Chapter 4 : Hierarchy theorems and intractability

Recall that we started this course with the basic aim of proving certain computational problems to be more difficult than some others. So far we have made extensive classifications of problems based on the time and space requirements of known algorithms for solving the problems. However, our goal is to analyze problems rather than designing few algorithms for solving them. NP-complete problems are most difficult in the class NP, but are not provably established to be more difficult than problems in P. In case $\mathrm{P} = \mathrm{NP}$, every non-trivial problem in this common class becomes complete (Exercise 2.2.1) — a true set-back to the notion of completeness. Similar comments hold for the class PSPACE and complete problems in it.

The obvious question that arises now is: *Do there exist problems that are provably not in* P*?* We call any such problem an intractable problem. We soon demonstrate that intractable problems *do* exist and are often as easy to describe as SAT or TQBF. Space and time hierarchy theorems allow us to ascertain the existence of and locate such problems.

We then extend the notion of nondeterministic acceptance by a Turing machine. This study leads to another interesting hierarchy of problems inside PSPACE. This hierarchy does not immediately identify intractable problems, but may be a powerful tool for settling the $\mathrm{P} = \mathrm{NP}$? question. We finally introduce relativization techniques in order to establish why the $\mathrm{P} = \mathrm{NP}$? question is unlikely to be tackled by a diagonalization argument, a procedure which plays the key role in undecidability and intractability proofs.

4.1 Space and time hierarchy

Space and time hierarchy theorems essentially state that if we allow asymptotically more space or time, TMs can decide strictly bigger classes of problems. Proving such an assertion requires some intricate details that we now introduce.

4.1 Definition A function $f(n) \geqslant \log n$ (of natural numbers) is said to be space constructible, if there exists an $\mathrm{O}(f(n))$-space DTM that halts with the binary representation of $f(n)$ on its work-tape, when started with 1^n (a string of n 1's) as input. For sublinear functions $f(n)$ we consider the two-tape model of a TM.

A function $f(n) \geqslant n \log n$ is called time constructible, if there exists an $\mathrm{O}(f(n))$-time DTM that halts with the binary representation of $f(n)$ on its tape, when started with 1^n as input.

Most common functions $\geqslant \log n$ are space constructible. Examples include $\log n$, n, n^2, $n^3 \log n$ and 2^n. Similarly, most natural functions $\geqslant n \log n$ are time constructible. Examples: $n \log n$, n^2, $n^3 \log^2 n$, 2^n.

4.2 Theorem [Space hierarchy theorem] For any space constructible function $f(n)$ there exists a language L that is decidable in $\mathrm{O}(f(n))$ space, but not in $\mathrm{o}(f(n))$ space.

This theorem addresses deterministic space. A similar theorem holds for nondeterministic space too. The space hierarchy theorem implies the following proper containments:

$$\begin{aligned}
\mathrm{SPACE}(g(n)) &\subsetneqq \mathrm{SPACE}(f(n)), \text{ where } f \text{ is space constructible and } g(n) = \mathrm{o}(f(n)).\\
\mathrm{SPACE}(n^{c_1}) &\subsetneqq \mathrm{SPACE}(n^{c_2}) \text{ for real values } 0 \leqslant c_1 < c_2 \text{ with } n^{c_2} \text{ space constructible.}\\
\mathrm{PSPACE} &\subsetneqq \mathrm{EXPSPACE}.\\
\mathrm{L} &\subsetneqq \mathrm{PSPACE}.\\
\mathrm{NL} &\subsetneqq \mathrm{PSPACE} \text{ (since by Savitch's theorem NL} \subseteq \mathrm{SPACE}(\log^2 n)).
\end{aligned} \tag{4.1}$$

The similar theorem for time hierarchy is somewhat weaker. In case of space, any asymptotic (and space constructible) increase allows a bigger class of languages to be decided, whereas for time we have to take care of an additional logarithmic factor. Stronger time hierarchy theorems may be true, but no proofs are known. Again we concentrate on deterministic time. A similar result holds for nondeterministic time too.

4.3 Theorem [Time hierarchy theorem] For any space constructible function $f(n)$ there exists a language L that is decidable in $\mathrm{O}(f(n))$ time, but not in $\mathrm{o}(f(n)/\log f(n))$ time.

