Computational Complexity: A Modern Approach

Draft of a book: Dated January 2007

Comments welcome!

Sanjeev Arora and Boaz Barak
Princeton University
complexitybook@gmail.com

This is an Internet draft. Some chapters are more finished than others. References and attributions are very preliminary and we apologize in advance for any omissions (but hope you will nevertheless point them out to us).

Please send us bugs, typos, missing references or general comments to complexitybook@gmail.com — **Thank You!!**

Chapter 5

The Polynomial Hierarchy and Alternations

> *"..synthesizing circuits is exceedingly difficulty. It is even more difficult to show that a circuit found in this way is the* most *economical one to realize a function. The difficulty springs from the large number of essentially different networks available."*
> Claude Shannon 1949

This chapter discusses the *polynomial hierarchy*, a generalization of **P**, **NP** and **coNP** that tends to crop up in many complexity theoretic investigations (including several chapters of this book). We will provide three equivalent definitions for the polynomial hierarchy, using quantified predicates, alternating Turing machines, and oracle TMs (a fourth definition, using uniform families of circuits, will be given in Chapter 6). We also use the hierarchy to show that solving the SAT problem requires either linear space or super-linear time.

5.1 The classes $\mathbf{\Sigma}_2^p$ and $\mathbf{\Pi}_2^p$

To understand the need for going beyond nondeterminism, let's recall an **NP** problem, INDSET, for which we *do* have a short certificate of membership:

$$\mathsf{INDSET} = \{\langle G, k\rangle : \text{ graph } G \text{ has an independent set of size } \geq k\}\,.$$

Consider a slight modification to the above problem, namely, determining the largest independent set in a graph (phrased as a decision problem):

$$\mathsf{EXACT\,INDSET} = \{\langle G, k\rangle : \text{ the largest independent set in } G \text{ has size exactly } k\}\,.$$

Now there seems to be no short certificate for membership: $\langle G, k\rangle \in \mathsf{EXACT\,INDSET}$ iff *there exists* an independent set of size k in G and *every other* independent set has size at most k.

Similarly, consider the language MIN-DNF, the decision version of a problem in circuit minimization, a topic of interest in electrical engineering (and referred to in Shannon's paper). We say that two boolean formulae are *equivalent* if they have the same set of satisfying assignments.

$$\begin{aligned}\mathsf{MIN-DNF} &= \{\, \llcorner\varphi\lrcorner :\ \varphi \text{ is a DNF formula not equivalent to any smaller DNF formula}\} \,.\\ &= \{\, \llcorner\varphi\lrcorner :\ \forall \psi, |\psi| < |\varphi|\,, \exists \text{ assignment } s \text{ such that } \varphi(s) \neq \psi(s)\} \,.\end{aligned}$$

Again, there is no obvious notion of a certificate of membership. Note that both the above problems are in **PSPACE**, but neither is believed to be **PSPACE**-complete.

It seems that the way to capture such languages is to allow not only an “exists“ quantifier (as in Definition 2.1 of **NP**) or only a “for all” quantifier (as Definition 2.22 of **coNP**) but a combination of both quantifiers. This motivates the following definition:

Definition 5.1
The class $\mathbf{\Sigma}_2^p$ is defined to be the set of all languages L for which there exists a polynomial-time TM M and a polynomial q such that

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{q(|x|)}\, \forall v \in \{0,1\}^{q(|x|)}\, M(x,u,v) = 1$$

for every $x \in \{0,1\}^*$.

Note that $\mathbf{\Sigma}_2^p$ contains both the classes **NP** and **coNP**.

