COM S 6810 Theory of Computing February 3, 2009

Lecture 5: Polynomial Hierarchy

Instructor: Rafael Pass *Scribe: Navin Sivakumar*

Recall that NP can be thought of as the class of languages consisting of strings for which there exists an easily verifiable proof. Similarly, the complementary class coNP can be interpreted as the class of languages consisting of strings for which all proofs fail. Intuitively, problems in NP ask whether there *exists* a string satisfying certain properties, whereas problems in coNP whether *all* strings satisfy certain properties. A natural extension is to consider problems which combine existential and universal quantifiers. The complexity classes which emerge from this process make up the *polynomial hierarchy.*

In order to define the complexity classes formally, it is useful to introduce the notion of a polynomial time relation:

Definition 1 *A relation $R(x, y_1, \ldots, y_n)$ is a* polynomial time relation *if there is a (deterministic) Turing machine V and polynomial p such that V decides $R(x, y_1, \ldots, y_n)$ in time $p(|x|)$.*

We then define NP as follows:

Definition 2 *A language $\mathcal{L}$ is in the class NP if and only if there exists a polynomial time relation R such that $x \in \mathcal{L}$ if and only if $\exists y$ such that $R(x, y)$ holds.*

We can also define the complementary class coNP:

Definition 3 *A language $\mathcal{L}$ is in the class coNP if and only if its complement $\overline{\mathcal{L}}$ is in NP; equivalently, a language $\mathcal{L}$ is in coNP if and only if there exists a polynomial time relation R such that $x \in \mathcal{L}$ if and only if $\forall y$, $R(x, y)$ holds.*[1]

The following example uses the languages SAT and coSAT to illustrate the definitions given above:

Example 1 *To see that SAT $\in$ NP, we interpret the string x as representing a formula and the string y as representing an assignment to the variables in the formula; $R(x, y)$ holds if and only if the assignment given by y satisfies the formula given by x. To see that coSAT $\in$ coNP, we interpret x and y as before and define $R(x, y)$ to hold if and only if the assignment given by y does not satisfy x.*

[1] To see that the definitions are equivalent, consider the complementary relations R and $\overline{R}$ such that $R(x, y)$ holds if and only if $\overline{R}(x, y)$ does not hold; if $\overline{R}$ is the relation demonstrating that $\overline{\mathcal{L}}$ is in NP, then R shows that $\mathcal{L}$ is in coNP, and vice versa.

Defining NP and coNP in this way highlights the role of the existential and universal quantifiers. The following example demonstrates how we might build more complicated questions by combining different quantifiers:

Example 2 *Consider the problem which gives a formula ϕ and integer k and asks whether there exists a formula ϕ' which is equivalent to ϕ such that $|\phi'| \le k$. This can be written in terms of a polynomial-time relation as follows: define relation $R((\phi, k), \phi', y)$ to hold if and only if $|\phi'| \le k$ and $phi(y) = \phi'(y)$, where y is interpreted as an assignment to the variables in the formulas. Then the problem defines a language which contains (ϕ, k) if and only if $\exists \phi'$ such that $\forall y$, $R((\phi, k), \phi', y)$ holds.*

Generalizing the pattern allows us to define the classes Σ_i and Π_i of the polynomial hierarchy:

Definition 4 *A language $\mathcal{L}$ is in Σ_i if and only if $\exists$ a polynomial-time relation R such that $x \in \mathcal{L}$ if and only if*

$$\exists y_1 \forall y_2 \cdots Q_i y_i R(x, y_1, y_2, \ldots, y_i),$$

where $Q_j = \forall$ if j is even and $Q_j = \exists$ if j is odd.

A language $\mathcal{L}$ is in Π_i if and only if $\exists$ a polynomial-time relation R such that $x \in \mathcal{L}$ if and only if

$$\forall y_1 \exists y_2 \cdots Q_i y_i R(x, y_1, y_2, \ldots, y_i),$$

where $Q_j = \exists$ if j is even and $Q_j = \forall$ if j is odd. Equivalently, $\mathcal{L}$ is in Π_i if and only if $\overline{\mathcal{L}}$ is in Σ_i (i.e. $\Pi_i = \mathsf{co}\Sigma_i$).

Define the class PH by $\mathsf{PH} = \bigcup_i \Sigma_i$.

Note that it follows directly from the definitions that $\Sigma_1 = \mathsf{NP}$ and $\Pi_1 = \mathsf{coNP}$. It is also straightforward to see that for all i, the following hold:

$$\begin{aligned} \Sigma_i &\subseteq \Sigma_{i+1} \\ \Pi_i &\subseteq \Pi_{i+1} \\ \Sigma_i &\subseteq \Pi_{i+1} \\ \Pi_i &\subseteq \Sigma_{i+1}. \end{aligned}$$

Besides being natural extensions of the NP and coNP, the classes of the polynomial hierarchy can be interpreted in the context of games. In this setting, languages in Σ_i can be thought of asking whether there exists a winning strategy in $\frac{i}{2}$ rounds for the first player in a game. To see this, we can interpret the quantifiers by asking whether there exists a move y_1 such that no matter what response y_2 is played, there exists a move y_3, and so on for $\frac{i}{2}$ rounds, such that player 1 wins. Similarly, languages in Π_i can be interpreted as asking about winning strategies for the second player.