The time hierarchy theorem immediately implies the following proper inclusions:

$$\begin{aligned} \mathrm{TIME}(g(n)) &\subsetneq \mathrm{TIME}(f(n)), \text{ where } f \text{ is time constructible and } g(n) = \mathrm{o}(f(n)/\log f(n)). \\ \mathrm{TIME}(n^{c_1}) &\subsetneq \mathrm{TIME}(n^{c_2}) \text{ for real values } 1 \leqslant c_1 < c_2 \text{ with } n^{c_2} \text{ time constructible.} \\ \mathrm{P} &\subsetneq \mathrm{EXP}. \end{aligned} \tag{4.2}$$

The proper inclusions (4.1) and (4.2) imply existence of (provably) intractable problems. Let us first define complete problems in these exponential classes.

4.4 Definition A language L is called EXPSPACE-complete, if the following two conditions hold:

(1) $L \in \mathrm{EXPSPACE}$.

(2) $A \leqslant_P L$ for every $A \in \mathrm{EXPSPACE}$.

Similarly, for the exponential time class $\mathrm{EXP} = \mathrm{EXPTIME}$ we define:

4.5 Definition A language L is called EXP-complete, if the following two conditions hold:

(1) $L \in \mathrm{EXP}$.

(2) $A \leqslant_P L$ for every $A \in \mathrm{EXP}$.

If any complete problem in the class EXP or EXPSPACE is also in P, the exponential class collapses to the polynomial time class P, which is impossible by the hierarchy theorems. Thus EXP-complete and EXPSPACE-complete problems are provably intractable.

4.6 Example Consider the following problems on equivalence of two regular expressions:

$$\begin{aligned} \mathrm{EQ}_{\mathrm{REX}} &:= \{\langle R, R' \rangle \mid R \text{ and } R' \text{ are equivalent regular expressions}\} \\ \mathrm{EQ}_{\mathrm{REX}^*} &:= \{\langle R, R' \rangle \mid R \text{ and } R' \text{ are equivalent star-free regular expressions}\} \\ \mathrm{EQ}_{\mathrm{REX}\cap} &:= \{\langle R, R' \rangle \mid R \text{ and } R' \text{ are equivalent regular expressions with intersection}\} \\ \mathrm{EQ}_{\mathrm{REX}\uparrow} &:= \{\langle R, R' \rangle \mid R \text{ and } R' \text{ are equivalent regular expressions with exponentiation}\} \end{aligned}$$

$\mathrm{EQ}_{\mathrm{REX}^*}$ is coNP-complete, $\mathrm{EQ}_{\mathrm{REX}}$ is PSPACE-complete, $\mathrm{EQ}_{\mathrm{REX}\cap}$ is EXP-complete, and $\mathrm{EQ}_{\mathrm{REX}\uparrow}$ is EXPSPACE-complete. Thus $\mathrm{EQ}_{\mathrm{REX}\cap}$ and $\mathrm{EQ}_{\mathrm{REX}\uparrow}$ are provably intractable.

Exercises for Section 4.1

1. Verify that $2n^2 + 3n + 4$, 2^n, 3^n and 5^{n^2} are space and time constructible functions of n.

2. It can be proved that $\mathrm{TIME}(f(n)) \subseteq \mathrm{SPACE}(f(n)/\log f(n))$ for a time and space constructible function $f(n)$. Conclude that $\mathrm{TIME}(n^k) \subsetneqq \mathrm{SPACE}(n^k)$ for every $k \in \mathbb{N}$. Does this imply that $\mathrm{P} \subsetneqq \mathrm{PSPACE}$?

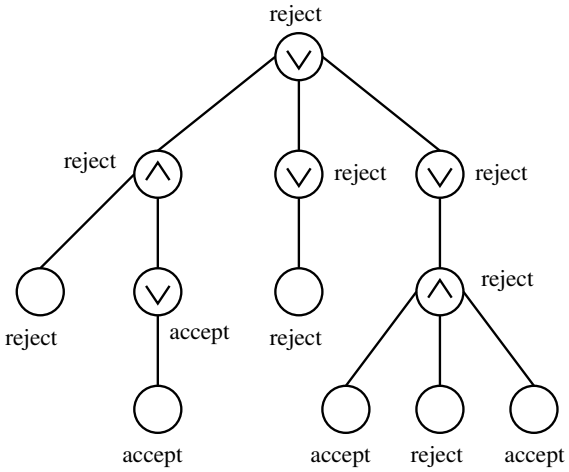
3. Prove that $\mathrm{NTIME}(n^k) \subsetneqq \mathrm{PSPACE}$ for every $k \in \mathbb{N}$. Does this imply that $\mathrm{NP} \subsetneqq \mathrm{PSPACE}$?

