

The Polynomial Hierarchy

Slides based on *S.Aurora, B.Barak. Complexity Theory: A Modern Approach.*

Ahto Buldas

Ahto.Buldas@ut.ee

Motivation

“..synthesizing circuits is exceedingly difficult. 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 lecture discusses the polynomial hierarchy, a generalization of P , NP and $coNP$. We will provide four equivalent definitions for the polynomial hierarchy, using:

- quantified predicates,
- alternating Turing machines,
- oracle Turing machines, and
- uniform families of circuits (next lecture).

We also use the hierarchy to show that solving the SAT problem requires either super-logarithmic space or super-linear time.

INDSET and EXACTINDSET

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:

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

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

$$\text{EXACTINDSET} = \{ \langle G, k \rangle : \text{The largest ind. set in } G \text{ has size exactly } k \} .$$

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

MIN – DNF

Similarly, consider the language MIN – DNF, the decision version of a problem in circuit minimization, a topic of interest in electrical engineering.

We say that two boolean formulae are *equivalent* if they have the same set of satisfying assignments.

$$\begin{aligned} \text{MIN – DNF} &= \{\varphi : \varphi \text{ is a DNF not equivalent to any smaller DNF.}\} \\ &= \{\varphi : \forall \psi, |\psi| < |\varphi|, \exists s : \psi(s) \neq \varphi(s)\} . \end{aligned}$$

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.

The classes Σ_2^p and Π_2^p

Def.: The class Σ_2^p is the set of all languages L for which there exists a poly-time M and a polynomial q such that for every $x \in \{0, 1\}^*$:

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

Def.: The class Π_2^p is the set of all languages L for which there exists a poly-time M and a polynomial q such that for every $x \in \{0, 1\}^*$:

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

It follows from the definitions that:

- $\text{NP}, \text{coNP} \in \Sigma_2^p \cap \Pi_2^p$;
- $L \in \Pi_2^p$ iff $\bar{L} \in \Sigma_2^p$.

Examples

EXACTINDSET $\in \Sigma_2^p$: Since a pair $\langle G, k \rangle$ is in EXACTINDSET iff:

$\exists S \forall S'$: S is an ind-set of size k and S' is not a ind-set of size $\geq k + 1$.

EXACTINDSET $\in \Pi_2^p$: Since a pair $\langle G, k \rangle$ is in EXACTINDSET iff:

$\forall S' \exists S$: if $|S'| \geq k + 1$ then S' is not an ind-set, but S is an ind-set of size k .

MIN – DNF $\in \Pi_2^p$: Since a pair φ is in MIN – DNF iff:

$\forall \psi \exists s$: $|\psi| < |\varphi| \Rightarrow \psi(s) \neq \varphi(s)$.

MIN – DNF is conjectured to be *complete* for Π_2^p .

The Polynomial Hierarchy

Def. 5.4 (Polynomial Hierarchy): For every $i \geq 1$, a language L is in Σ_i^p if there exists a poly-time M and a polynomial q such that for every $x \in \{0, 1\}^*$:

$$x \in L \Leftrightarrow \exists u_1 \in C, \forall u_2 \in C \dots Q_i u_i \in C: M(x, u_1, \dots, u_i) = 1,$$

where $C = \{0, 1\}^{q(|x|)}$ and Q_i denotes \forall or \exists depending on whether i is even or odd respectively. We say that L is in Π_i^p if there exists a poly-time M and a polynomial q such that for every $x \in \{0, 1\}^*$:

$$x \in L \Leftrightarrow \forall u_1 \in C, \exists u_2 \in C \dots Q_i u_i \in C: M(x, u_1, \dots, u_i) = 1,$$

where Q_i denotes \forall or \exists depending on whether i is odd or even respectively. The *polynomial hierarchy* is the set $\mathbf{PH} = \cup_i \Sigma_i^p$.