Example 5.2
The language $\mathsf{EXACT\ INDSET}$ above is in $\mathbf{\Sigma}_2^p$, since as we noted above, a pair $\langle G, k\rangle$ is in $\mathsf{EXACT\ INDSET}$ iff

$$\begin{aligned}\exists\ S\ \forall S' \quad &\text{set } S \text{ is an independent set of size } k \text{ in } G \text{ and}\\ &\qquad\qquad S' \text{ is not a independent set of size} \geq k+1.\end{aligned}$$

We define the class $\mathbf{\Pi}_2^p$ to be the set $\left\{\overline{L} : L \in sig_2^p\right\}$. It is easy to see that an equivalent definition is that $L \in \mathbf{\Pi}_2^p$ if there is a polynomial-time TM M and a polynomial q such that

$$x \in L \Leftrightarrow \forall u \in \{0,1\}^{q(|x|)}\, \exists v \in \{0,1\}^{q(|x|)}\, M(x,u,v) = 1$$

for every $x \in \{0,1\}^*$.

Example 5.3
The language $\mathsf{EXACT\ INDSET}$ is also in $\mathbf{\Pi}_2^p$ since a pair $\langle G, k\rangle$ is in $\mathsf{EXACT\ INDSET}$ if *for every* S', if S' has size at least $k+1$ then it is not an independent set, but *there exists* an independent set S of size k in G. (Exercise 8 shows a finer placement of $\mathsf{EXACT\ INDSET}$.)

The reader can similarly check that $\mathsf{MIN-DNF}$ is in $\mathbf{\Pi}_2^p$. It is conjectured to be complete for $\mathbf{\Pi}_2^p$.

5.2 The polynomial hierarchy.

The polynomial hierarchy generalizes the definitions of $\mathbf{NP}, \mathbf{coNP}, \mathbf{\Sigma}_2^p, \mathbf{\Pi}_2^p$ to consists all the languages that can be defined via a combination of a polynomial-time computable predicate and a constant number of $\forall/\exists$ quantifiers:

Definition 5.4 (Polynomial Hierarchy)
For every $i \geq 1$, a language L is in $\mathbf{\Sigma}_i^p$ if there exists a polynomial-time TM M and a polynomial q such that

$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} \, \forall u_2 \in \{0,1\}^{q(|x|)} \cdots Q_i u_i \in \{0,1\}^{q(|x|)} \, M(x, u_1, \ldots, u_i) = 1,$$

where Q_i denotes $\forall$ or $\exists$ depending on whether i is even or odd respectively.

We say that L is in $\mathbf{\Pi}_i^p$ if there exists a polynomial-time TM M and a polynomial q such that

$$x \in L \Leftrightarrow \forall u_1 \in \{0,1\}^{q(|x|)} \, \exists u_2 \in \{0,1\}^{q(|x|)} \cdots Q_i u_i \in \{0,1\}^{q(|x|)} \, M(x, u_1, \ldots, u_i) = 1,$$

where Q_i denotes $\exists$ or $\forall$ depending on whether i is even or odd respectively.

The *polynomial hierarchy* is the set $\mathbf{PH} = \cup_i \mathbf{\Sigma}_i^p$.

Remark 5.5
Note that $\mathbf{\Sigma}_1^p = \mathbf{NP}$ and $\mathbf{\Pi}_2^p = \mathbf{coNP}$. More generally, for evert $i \geq 1$, $\mathbf{\Pi}_i^p = \mathbf{co}\mathbf{\Sigma}_i^p = \left\{ \overline{L} : L \in \mathbf{\Sigma}_i^p \right\}$. Note also that that $\mathbf{\Sigma}_i^p \subseteq \mathbf{\Pi}_{i+1}^p$, and so we can also define the polynomial hierarchy as $\cup_{i>0} \mathbf{\Pi}_i^p$.

5.2.1 Properties of the polynomial hierarchy.

We believe that $\mathbf{P} \neq \mathbf{NP}$ and $\mathbf{NP} \neq \mathbf{coNP}$. An appealing generalization of these conjectures is that for every i, $\mathbf{\Sigma}_i^p$ is strictly contained in $\mathbf{\Sigma}_{i+1}^p$. This is called the conjecture that the polynomial hierarchy does not collapse, and is used often in complexity theory. If the polynomial hierarchy does collapse this means that there is some i such that $\mathbf{\Sigma}_i^p = \cup_j \mathbf{\Sigma}_j^p = \mathbf{PH}$. In this case we say that the polynomial hierarchy has collapsed to the i^{th} level. The smaller i is, the weaker, and hence more plausible, is the conjecture that $\mathbf{PH}$ does not collapse to the i^{th} level.

