Exercises on the Polynomial Hierarchy PH
CSCI 6114 Fall 2021

Joshua A. Grochow

September 2, 2021

Definition 1. We use $\exists^p, \forall^p$ to denote the polynomially-bounded version of these quantifiers.

For example, we can (re)define NP as the class of languages L such that there is a polynomial-time verifier V, and for all x,

$$\begin{aligned} x \in L &\iff (\exists^p y)[V(x,y) = 1] \\ &\iff (\exists y)[|y| \leq \text{poly}(|x|) \text{ and } V(x,y) = 1] \end{aligned}$$

Definition 2.
1. A language L is in $\Sigma_{\mathsf{k}}\mathsf{P}$ ($k \geq 0$) if there is a polynomial-time verifier V such that, for all x,

$$x \in L \iff (\exists^p y_1)(\forall^p y_2) \cdots (\exists^p/\forall^p y_k) V(x, y_1, y_2, \ldots, y_k) = 1.$$

   where the final quantifier is $\exists^p$ if k is odd and $\forall^p$ if k is even.

2. We similarly define $\Pi_{\mathsf{k}}\mathsf{P}$ except where the right-hand side starts with $\forall^p y_1$ (and then alternate).

3. Finally, we define $\mathsf{PH} = \bigcup_{k \geq 0} \Sigma_{\mathsf{k}}\mathsf{P}$.

Exercises

1. Show that $\mathsf{P} = \Sigma_0\mathsf{P} = \Pi_0\mathsf{P}$ and $\mathsf{NP} = \Sigma_1\mathsf{P}$.

2. (a) Show that $\mathsf{PH} \subseteq \mathsf{EXP}$, where EXP is the class of decision problems that can be decided by a Turing machine that runs in time $2^{\text{poly}(n)}$.

(b) Show that $\mathsf{PH} \subseteq \mathsf{PSPACE}$, where PSPACE is the class of decision problems that can be decided by a Turing machine that uses an amount of *space* that is poly(n) (with no *a priori* upper bound on its runtime).

3. Show that $\Sigma_k\mathsf{P} = \mathsf{co}\Pi_k\mathsf{P}$. That is, $L \in \Sigma_k\mathsf{P}$ iff $\overline{L} \in \Pi_k\mathsf{P}$ ($\overline{L}$ is our notation for the complement language, $\overline{L} := \Sigma^* \backslash L = \{x \in \Sigma^* | x \notin L\}$). If this feels too abstract, start with $k = 1$.

4. Is $\mathsf{NP} = \mathsf{coNP}$? This is a hard problem. Try to convince each other one way or the other.

5. Show that $\Sigma_k\mathsf{P} \subseteq \Sigma_{k+1}\mathsf{P} \cap \Pi_{k+1}\mathsf{P}$. Conclude that (a) $\Sigma_k\mathsf{P} \cup \Pi_k\mathsf{P} \subseteq \Sigma_{k+1}\mathsf{P} \cap \Pi_{k+1}\mathsf{P}$, (b) $\mathsf{PH} = \bigcup_{k\geq 0} \Pi_k\mathsf{P}$.

6. (a) Show that a language L is in NP iff there exists a poly-time verifier V such that for all x,

$$x \in L \iff (\exists^p y_1)(\exists^p y_2)V(x, y_1, y_2) = 1.$$

(b) Show that it is only the number of quantifier *alternations* that matter, and not the total number of quantifiers in the definition of $\Sigma_k\mathsf{P}$. More specifically, if in the definition of $\Sigma_k\mathsf{P}$ we allow a block of $\exists^p$ quantifier or a block of $\forall^p$ quantifiers in place of any one of the $\exists^p/\forall^p$ quantifiers in the definition above, we get back the same class.

Definition 3. If $\mathsf{PH} = \Sigma_k\mathsf{P}$ for some fixed k, we say that PH *collapses* (to the k-th level), and otherwise that PH is *infinite.* (Note the latter is a slight abuse of terminology since PH always contains infinitely many langauges.)

7. (a) Show that if there exists $k \geq 0$ such that $\Sigma_k\mathsf{P} = \Pi_k\mathsf{P}$ then $\mathsf{PH} = \Sigma_k\mathsf{P}$. *Hint:* Use the previous problem.

(b) Show that if there exists a $k \geq 0$ such that $\Sigma_k\mathsf{P} = \Sigma_{k+1}\mathsf{P}$, then $\mathsf{PH} = \Sigma_k\mathsf{P}$.

(c) Show that if PH has a complete problem, then PH collapses.

8. We define the decision problem $\Sigma_k CIRCUIT\text{-}SAT$ as follows:

Σ_k **CIRCUIT-SAT**

Input: A Boolean circuit $\varphi(x_1, \ldots, x_m)$, together with a partition of $\{1, \ldots, m\}$ into k subsets $S_1, \ldots, S_k$.
Decide: It is the case that $\exists \vec{y} \forall \vec{z} \cdots (\exists/\forall \vec{w}) \varphi(\vec{y}, \vec{z}, \ldots, \vec{w}) = 1$, where $\vec{y} = \vec{x}|_{S_1}, \vec{z} = \vec{x}|_{S_2}, \ldots, \vec{w} = \vec{x}|_{S_k}$, and the final quantifier is $\exists$ if k is odd and $\forall$ if k is even.

Note 1: these are *not* "$\exists^p$"-style quantifiers, and that each vector $\vec{y}, \vec{z}, \ldots, \vec{w}$ is a vector of Boolean variables. The decision problem is to decide whether the quantified mathematical statement is true or false (note: the question is *not* satisfiable vs unsatisfiable, since all variables are quantified, but literally a true statement or a false statement).

Note 2: CIRCUIT-SAT is the same as $\Sigma_1 CIRCUIT\text{-}SAT$. (That is, satisfiable unquantified circuits are in essence the same as true statements that are $\exists$-quantified circuits.)

Question. Show that for any $k \geq 1$, $\Sigma_k CIRCUIT\text{-}SAT$ is $\mathsf{\Sigma_k P}$-complete. (It's also true for $k = 0$, but for somewhat trivial reasons.) *Hint:* Use the idea of the proof that $\mathsf{P} \subseteq \mathsf{P/poly}$ from the first set of exercises.

(Foreshadowing: when we get to PSPACE, we will see that a related problem, Totally Quantified Boolean Formulas, or TQBF, is PSPACE-complete. TQBF is just like $\Sigma_k CIRCUIT\text{-}SAT$ except that there is no limit placed on how many quantifier alternations there can be.)

Resources

- Defined in Stockmeyer, *Theoret. Comp. Sci.*, 1976
- Arora & Barak Ch. 5
- Du & Ko Ch. 3
- Schöning & Pruim, *Gems of TCS*, Ch. 16
- Hemaspaandra & Ogihara, *Complexity Theory Companion*, Appendix A.4.1
- Homer & Selman §7.4 do PH in terms of oracles; we'll see that characterization later, so I'm including it here for future reference, but we haven't gotten to it yet.