Remark: $\Sigma_1^p = \mathbf{NP}$ and $\Pi_1^p = \mathbf{coNP}$. More generally, $\Pi_i^p = \mathbf{co}\Sigma_i^p$. Note also that $\Sigma_i^p \subseteq \Pi_{i+1}^p$, and so $\mathbf{PH} = \cup_i \Pi_i^p$.

Properties of the Polynomial Hierarchy

We believe that $P \neq NP$ and $NP \neq \text{coNP}$. An appealing generalization of these conjectures is that for every i , the class Σ_i^P is strictly contained in Σ_{i+1}^P . This is called the conjecture that the polynomial hierarchy *does not collapse*. If the polynomial hierarchy does collapse this means that there is i such that $\Sigma_i^P = \cup_j \Sigma_j^P = \text{PH}$. 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 PH does not collapse to the i -th level.

Theorem: For every $i \geq 1$, if $\Sigma_i^P = \Pi_i^P$ then $\text{PH} = \Sigma_i^P$, i.e. PH collapses to the i^{th} level. If $P = NP$ then $\text{PH} = P$, i.e. PH collapses to P .

Proof: We do the second part: the first part is similar and also easy. Assuming $P = NP$, we prove by induction on i that $\Sigma_i^P, \Pi_i^P \subseteq P$. Clearly, this

is true for $i = 1$ since under our assumption $\mathbf{NP} = \mathbf{P} = \mathbf{coP} = \mathbf{coNP}$. We assume it is true for $i - 1$ and prove it for i . Assuming $L \in \Sigma_i^P$, we will show that $L \in \mathbf{P}$. By definition, there is a poly-time M and a polynomial q such that (if every certificate $u_i \in C = \{0, 1\}^{q(|x|)}$):

$$x \in L \Leftrightarrow \exists u_1 \forall u_2 \dots Q_i u_i : M(x, u_1, \dots, u_i) = 1 .$$

Define the language L' as follows:

$$(x, u) \in L' \Leftrightarrow \forall u_2 \dots Q_i u_i : M(x, u, u_2, \dots, u_i) = 1 ,$$

i.e. $x \in L$ iff there is u_1 such that $(x, u_1) \in L'$. By assumptions, $L' \in \Pi_{i-1}^P = \mathbf{P}$ and so there is a poly-time M' such that $M'(x, u) = 1 \Leftrightarrow (x, u) \in L'$. Hence,

$$x \in L \Leftrightarrow \exists u_1 : M'(x, u_1) = 1 ,$$

which means that $L \in \mathbf{NP} = \mathbf{P}$.

Complete Problems for Levels of PH

Def.: For every i , we say that a language L is Σ_i^P -complete if $L \in \Sigma_i^P$ and $L' \leq_p L$ for every $L' \in \Sigma_i^P$. We define Π_i^P -completeness and **PH**-completeness in the same way.

We will show that for every $i \in \mathbb{N}$, both Σ_i^P and Π_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: Suppose that there exists a language L that is **PH**-complete, then there exists an i such that $\mathbf{PH} = \Sigma_i^P$ and hence the hierarchy collapses to its i th level.

Proof sketch: Since $L \in \mathbf{PH} = \cup_i \Sigma_i^P$, there exists i such that $L \in \Sigma_i^P$.

Since L is **PH**-complete, we can reduce every language of **PH** to Σ_i^p to L , and thus $\mathbf{PH} \subseteq \Sigma_i^p$.

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

Example: For every $i \geq 1$, the class Σ_i^p has the following complete problem $\Sigma_i^p\text{SAT}$ involving quantified boolean expression with limited number of alternations:

$$\exists u_1 \forall u_2 \exists u_3 \dots Q_i u_i \varphi(u_1, u_2, \dots, u_i) = 1 ,$$

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. One can similarly define a problem $\Pi_i\text{SAT}$ that is Π_i^p -complete.

Example: In the SUCCINCT – SET – COVER problem we are given a collection $S = \{\varphi_1, \varphi_2, \dots, \varphi_m\}$ of 3-DNF formulae on n variables, and an integer k . We need to determine whether there is a subset $S' \subseteq \{1, 2, \dots, m\}$ of size at most k for which $\bigvee_{i \in S'} \varphi_i$ is a tautology. Umans showed that this problem is Σ_2^p -complete.