As we have seen with NP and NL, it is often convenient when reasoning about complexity classes to make use of problems which are complete for the class. We can construct Σ_i-complete problems by generalizing SAT (which is NP-complete and therefore Σ_1-complete):

Definition 5 *Define the language Σ_i-SAT to be the set of formulas ϕ such that*

$$\exists y_1 \forall y_2 \cdots Q_i y_i, \phi(y_1, \dots, y_i),$$

where $Q_j = \exists$ if j is odd and $Q_j = \forall$ if j is even.

Lemma 1 *For each i, the language Σ_i-SAT is Σ_i-complete.*

Alternatively, we can define the classes Σ_i (and similarly Π_i recursively by

$$\Sigma_i = \mathsf{NP}^{\Sigma_{i-1}} = \mathsf{NP}^{\Sigma_{i-1}\text{-}\mathsf{SAT}},$$

where $\mathsf{NP}^{\Sigma_{i-1}}$ denotes the class of languages decidable by a non-deterministic Turing machine in polynomial time given access to an oracle which decides languages in Σ_{i-1}. This recursive definition can be easier to work with in some cases. The following theorem establishes that both definitions define the same complexity classes[2]:

Theorem 2 *Define Σ_i as in Definition 5. Then $\Sigma_i = \mathsf{NP}^{\Sigma_{i-1}\text{-}\mathsf{SAT}}$.*

Proof. We prove the special case $\Sigma_2 = \mathsf{NP}^{\mathsf{SAT}}$. The result follows from an induction argument where the proof of the induction step is essentially identical to the special case with some extra notation.

First, we will show that $\Sigma_2 \subseteq \mathsf{NP}^{\mathsf{SAT}}$. Let $\mathcal{L}$ be in Σ_2. Then there exists a polynomial-time relation R such that $x \in \mathcal{L}$ iff $\exists y_1$ such that $\forall y_2$, $R(x, y_1, y_2)$. Intuitively, the approach is to non-deterministically guess y_1; then we can use the oracle to decide if $\exists y_2$ such that the complement of R holds on (x, y_1, y_2) and negate the answer. More formally, consider the following non-deterministic oracle machine M operating on input x:

- Make a non-deterministic guess y_1.
- Use a Karp reduction to write write $\overline{R}(x, y_1, y_2)$ as a SAT formula $\phi(y_2)$ with x and y_1 hard-coded

[2]Note that the second equality follows directly from the fact that Σ_{i-1}-SAT is Σ_{i-1}-complete.

- Feed ϕ to the oracle and accept if and only if the oracle rejects.

It follows that $M(x)$ accepts if and only if $\exists y_1$ such that $\forall y_2$, y_2 does not satisfy ϕ, or, equivalently, $R(x, y_1, y_2)$ holds.

Now we will show that $\mathsf{NP}^{\mathsf{SAT}} \subseteq \Sigma_2$. Let $\mathcal{L} \in \mathsf{NP}^{\mathsf{SAT}}$. Intuitively, if $\mathcal{L}$ can be decided by a polynomial time non-deterministic oracle machine M, then we can say something like, "There exist non-deterministic choices y, oracle queries $q_1, \ldots, q_k$, and oracle answers $a_1, \ldots, a_k$ such that M accepts x in polynomial time." However, this intuition suggests that $\mathsf{NP}^{\mathsf{SAT}} \subseteq \mathsf{NP}$; we have seen in an earlier lecture that this would imply $\mathsf{NP} = \mathsf{coNP}$, which would be a remarkable result. The flaw in this reasoning is that we must include a condition requiring that the oracle answers $a_1, \ldots, a_k$ are valid answers to the queries $q_1, \ldots, q_k$, i.e. $a_j = 1$ if and only if $q_j \in \mathsf{SAT}$. Therefore, we can describe $\mathcal{L}$ by observing that $x \in \mathcal{L}$ if and only if $\exists y, q_1, \ldots, q_k, a_1, \ldots, a_k$ such that M accepts x and $a_j = 1$ if and only if $q_j \in \mathsf{SAT}$. However, deciding the relation "M accepts x and $a_j = 1$ if and only if $q_j \in \mathsf{SAT}$" requires deciding SAT, so we would like to rewrite this characterization in terms of a polynomial time relation. We do this by observing that $q_j \in \mathsf{SAT}$ if and only if $\exists x_j^Y$ such that $q_j(x_j^Y) = 1$, and $q_j \notin \mathsf{SAT}$ if and only if $\forall x_j^N$, $q_j(x_j^N) = 0$. Thus, we can rewrite the characterization of $\mathcal{L}$ as

$$x \in \mathcal{L} \text{ iff } \exists y, q_1, \ldots, q_k, a_1, \ldots, a_k, x_1^Y, \ldots, x_k^Y \text{ such that } \forall x_1^N, \ldots, x_k^N$$