Theorem 5.6

1. *For every $i \geq 1$, if $\mathbf{\Sigma}_i^p = \mathbf{\Pi}_i^p$ then $\mathbf{PH} = \mathbf{\Sigma}_i^p$ (i.e., the hierarchy collapses to the i^{th} level).*

2. *If $\mathbf{P} = \mathbf{NP}$ then $\mathbf{PH} = \mathbf{P}$ (i.e., the hierarchy collapses to $\mathbf{P}$).*

Proof: We do the second part; the first part is similar and also easy.

Assuming $\mathbf{P} = \mathbf{NP}$ we prove by induction on i that $\mathbf{\Sigma}_i^p, \mathbf{\Pi}_i^p \subseteq \mathbf{P}$. Clearly this is true for $i = 1$ since under our assumption $\mathbf{P} = \mathbf{NP} = \mathbf{coNP}$. We assume it is true for $i-1$ and prove it for i. Let

$L \in \mathbf{\Sigma}_i^p$, we will show that $L \in \mathbf{P}$. By definition, there is a polynomial-time M and a polynomial q such that

$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)}\, \forall u_2 \in \{0,1\}^{q(|x|)} \cdots Q_i u_i \in \{0,1\}^{q(|x|)}\, M(x, u_1, \ldots, u_i) = 1,$$

where Q_i is $\exists/\forall$ as in Definition 5.4. Define the language L' as follows:

$$u \in L' \Leftrightarrow \exists\forall u_2 \in \{0,1\}^{q(|x|)} \cdots Q_i u_i \in \{0,1\}^{q(|x|)}\, M(u_1, u_2, \ldots, u_i) = 1.$$

Clearly, $L' \in \mathbf{\Pi}_{i-1}^p$ and so under our assumption is in $\mathbf{P}$. This implies that there is a TM M' such that

$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)}\, M'(x, u_1) = 1\,.$$

But this means $L \in \mathbf{NP}$ and hence under our assumption $L \in \mathbf{P}$. The same idea shows that if $L \in \mathbf{\Pi}_i^p$ then $L \in \mathbf{P}$. ■

5.2.2 Complete problems for levels of PH

For every i, we say that a language L is $\mathbf{\Sigma}_i^p$-complete if $L \in \mathbf{\Sigma}_i^p$ and for every $L' \in \mathbf{\Sigma}_i^p$, $L' \leq_p L$. We define $\mathbf{\Pi}_i^p$-completeness and **PH**-completeness in the same way. In this section we show that for every $i \in \mathbb{N}$, both $\mathbf{\Sigma}_i^p$ and $\mathbf{\Pi}_i^p$ have complete problems. In contrast the polynomial hierarchy itself is believed not to have a complete problem, as is shown by the following simple claim:

CLAIM 5.7
Suppose that there exists a language L that is **PH**-*complete, then there exists an i such that* $\mathbf{PH} = \mathbf{\Sigma}_i^p$ *(and hence the hierarchy collapses to its i^{th} level.)*

PROOF SKETCH: Since $L \in \mathbf{PH} = \cup_i \mathbf{\Sigma}_i^p$, there exists i such that $L \in \mathbf{\Sigma}_i^p$. Since L is **PH**-complete, we can reduce every language of **PH** to $\mathbf{\Sigma}_i^p$ to L, and thus $\mathbf{PH} \subseteq \mathbf{\Sigma}_i^p$. ■

REMARK 5.8
It is not hard to see that $\mathbf{PH} \subseteq \mathbf{PSPACE}$. A simple corollary of Claim 5.7 is that unless the polynomial hierarchy collapses, $\mathbf{PH} \neq \mathbf{PSPACE}$. Indeed, otherwise the problem TQBF would be **PH**-complete.