Alternating Turing Machines (ATM)

ATMs are generalizations of nondeterministic Turing machines. 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.

Def. Let M be an alternating TM. For a function $T: \mathbb{N} \rightarrow \mathbb{N}$, 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 C' can be obtained from C in one step using one of the two transition functions. 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 in 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 $\text{ATIME}(T(n))$ the set of all languages accepted by some $T(n)$ -time ATM.

For every $i \in \mathbb{N}$, we define $\Sigma_i \text{TIME}(T(n))$ (resp. $\Pi_i \text{TIME}(T(n))$) 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:

Claim: For every $i \in \mathbb{N}$:

$$\Sigma_i^P = \cup_c \Sigma_i \text{TIME}(n^c), \quad \Pi_i^P = \cup_c \Pi_i \text{TIME}(n^c) .$$

Unlimited number of alternations?

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

Theorem: $\text{AP} = \text{PSPACE}$.

Proof: $\text{PSPACE} \subseteq \text{AP}$ 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. $\text{AP} \subseteq \text{PSPACE}$ follows using a recursive procedure similar to the one used to show that $\text{TQBF} \in \text{PSPACE}$.

Similarly, one can consider alternating Turing machines that run in polynomial space. The class of languages accepted by such machines is called

APSPACE. We can prove that $\text{APSPACE} = \text{EXP}$. One can similarly consider alternating log-space machines; the set of languages accepted by them is exactly P .

The Padding Argument

If $\mathbf{CL}_1(f(n))$ and $\mathbf{CL}_2(g(n))$ are complexity classes that are characterized by the resources (time or space) they allow to spend to the machines that accept languages belonging to these classes. The resources are measured by functions $f(n)$ and $g(n)$ respectively (with O -precision), where n is the input size.

Theorem: If $\mathbf{CL}_1(f(n)) \subseteq \mathbf{CL}_2(g(n))$ then $\mathbf{CL}_1(f(n^c)) \subseteq \mathbf{CL}_2(g(n^c))$ for every constant $c \in \mathbb{N}$.

Proof: Let $L \in \mathbf{CL}_1(f(n^c))$ and M be a machine that decides L in $f(n^c)$ -time(or space). If $c = 1$, the statement is trivial. Otherwise, if $c \geq 2$, define a new language

$$L' = \{x01^{|x|^c - |x| - 1} : x \in L\} .$$

Define a new machine $M'(y)$ that accepts iff y is in the form $x01^{|x|^c-|x|-1}$ and $M(x) = 1$. Machine M' works with resources $f(|x|^c) = f(|y|)$ and hence,

$$L' \in \mathbf{CL}_1(f(n)) \subseteq \mathbf{CL}_2(g(n)) .$$

Hence, there is a \mathbf{CL}_2 -machine N' that decides L' in $g(n)$ -time (or space). Finally, define a machine $N(x) \equiv N'(x01^{|x|^c-|x|-1})$, which decides L with resources

$$g(|x01^{|x|^c-|x|-1}|) = g(|x|^c) .$$

Hence, $L \in \mathbf{CL}_2(g(|x|^c))$.

Corollary: If $\mathbf{NTIME}(n) \subseteq \mathbf{DTIME}(n^{1.2})$ then

$$\mathbf{NTIME}(n^{10}) \subseteq \mathbf{DTIME}(n^{12}) .$$

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 functions $S, T: \mathbb{N} \rightarrow \mathbb{N}$, 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, $\text{SAT} \notin \mathbf{TISP}(n^{1.1}, n^{0.1})$.

Remark: 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). The 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 $\text{SAT} \notin \mathbf{TISP}(n^c, n^d)$ and furthermore, d approaches 1 from below as c approaches 1 from above.