4. Let $f(n)$ be a time constructible function. We say that a language L is d e c i d a b l e in time $f(n)$, if L is the language of a TM that requires $\mathrm{O}(f(n))$ steps in every branch of computation (accepting or rejecting) on an input of size n. Let us also call a language L r e c o g n i z a b l e in time $f(n)$, if there exists a TM that accepts L with $\mathrm{O}(f(n))$ steps in every *accepting* branch of computation on an input of size n (The rejecting branches need not even halt!). Prove that if L is recognizable in $f(n)$ time, then it is decidable in $f(n)\log f(n)$ time. Conclude that L is recognizable in poly-time, if and only if it is decidable in poly-time.

4.2 Polynomial time hierarchy

Recall that an NTM accepts an input, if and only if there exists at least one accepting branch of computation. However, we could arrange that an NTM accepts if and only if all branches of computation accept. In fact, we may go more general so as to assign each node in the computation tree to be accepting or rejecting depending on the acceptability of 'some' or 'all' of its children. This concept is illustrated in Figure 4.1 which illustrates the computation tree for some nondeterministic computation. The leaf nodes are marked 'accept' or 'reject' depending on the corresponding configurations.

Figure 4.1: An alternating computation



If this computation were carried out in a usual NTM, the input would be accepted, since there exist accepting branches of computations (paths from the root to accepting leaves). However, in our example we have marked each non-leaf node by the OR ($\vee$) or the AND ($\wedge$) symbol. A non-leaf node marked with $\vee$ accepts, if *some* of its children is/are accepting, whereas a non-leaf node marked by $\wedge$ is accepting, if *all* of its children are accepting. Thus we may work upwards starting from the leaves and determining accept and reject status of the non-leaf nodes. The input is accepted, if and only if the root node is accepting. More formally, we define the following. We may assume without loss of generality that an NTM has only one accepting state q_a and only one rejecting state q_r. Moreover, whenever the machine enters one of these states, it accepts/rejects and halts.

4.7 Definition A l t e r n a t i n g T u r i n g m a c h i n e (A T M) is an NTM for which each state other than q_a and q_r is either an existential state or a universal state. The computation tree of the machine on an input has each non-leaf node marked by $\vee$ or $\wedge$ depending on whether the state in the configuration at that node is an existential or a universal state. The machine accepts the input based on the procedure mentioned just before this definition.

Thus ATMs are NTMs with a more general acceptance criterion. In other words, usual NTMs incorporate only the 'there exists' ($\exists$) logical structure. An ATM, on the other hand, can handle both 'there exists' and 'for all' ($\forall$) logical constructs. Moreover, these constructs may come in any number and in any order in the span of a computation. The question that now arises is if this new feature adds to the language deciding capabilities of NTMs. The precise answer is not known. However, based on the number of nondeterministic choices an ATM makes, we can define an interesting hierarchy.

Since an ATM is essentially an NTM, its time and space requirements can be defined in an analogous manner as we did for a usual NTM.

$$\begin{aligned}
\mathrm{ATIME}(f(n)) &:= \{L \mid L \text{ is decided by an } \mathrm{O}(f(n))\text{-time ATM}\},\\
\mathrm{ASPACE}(f(n)) &:= \{L \mid L \text{ is decided by an } \mathrm{O}(f(n))\text{-space ATM}\},\\
\mathrm{AP} &:= \bigcup_{k\in\mathbb{N}} \mathrm{ATIME}(n^k),\\
\mathrm{APSPACE} &:= \bigcup_{k\in\mathbb{N}} \mathrm{ASPACE}(n^k),\\
\mathrm{AL} &:= \mathrm{ASPACE}(\log n).
\end{aligned}$$

4.8 Example (1) Consider the language

$$\mathrm{TAUTOLOGY} := \{\langle\phi\rangle \mid \phi \text{ is a tautology}\},$$

where tautology means a Boolean expression that evaluates to true for every assignment of its variables. I claim that $\mathrm{TAUTOLOGY} \in \mathrm{AP}$. For the proof one may, given ϕ, universally generate all possible assignments of the variables, and evaluate ϕ at the assignment selected. If the evaluation result is 0, then the input is rejected, else it is accepted. All problems in coNP (like $\mathrm{TAUTOLOGY}$) can be shown to be in AP (See Exercise 4.2.1).