EXAMPLE 5.9
The following are some examples for complete problems for individual levels of the hierarchy:

For every $i \geq 1$, the class $\mathbf{\Sigma}_i^p$ has the following complete problem involving quantified boolean expression with limited number of alternations:

$$\Sigma_i\mathsf{SAT} = \exists u_1 \forall u_2 \exists \cdots Q_i u_i\ \varphi(u_1, u_2, \ldots, u_i) = 1, \tag{1}$$

where φ is a Boolean formula (not necessarily in CNF form, although this does not make much difference), each u_i is a vector of boolean variables, and Q_i is $\forall$ *or*$\exists$ depending on whether i is odd or even. Notice that this is a special case of the TQBF problem defined in Chapter 4. Exercise 1 asks you to prove that $\Sigma_i\mathsf{SAT}$ is indeed $\mathbf{\Sigma}_i^p$-complete. One can similarly define a problem $\Pi_i\mathsf{SAT}$ that is $\mathbf{\Pi}_i^p$-complete.

In the SUCCINCT SET COVER problem we are given a collection $S = \{\varphi_1, \varphi_2, \ldots, \varphi_m\}$ of 3-DNF formulae on n variables, and an integer k. We need to determine whether there is a subset $S' \subseteq \{1, 2, \ldots, m\}$ of size at most K for which $\vee_{i \in S'} \varphi_i$ is a tautology (i.e., evaluates to 1 for every assignment to the variables). Umans showed that SUCCINCT SET COVER is $\mathbf{\Sigma}_2^p$-complete [**?**].

5.3 Alternating Turing machines

Alternating Turing Machines (ATM), are generalizations of nondeterministic Turing machines. Recall that even though NDTMs are not a realistic computational model, studying them helps us to focus on a natural computational phenomenon, namely, the apparent difference between *guessing* an answer and *verifying* it. ATMs plays a similar role for certain languages for which there is no obvious short *certificate* for membership and hence cannot be characterized using nondeterminism alone.

Alternating TMs are similar to NDTMs in the sense that they have *two* transition functions between which they can choose in each step, but they also have the additional feature that every internal state except q_{accept} and q_{halt} is labeled with either $\exists$ or $\forall$. Similar to the NDTM, an ATM can evolve at every step in two possible ways. Recall that a non-deterministic TM accepts its input if there *exists* some sequence of choices that leads it to the state q_{accept}. In an ATM, the *exists* quantifier over each choice is replaced with the appropriate quantifier according to the labels.

Definition 5.10

Let M be an alternating TM. For a function $T : \mathbb{N} \to \mathbb{N}$, we say that M is an $T(n)$-time ATM if for every input $x \in \{0,1\}^*$ and for every possible sequence of transition function choices, M will halt after at most $T(|x|)$ steps.

For every $x \in \{0,1\}^*$, we let $G_{M,x}$ be the configuration graph of x, whose vertices are the configurations of M on input x and there is an edge from configuration C to C' if there C' can be obtained from C in one step using one of the two transition functions (see Section 4.1). Recall that this is a directed acyclic graph. We label some of the nodes in the graph by "ACCEPT" by repeatedly applying the following rules until they cannot be applied anymore:

- The configuration C_{accept} where the machine is in q_{accept} is labeled "ACCEPT".
- If a configuration C is in a state labeled $\exists$ and one of the configurations C' reachable from it in one step is labeled "ACCEPT" then we label C "ACCEPT".
- If a configuration C is in a state labeled $\forall$ and both the configurations C', C'' reachable from it one step is labeled "ACCEPT" then we label C "ACCEPT".

We say that M *accepts* x if at the end of this process the starting configuration C_{start} is labeled "ACCEPT". The *language accepted by* M is the set of all x's such that M accepts x. We denote by $\mathbf{ATIME}(T(n))$ the set of all languages accepted by some $T(n)$-time ATM.

