PRINCETON UNIVERSITY COS 522: COMPUTATIONAL COMPLEXITY

Lecture 4: Polynomial Hierarchy

Lecturer: *Sanjeev Arora* Scribe: *self*

The polynomial hierarchy is a hierarchy of complexity classes that generalizes **P**, **NP**, and **coNP**. Recall the definition of **NP**, **coNP**.

DEFINITION 1 *Language L is in* **NP** *if there is a language $L_0 \in \mathbf{P}$ and constants $c, d > 0$ such that*

$$\forall x \in \{0,1\}^* \quad x \in L \iff \quad \exists y \in \{0,1\}^*, |y| \le d\,|x|^c \quad \text{and} \quad (x,y) \in L_0.$$

Language L in in **coNP** *iff $\overline{L} \in \mathbf{NP}$. In other words, there is a language $L_0 \in \mathbf{P}$ and constants $c, d > 0$ such that*

$$\forall x \in \{0,1\}^* \quad x \in L \iff \quad \forall y \in \{0,1\}^*, |y| \le d\,|x|^c \quad \text{and} \quad (x,y) \in L_0.$$

DEFINITION 2 (POLYNOMIAL HIERARCHY) *The polynomial hierarchy is defined as $\cup_{i \ge 0} \mathbf{\Sigma}_i^p$ or equivalently $\cup_{i \ge 0} \mathbf{\Pi}_i^p$, where*

1. $\mathbf{\Sigma}_0^p = \mathbf{\Pi}_0^p = \mathbf{P}$.
2. $\mathbf{\Sigma}_1^p = \mathbf{NP}$, $\mathbf{\Pi}_1^p = \mathbf{coNP}$.
3. $\mathbf{\Sigma}_i^p$ *consists of any language for which there is a language $L_0 \in \mathbf{\Pi}_{i-1}^p$ such that*

$$\forall x \in \{0,1\}^* \quad x \in L \iff \quad \forall y \in \{0,1\}^*, |y| \le d\,|x|^c \quad \text{and} \quad (x,y) \in L_0.$$

4. $\mathbf{\Pi}_i^p$ *consists of any language for which there is a language $L_0 \in \mathbf{\Sigma}_{i-1}^p$ such that*

$$\forall x \in \{0,1\}^* \quad x \in L \iff \quad \exists y \in \{0,1\}^*, |y| \le d\,|x|^c \quad \text{and} \quad (x,y) \in L_0.$$

EXAMPLE 1 To understand this definition, let us unwrap it for $i = 2$. Language L is in $\mathbf{\Sigma}_2^p$ if there is a language $L_0 \in \mathbf{P}$ and constants $c, d > 0$ such that a string x is in L iff

$$\exists y_1 \ \le d\,|x|^c \ \forall y_2 \le d\,|x|^c \quad (x, y_1, y_2) \in L_0.$$

Similarly Language L is in $\mathbf{\Pi}_2^p$ if there is a language $L_0 \in \mathbf{P}$ and constants $c, d > 0$ such that a string x is in L iff

$$\forall y_1 \ \le d\,|x|^c \ \exists y_2 \le d\,|x|^c \quad (x, y_1, y_2) \in L_0.$$

Clearly, $L \in \mathbf{\Pi}_2^p$ iff $\overline{L} \in \mathbf{\Sigma}_i^p$.

Similarly we can unwrap the definition for general i and directly define $\mathbf{\Sigma}_i^p$ using i quantifiers, the first being $\exists$ and the rest alternating between $\exists$ and $\forall$. The class $\mathbf{\Pi}_i^p$ involves involves i quantifiers, alternating between $\exists$ and $\forall$ and beginning with $\forall$.

What are some natural problems in these classes? Consider the language EXACT-TSP, defined as

$$\{< G, C >: \ G \text{ is a weighted graph and its shortest salesman tour has length } C\}. \tag{1}$$

$$= \{< G, C >: \ \exists \text{ salesman tour } \pi \text{ s.t. } \mathrm{cost}(\pi) = C \text{ and } \forall \text{ tour } \pi' \ \mathrm{cost}(\pi') \geq C\} \tag{2}$$

Then EXACT-TSP $\in \mathbf{\Sigma}_2^p$.

Now we describe a language MIN-CNF in $\mathbf{\Pi}_2^p$; this language is of interest in electrical engineering, specifically, circuit minimization. We say that two boolean formulae are *equivalent* if they have the same set of satisfying assignments.

$$\text{MIN-CNF} = \{< \varphi >: \ \varphi \text{ is not equivalent to any smaller formula}\}. \tag{3}$$

$$= \{< \varphi >: \ \forall \psi, |\psi| < |\varphi|, \exists \text{ assignment } s \text{ such that } \varphi(s) \neq \psi(s)\}. \tag{4}$$

The class Σ_i^p has a complete problem involving quantified boolean formulae[1] with limited number of alternations. Specifically, it is

$$\Sigma_i\text{-SAT} = \exists \vec{x}_1 \forall \vec{x}_2 \exists \cdots Q \vec{x}_i \ \varphi(\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_i) = 1, \tag{5}$$

where φ is a boolean formula, each $\vec{x}_i$ is a vector of boolean variables, and Q is $\exists$ or $\forall$ depending on whether i is odd or even.

The next simple theorem is the only theorem in this lecture.

Theorem 1
If $\mathbf{P} = \mathbf{NP}$ *then* $\mathbf{PH} = \mathbf{P}$.

Proof: Easy. $\square$

We strongly believe that not only is $\mathbf{P} \neq \mathbf{NP}$ but also that all levels of $\mathbf{PH}$ are distinct. This latter conjecture will be useful in the rest of the course; we will reduce other conjectures to it (that is to say, prove other conjectures assuming this is true).

1 Alternating Turing machines

Alternating TMs are like nondeterministic TMs except the states are labelled with either $\exists$ or $\forall$. A nondeterministic TM is a special case in which states are labelled with only $\exists$. The acceptance criterion for such a machine is defined in the obvious way by looking at the tree of all possible computation branches, and propagating the YES/NO decisions at the leaves to the root using the

[1]The resemblance to TQBF is not coincidental. In the definition of $\mathbf{\Sigma}_i^p$ if we allow the number of alternations i to be polynomial in the input, then we get a class called AP, which is exactly PSPACE, and hence has TQBF as its complete problem. Verify this!

obvious semantics for $\exists$ and $\forall$. Then $\mathbf{\Sigma}_i^p$ is the class of languages accepted by polynomial time alternating TMs in which each computation branches features at most $(i-1)$ alternations between $\exists$ and $\forall$, and the machine starts in an $\exists$ state. The class $\mathbf{\Pi}_i^p$ is similarly defined except the machine starts in a $\forall$ state.

It is easily checked that this definition is equivalent to our earlier definitions. We note that the levels of the polynomial hierarchy can also be defined using oracle turing machines. This is explored in the exercises.

Problems

§1 Show that the language in (5) is complete for $\mathbf{\Sigma}_i^p$ under polynomial time reductions. (Hint use the **NP**-completeness of SAT.)

§2 Prove Theorem 1.

§3 Prove that $\mathbf{AP} = \mathbf{PSPACE}$, as claimed in the footnote.

§4 Suppose we define logspace computation.

§5 Show that $\mathbf{\Sigma}_2^p = \mathbf{NP}^{\mathrm{SAT}}$.