Proof of Theorem: We will show that $\mathbf{NTIME}(n) \not\subseteq \mathbf{TISP}(n^{1.2}, n^{0.2})$. This implies the result for SAT by following the ideas of the proof of the Cook-Levin Theorem. 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 poly-logarithmic time and space. Hence, if $\text{SAT} \in \mathbf{TISP}(n^{1.1}, n^{0.1})$ then $\mathbf{NTIME}(n) \subseteq \mathbf{TISP}(n^{1.1} \cdot \text{polylog}(n), n^{0.1} \cdot \text{polylog}(n))$.

Replacing Time with Alternations

The main step in proving $\text{NTIME}(n) \not\subseteq \text{TISP}(n^{1.2}, n^{0.2})$ is:

Claim 5.14.1: $\text{TISP}(n^{12}, n^2) \subseteq \Sigma_2\text{TIME}(n^8)$.

Proof: The proof is similar to the proofs of Savitch's Theorem and the PSPACE-completeness of TQBF. 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, \dots, C_{n^6} (requiring $O(n^8)$ to specify) such that if we let $C'_1 = C_{\text{start}}$ then C_{n^6} is accepting and for every $i \in \{1, \dots, n^6\}$ 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 get an $O(n^8)$ -time Σ_2 -TM for deciding membership in L .

Next step: if $\text{NTIME}(n) \not\subseteq \text{TISP}(n^{1.2}, n^{0.2})$ does not hold, and hence $\text{NTIME}(n) \subseteq \text{TISP}(n^{1.2}, n^{0.2})$, we can replace alternations with time:

Claim 5.14.2: If $\text{NTIME}(n) \subseteq \text{DTIME}(n^{1.2})$ then

$$\Sigma_2\text{TIME}(n^8) \subseteq \text{NTIME}(n^{9.6}) .$$

Proof: Using the characterization of the PH by alternating machines, L is in $\Sigma_2\text{TIME}(n^8)$ if and only if there is an $O(n^8)$ -time 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 $\text{NTIME}(n) \subseteq \text{DTIME}(n^{1.2})$, then by the padding argument we have a deterministic algorithm D that on inputs x, u with $|x| = n$ and $|u| = cn^8$ runs in time $O((n^8)^{1.2}) = O(n^{9.6})$ and returns 1 if there is $v \in \{0, 1\}^{dn^8}$ 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 $L \in \text{NTIME}(n^{9.6})$.

Final Proof: Claims 5.14.1 and 5.14.2 show that the assumption that

$$\text{NTIME}(n) \subseteq \text{TISP}(n^{1.2}, n^{0.2}) \stackrel{\text{NB!}}{\subseteq} \text{DTIME}(n^{1.2})$$

leads to a contradiction: by the padding argument, it implies that

$$\text{NTIME}(n^{10}) \subseteq \text{TISP}(n^{12}, n^2)$$

which by Claim 5.14.1 implies that $\text{NTIME}(n^{10}) \subseteq \Sigma_2\text{TIME}(n^8)$. But together with Claim 5.14.2 this implies that

$$\text{NTIME}(n^{10}) \subseteq \text{NTIME}(n^{9.6}) ,$$

contradicting the non-deterministic time hierarchy theorem.

Defining PH with Oracle Machines

Recall that oracle machines are executed with access to queries of the form “is $q \in \mathcal{O}$ ” for some language \mathcal{O} . For every $\mathcal{O} \subseteq \{0, 1\}^*$, an oracle machine M and input x , we denote by $M^{\mathcal{O}}(x)$ the output of M on x with access to \mathcal{O} as an oracle. We have the following characterization of the polynomial hierarchy:

Theorem 5.15: For every $i \geq 2$, $\Sigma_i^p = \text{NP}^{\Sigma_{i-1}\text{SAT}}$, where the latter class denotes the set of languages decided by poly-time non-deterministic machines with access to the oracle $\Sigma_{i-1}\text{SAT}$.

Remark: 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 $\Sigma_2^p = \text{NP}^{\text{SAT}}$ by NP^{NP} , the class Σ_3^p by $\text{NP}^{\text{NP}^{\text{NP}}}$, 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 logarithmic space.