For every $i \in \mathbb{N}$, we define $\mathbf{\Sigma}_i\mathbf{TIME}(T(n))$ (resp. $\mathbf{\Pi}_i\mathbf{TIME}(T(n))$ to be the set of languages accepted by a $T(n)$-time ATM M whose initial state is labeled "$\exists$" (resp. "$\forall$") and on which every input and sequence of choices leads M to change at most $i-1$ times from states with one label to states with the other label.

The following claim is left as an easy exercise (see Exercise 2):

Claim 5.11
For every $i \in \mathbb{N}$,

$$\mathbf{\Sigma}_i^p = \cup_c \mathbf{\Sigma}_i\mathbf{TIME}(n^c)$$
$$\mathbf{\Pi}_i^p = \cup_c \mathbf{\Pi}_i\mathbf{TIME}(n^c)$$

5.3.1 Unlimited number of alternations?

What if we consider polynomial-time alternating Turing machines with no *a priori* bound on the number of quantifiers? We define the class **AP** to be $\cup_c \mathbf{ATIME}(n^c)$. We have the following theorem:

Theorem 5.12
AP = **PSPACE**.

Proof: **PSPACE** ⊆ **AP** follows since TQBF is trivially in **AP** (just "guess" values for each existentially quantified variable using an $\exists$ state and for universally quantified variables using a $\forall$ state) and every **PSPACE** language reduces to TQBF.

AP ⊆ **PSPACE** follows using a recursive procedure similar to the one used to show that TQBF ∈ **PSPACE**. ■

Similarly, one can consider alternating Turing machines that run in polynomial space. The class of languages accepted by such machines is called **APSPACE**, and Exercise 6 asks you to prove that **APSPACE** = **EXP**. One can similarly consider alternating logspace machines; the set of languages accepted by them is exactly **P**.

5.4 Time versus alternations: time-space tradeoffs for SAT.

Despite the fact that SAT is widely believed to require exponential (or at least super-polynomial) time to solve, and to require linear (or at least super-logarithmic) space, we currently have no way to prove these conjectures. In fact, as far as we know, SAT may have both a linear time algorithm and a logarithmic space one. Nevertheless, we *can* prove that SAT does not have an algorithm that runs *simultaneously* in linear time and logarithmic space. In fact, we can prove the following stronger theorem:

Theorem 5.13 (??)
For every two functions $S, T : \mathbb{N} \to \mathbb{N}$, define $\mathbf{TISP}(T(n), S(n))$ to be the set of languages decided by a TM M that on every input x takes at most $O(T(|x|))$ steps and uses at most $O(S(|x|))$ cells of its read/write tapes. Then, SAT $\notin \mathbf{TISP}(n^{1.1}, n^{0.1})$.

Remark 5.14
The class $\mathbf{TISP}(T(n), S(n))$ is typically defined with respect to TM's with RAM memory (i.e., TM's that have random access to their tapes; such machines can be defined in a similar way to the definition of oracle TM's in Section 3.5). Theorem 5.13 and its proof carries over for that model as well. We also note that a stronger result is known for both models: for every $c < (\sqrt{5}+1)/2$, there exists $d > 0$ such that $\mathsf{SAT} \notin \mathbf{TISP}(n^c, n^d)$ and furthermore, d approaches 1 from below as c approaches 1 from above.

Proof: We will show that

$$\mathbf{NTIME}(n) \nsubseteq \mathbf{TISP}(n^{1.2}, n^{0.2})\,. \tag{2}$$

This implies the result for SAT by following the ideas of the proof of the Cook-Levin Theorem (Theorem 2.10). A careful analysis of that proof yields a reduction from the task of deciding membership in an $\mathbf{NTIME}(T(n))$-language to the task deciding whether an $O(T(n)\log T(n))$-sized formula is satisfiable, such that every output bit of this reduction can be computed in polylogarithmic time and space. (See also the proof of Theorem 6.7 for a similar analysis.) Hence, if $\mathsf{SAT} \in \mathbf{TISP}(n^{1.1}, n^{0.1})$ then $\mathbf{NTIME}(n) \subseteq \mathbf{TISP}(n^{1.1}\mathrm{polylog}(n), n^{0.1}\mathrm{polylog}(n))$. Our main step in proving (2) is the following claim, showing how to replace time with alternations:

Claim 5.14.1
$\mathbf{TISP}(n^{12}, n^2) \subseteq \mathbf{\Sigma}_2\mathbf{TIME}(n^8)$.