- M accepts x
- If $a_j = 1$, then $q_j(x_j^Y) = 1$
- If $a_j = 0$, then $q_j(x_j^N) = 0$

■

An interesting property of the polynomial hierarchy is that if any two classes in the hierarchy are equal, then the hierarchy "collapses." The following theorem makes this statement precise:

Theorem 3 *If $\Sigma_i = \Pi_i$, then $\mathsf{PH} = \Sigma_i$.*

Proof. We prove the following special case: if $\mathsf{NP} = \mathsf{coNP}$, then $\mathsf{PH} = \mathsf{NP}$. As for the proof of theorem 2, the proof of the general case is analogous but involves more cumbersome notation.

It is trivial that $\mathsf{NP} \subseteq \mathsf{PH}$. In order to show that $\mathsf{PH} \subseteq \mathsf{NP}$, we will show inductively that $\Sigma_i \subseteq \mathsf{NP}$ for all i. For $i = 1$, $\Sigma_1 = \mathsf{NP}$ by definition. Now assume that $\Sigma_i \subseteq \mathsf{NP}$. Recall from theorem 2 that $\Sigma_{i+1} = \mathsf{NP}^{\Sigma_i}$. By the inductive hypothesis, $Sigma_i = \mathsf{NP}$. Therefore, $\Sigma_{i+1} = \mathsf{NP}^{\mathsf{NP}} = \mathsf{NP}^{\mathsf{SAT}}$. Thus, it is sufficient to show that $\mathsf{NP}^{\mathsf{SAT}} \subseteq \mathsf{NP}$

under the assumption that $\mathsf{NP} = \mathsf{coNP}$. Let $\mathcal{L}$ be in $\mathsf{NP}^{\mathsf{SAT}}$. Then there exists a non-deterministic oracle machine M which decides $\mathcal{L}$. As in the proof of theorem 2, $\mathcal{L}$ can be characterized as follows: $x \in \mathcal{L}$ if and only if there exist non-deterministic choices y, oracle queries $q_1, \ldots, q_k$ and oracle answers $a_1, \ldots, a_k$ such that M accepts x and a_j is a valid answer to query q_j for each j. Again, we must convert the process of checking the validity of oracle answers into a polynomial-time relation. It is straightforward to formulate checking "Yes" oracle answers as a problem in NP; under the assumption that $\mathsf{NP} = \mathsf{coNP}$, we can apply a reduction from coSAT to SAT to verify "No" answers from the oracle. Formally, define the relation R by $R(x, y, q_1, \ldots, q_k, a_1, \ldots, a_k, x_1, \ldots, x_k)$ by the following procedure:

- Check if M accepts x using the non-deterministic choices y, oracle queries $q_1, \ldots, q_k$, and oracle answers $a_1, \ldots, a_k$; otherwise reject
- If $a_j = 1$, check that $q_j(x_j) = 1$; otherwise reject
- If $a_j = 0$, compute the Karp reduction from coSAT to SAT on q_j to find a formula q_j'. Check that $q_j'(x_j) = 1$; otherwise reject

Clearly the relation is polynomial time checkable. It is easy to check that $x \in \mathcal{L}$ if and only if

$$\exists y, q_1, \ldots, q_k, a_1, \ldots, a_k, x_1, \ldots, x_k \text{ such that } R(x, y, q_1, \ldots, q_k, a_1, \ldots, a_k, x_1, \ldots, x_k).$$

Therefore, $\mathcal{L} \in \mathsf{NP}$. ■

Much like the assumption $\mathsf{P} \neq \mathsf{NP}$, results in complexity are sometimes proved under the assumption that PH does not collapse. Observe that this is a stronger assumption than $\mathsf{P} \neq \mathsf{NP}$, since $\mathsf{P} \neq \mathsf{NP}$ implies $\mathsf{NP} = \mathsf{coNP}$.

We would also like to consider how PH relates to space-complexity classes such as PSPACE. It is straightforward to verify the following theorem:

Theorem 4 *PH $\subseteq$ PSPACE.*

It remains an open question whether the containment is proper. The relationship between PSPACE and PH is illustrated by the PSPACE-complete language of true quantified boolean formulas, or TQBF:

Definition 6 *A formula ϕ is in TQBF if and only if $\exists y_1 \forall y_2 \cdots Q_n y_n \phi(y_1, \ldots, y_n)$.*

Observe that TQBF has a form very similar to languages in PH; however, the number of quantifiers is allowed to depend on the input length n. Note also that if $\mathsf{PH} = \mathsf{PSPACE}$, then the polynomial hierarchy collapses, since the complete language TQBF would fall in Σ_i for some i.