with oracle machines. We can demonstrate this characterization specifically on the second level of the hierarchy.

Theorem 11.12. $\text{NP}^{\text{NP}} = \Sigma_2$

Proof. Will first demonstrate that $\Sigma_2 \subseteq \text{NP}^{\text{SAT}}$. Fix an $L \in \Sigma_2$, then

$$x \in L \iff \exists u_1 \forall u_2 M(x, u_1, u_2) \iff \exists u_1 \neg \underbrace{(\exists u_2 M(x, u_1, u_2))}_{\text{SAT query}} \tag{11.19}$$

By Cook-Levin, $\exists u_2 M(x, u_1, u_2)$ can be reduced to SAT, so $L \in \text{NP}^{\text{SAT}}$.

We must now show the reverse direction: $\text{NP}^{\text{SAT}} \subseteq \Sigma_2$. Consider a NTM with a SAT oracle M^{SAT} that decides L (see Figure 11.1). As depicted visually, M^{SAT} can make polynomially many queries to the SAT oracle in which future queries depend on past queries.

The core idea of the reduction is nondeterministically guess **all** answers and then verify them with a single coNP query. [1]

Observe that $M^{\mathrm{SAT}}(x) = 1$ if and only if there exists a nondeterministic guess u, SAT queries $\varphi_1, ..., \varphi_{|x|^c}$, and answers $a_1, ..., a_{|x|^c}$ such that for all i where φ_i is the query posted by M on input x, nondeterministic guess u, and oracle answers $a_1, ..., a_{i-1}$ the following hold:

- $a_i = 1 \implies \varphi_i \in \mathrm{SAT}$
- $a_i = 0 \implies \varphi_i \notin \mathrm{SAT}$

Observe that the previous two conditions can be phrased as computations with NP and coNP oracles. That is, we can rewrite them as follows

- $a_i = 0 \vee \exists w_i \varphi_i(w_i) = 1$
- $a_i = 1 \vee \forall z_i \varphi_i(z_i) = 0$

We are able to pull the quantifiers out because the expressions are monotone. Thus, our reduction is complete. We conclude that $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2$. $\square$

While $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2$ is a specific relationship between oracle machines and the Polynomial Hierarchy, we leave it as an exercise to the reader to discover the full structure relating these two objects.

Problem 11.13. Generalize the previous result to $\mathrm{coNP}^{\mathrm{coNP}}, \mathrm{coNP}^{\mathrm{NP}}, \mathrm{NP}^{\mathrm{coNP}}, \mathrm{NP}^{\mathrm{NP}}$. What about $\mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}^{\cdots}}}$ where NP is repeated k times?

References

[1] S. Arora and B. Barak. *Computational complexity: a modern approach.* Cambridge University Press, 2009.

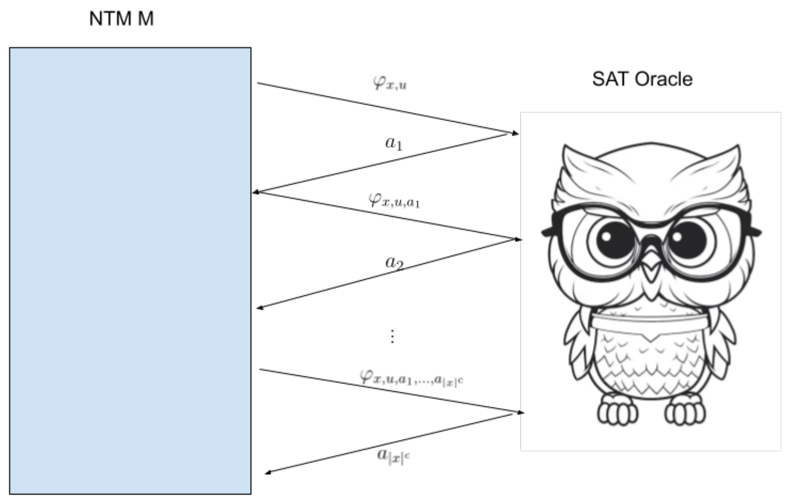

FIGURE 11.1: Visualization of Nondeterministic Turing Machine with a SAT Oracle