Proof: The proof is similar to the proofs of Savitch's Theorem and the **PSPACE**-completeness of TQBF (Theorems 4.12 and 4.11). Suppose that L is decided by a machine M using n^{12} time and n^2 space. For every $x \in \{0,1\}^*$, consider the configuration graph $G_{M,x}$ of M on input x. Each configuration in this graph can be described by a string of length $O(n^2)$ and x is in L if and only if there is a path of length n^{12} in this graph from the starting configuration C_{start} to an accepting configuration. There is such a path if and only if there *exist* n^6 configurations $C_1, \ldots, C_{n^6}$ (requiring $O(n^8)$ to specify) such that if we let $C_0 = C_{\mathsf{start}}$ then C_{n^6} is accepting and *for every* $i \in [n^6]$ the configuration C_i is computed from C_{i-1} within n^6 steps. Because this condition can be verified in n^6 time, we can we get an $O(n^8)$-time σ_2-TM for deciding membership in L. ■

Our next step will show that, under the assumption that (2) does not hold (and hence $\mathbf{NTIME}(n) \subseteq \mathbf{TISP}(n^{1.2}, n^{0.2})$), we can replace alternations with time:

Claim 5.14.2
Suppose that $\mathbf{NTIME}(n) \subseteq \mathbf{DTIME}(n^{1.2})$. *Then* $\mathbf{\Sigma}_2\mathbf{TIME}(n^8) \subseteq \mathbf{NTIME}(n^{9.6})$.

Proof: Using the characterization of the polynomial hierarchy by alternating machines, L is in $\mathbf{\Sigma}_2\mathbf{TIME}(n^8)$ if and only if there is an $O(n^8)$-time TM M such that

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{c|x|^8}\, \forall v \in \{0,1\}^{d|x|^8}\, M(x,u,v) = 1\,.$$

for some constants c, d. Yet if $\mathbf{NTIME}(n) \subseteq \mathbf{DTIME}(n^{1.2})$ then by a simple padding argument (a la the proof of Theorem 2.25) we have a deterministic algorithm D that on inputs x, u with $|x| = n$

and $|u| = cn^8$ runs in time $O((n^8)^{1.2}) = O(n^{9.6})$-time and returns 1 if and only if there exists some $v \in \{0,1\}^{dn^8}$ such that $M(x,u,v) = 0$. Thus,

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{c|x|^8} D(x,u) = 0 \,.$$

implying that $L \in \mathbf{NTIME}(n^{9.6})$. ■

Claims 5.14.1 and 5.14.2 show that the assumption that $\mathbf{NTIME}(n) \subseteq \mathbf{TISP}(n^{1.2}, n^{0.2})$ leads to contradiction: by simple padding it implies that $\mathbf{NTIME}(n^{10}) \subseteq \mathbf{TISP}(n^{12}, n^2)$ which by Claim 5.14.1 implies that $\mathbf{NTIME}(n^{10}) \subseteq \mathbf{\Sigma}_2\mathbf{TIME}(n^8)$. But together with Claim 5.14.2 this implies that $\mathbf{NTIME}(n^{10}) \subseteq \mathbf{NTIME}(n^{9.6})$, contradicting the non-deterministic time hierarchy theorem (Theorem 3.3). ■

5.5 Defining the hierarchy via oracle machines.

Recall the definition of *oracle machines* from Section 3.5. These are machines that are executed with access to a special tape they can use to make queries of the form "is $q \in O$" for some language O. For every $O \subseteq \{0,1\}^*$, oracle TM M and input x, we denote by $M^O(x)$ the output of M on x with access to O as an oracle. We have the following characterization of the polynomial hierarchy:

Theorem 5.15
For every $i \geq 2$, $\mathbf{\Sigma}_i^p = \mathbf{NP}^{\mathbf{\Sigma}_{i-1}\mathsf{SAT}}$, where the latter class denotes the set of languages decided by polynomial-time NDTMs with access to the oracle $\mathbf{\Sigma}_{i-1}\mathsf{SAT}$.

Proof: We showcase the idea by proving that $\mathbf{\Sigma}_2^p = \mathbf{NP}^{\mathsf{SAT}}$. Suppose that $L \in \mathbf{\Sigma}_2^p$. Then, there is a polynomial-time TM M and a polynomial q such that

$$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} M(x,u_1,u_2) = 1$$

yet for every fixed u_1 and x, the statement "for every u_2, $M(x,u_1,u_2) = 1$" is the negation of an **NP**-statement and hence its truth can be determined using an oracle for SAT. We get that there is a simple NDTM N that given oracle access for SAT can decide L: on input x, non-deterministically guess u_1 and use the oracle to decide if $\forall u_2 M(x,u_1,u_2) = 1$. We see that $x \in L$ iff there exists a choice u_1 that makes N accept.

On the other hand, suppose that L is decidable by a polynomial-time NDTM N with oracle access to SAT. Then, x is in L if and only if there exists a sequence of non-deterministic choices and correct oracle answers that makes N accept x. That is, there is a sequence of choices $c_1, \ldots, c_m \in \{0,1\}$ and answers to oracle queries $a_1, \ldots, a_k \in \{0,1\}$ such that on input x, if the machine N uses the choices $c_1, \ldots, c_m$ in its execution and receives a_i as the answer to its i^{th} query, then **(1)** M reaches the accepting state q_{accept} and **(2)** all the answers are correct. Let φ_i denote the i^{th} query that M makes to its oracle when executing on x using choices $c_1, \ldots, x_m$ and receiving answers $a_1, \ldots, a_k$. Then, the condition **(2)** can be phrased as follows: if $a_i = 1$ then there exists an assignment u_i such that $\varphi_i(u_i) = 1$ and if $a_i = 0$ then for every assignment v_i, $\varphi_i(v_i) = 0$. Thus, we have that

$$\begin{aligned} x \in L \Leftrightarrow & \exists c_1, \ldots, c_m, a_1, \ldots, a_k, u_1, \ldots, u_k \forall v_1, \ldots, v_k \text{ such that} \\ & N \text{ accepts } x \text{ using choices } c_1, \ldots, c_m \text{ and answers } a_1, \ldots, a_k \text{ AND} \\ & \forall i \in [k] \text{ if } a_i = 1 \text{ then } \varphi_i(u_i) = 1 \text{ AND} \\ & \forall i \in [k] \text{ if } a_i = 0 \text{ then } \varphi_i(v_i) = 0 \end{aligned}$$

implying that $L \in \mathbf{\Sigma}_2^p$. ■

REMARK 5.16
Because having oracle access to a complete language for a class allows to solve every language in that class, some texts use the class name instead of the complete language in the notation for the oracle. Thus, some texts denote the class $\mathbf{\Sigma}_2^p = \mathbf{NP}^{\mathsf{SAT}}$ by $\mathbf{NP}^{\mathbf{NP}}$, the class $\mathbf{\Sigma}_3^p$ by $\mathbf{NP}^{\mathbf{NP}^{\mathbf{NP}}}$ and etc.

WHAT HAVE WE LEARNED?

- The polynomial hierarchy is the set of languages that can be defined via a constant number of alternating quantifiers. It also has equivalent definitions via alternating TMs and oracle TMs. It contains several natural problems that are not known (or believed) to be in **NP**.
- We conjecture that the hierarchy does not collapse in the sense that each of its levels is distinct from the previous ones.
- We can use the concept of alternations to prove that SAT cannot be solved simultaneously in linear time and sublinear space.

Chapter notes and history

The polynomial hierarchy was formally defined by Stockmeyer [?], though the concept appears in the literature even earlier. For instance, Karp [?] notes that *"a polynomial-bounded version of Kleene's Arithmetic Hierarchy (Rogers 1967) becomes trivial if* $\mathbf{P} = \mathbf{NP}$*."*

The class **DP** was defined by Papadimitriou and Yannakakis [?], who used it to characterize the complexity of identifying the facets of a polytope.

The class of complete problems for various levels of **PH** is not as rich as it is for **NP**, but it does contain several interesting ones. See Schaeffer and Umans [?, ?] for a recent list. The SUCCINCT SET-COVER problem is from Umans [?], where it is also shown that the following optimization version of MIN-DNF is $\mathbf{\Sigma}_2^p$-complete:

$$\left\{ \langle \varphi, k \rangle : \ \exists \text{ DNF} \varphi' \text{ of size at most } k \text{, that is equivalent to DNF } \varphi \right\}.$$

Exercises

§1 Show that the language Σ_iSAT of (1) is complete for $\mathbf{\Sigma}_i^p$ under polynomial time reductions.

Hint:: Use the **NP**-completeness of SAT.

§2 Prove Claim 5.11.

§3 Show that if 3SAT is polynomial-time reducible to $\overline{\text{3SAT}}$ then $\mathbf{PH} = \mathbf{NP}$.

§4 Show that **PH** has a complete language iff it collapses to some finite level $\mathbf{\Sigma}_i^p$.

§5 Show that the definition of **PH** using ATMs coincides with our other definitions.

§6 Show that $\mathbf{APSPACE} = \mathbf{EXP}$.

Hint: The nontrivial direction $\mathbf{EXP} \subseteq \mathbf{APSPACE}$ uses ideas similar to those in the proof of Theorem 5.13.

§7 Show that $\mathbf{\Sigma}_2^p = \mathbf{NP}^{\mathsf{SAT}}$. Generalize your proof to give a characterization of **PH** in terms of oracle Turing machines.

§8 The class **DP** is defined as the set of languages L for which there are two languages $L_1 \in \mathbf{NP}, L_2 \in \mathbf{coNP}$ such that $L = L_1 \cap L_2$. (Do not confuse **DP** with $\mathbf{NP} \cap \mathbf{coNP}$, which may seem superficially similar.) Show that

(a) EXACT INDSET $\in$ **DP**.

(b) Every language in **DP** is polynomial-time reducible to EXACT INDSET.

§9 Suppose A is some language such that $\mathbf{P}^A = \mathbf{NP}^A$. Then show that $\mathbf{PH}^A \subseteq \mathbf{P}^A$ (in other words, the proof of Theorem **??** *relativizes*).

§10 Show that SUCCINCT SET-COVER $\in \mathbf{\Sigma}_2^p$.

§11 (Suggested by C. Umans) This problem studies VC-dimension, a concept important in machine learning theory. If $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ is a collection of subsets of a finite set U, the *VC dimension* of $\mathcal{S}$, denoted $VC(S)$, is the size of the largest set $X \subseteq U$ such that for every $X' \subseteq X$, there is an i for which $S_i \cap X = X'$. (We say that X is *shattered* by S.)

A boolean circuit C succinctly represents collection $\mathcal{S}$ if S_i consists of exactly those elements $x \in U$ for which $C(i, x) = 1$. Finally,

$$\text{VC-DIMENSION} = \{\langle C, k\rangle : \ C \text{ represents a collection } \mathcal{S} \text{ s.t. } VC(S) \geq k\}.$$

(a) Show that VC-DIMENSION $\in \mathbf{\Sigma}_3^p$.

(b) Show that VC-DIMENSION is $\mathbf{\Sigma}_3^p$-complete.

Hint: Reduce from $\mathbf{\Sigma}_3$-3SAT. Also, the collection $\mathcal{S}$ produced by your reduction can use the same set multiple times.