(2) Two Boolean formulas ϕ and ψ (on the same set of variables) are called equivalent, if they evaluate to the same value for every truth assignment of the variables. We measure the length of a Boolean formula by the number of symbols in it. A Boolean formula is called minimal, if it has no shorter equivalent. Define

$$\mathrm{MIN\text{-}FORMULA} := \{\langle\phi\rangle \mid \phi \text{ is a minimal Boolean formula}\}.$$

Note that ϕ is minimal, if and only if for every formula ψ, shorter than ϕ, there exists an assignment of the variables such that ϕ and ψ evaluate to different values. We have $\mathrm{MIN\text{-}FORMULA} \in \mathrm{AP}$, since one can universally select a shorter formula ψ, existentially select an assignment of the variables, and subsequently check if ϕ and ψ evaluate to different values under this assignment. $\mathrm{MIN\text{-}FORMULA}$ is not known to be in NP and coNP. Thus alternation appears to inject more power to our computing device.

The following theorem relates the alternating time and space classes with deterministic classes.

4.9 Theorem For any $f(n) \geqslant n$ one has:

$$\mathrm{ATIME}(f(n)) \subseteq \mathrm{SPACE}(f(n)) \subseteq \mathrm{ATIME}(f^2(n)),$$

whereas for any $f(n) \geqslant \log n$ one has:

$$\mathrm{ASPACE}(f(n)) = \mathrm{TIME}\left(2^{\mathrm{O}(f(n))}\right).$$

In particular, $\mathrm{AP} = \mathrm{PSPACE}$, $\mathrm{APSPACE} = \mathrm{EXP}$ and $\mathrm{AL} = \mathrm{P}$. (Note that neither of the similar equalities $\mathrm{NP} = \mathrm{PSPACE}$, $\mathrm{NPSPACE} = \mathrm{EXP}$ and $\mathrm{NL} = \mathrm{P}$ for NTMs could be proved till date.)

4.10 Definition Let $i \geqslant 0$ be an integer. A Σ_i- A T M is an ATM that makes at most i nondeterministic (existential or universal) choices in any branch of computation, with the first choice being existential. A Π_i- A T M is similar except that the first choice is universal. Let $\Sigma_i\mathrm{TIME}(f(n))$ and $\Pi_i\mathrm{TIME}(f(n))$ denote the classes of languages decided by $\mathrm{O}(f(n))$-time Σ_i- and Π_i-ATMs respectively. Define:

$$\begin{aligned}\Sigma_i\mathrm{P} &:= \bigcup_{k\in\mathbb{N}} \Sigma_i\mathrm{TIME}(n^k),\\ \Pi_i\mathrm{P} &:= \bigcup_{k\in\mathbb{N}} \Pi_i\mathrm{TIME}(n^k).\end{aligned}$$

The classes $\Sigma_i\mathrm{P}$ and $\Pi_i\mathrm{P}$, $i = 0, 1, 2, \ldots$, define a hierarchy of languages, called the p o l y n o m i a l t i m e h i e r a r c h y. Their union defines the class

$$\mathrm{PH} := \bigcup_{i\in\mathbb{Z}_+} \Sigma_i\mathrm{P} = \bigcup_{i\in\mathbb{Z}_+} \Pi_i\mathrm{P}.$$

The two unions in the definition of PH are the same, since one can always make dummy (unused) choices at the beginning of an algorithm.

We have $\mathrm{P} = \Sigma_0\mathrm{P} = \Pi_0\mathrm{P}$, $\mathrm{NP} = \Sigma_1\mathrm{P}$, $\mathrm{coNP} = \Pi_1\mathrm{P}$. We also have $\mathrm{MIN\text{-}FORMULA} \in \Pi_2\mathrm{P}$. By Theorem 4.9 $\mathrm{PH} \subseteq \mathrm{PSPACE}$. It is not known if this inclusion is proper (See Exercises 4.2.5, 4.2.6).

Exercises for Section 4.2

1. Demonstrate that $\mathrm{UNSAT} := \overline{\mathrm{SAT}} = \{\langle\phi\rangle \mid \phi \text{ is not satisfiable}\}$ is in AP. More generally, show that $\mathrm{coNP} \subseteq \mathrm{AP}$.

2. Give a poly-time alternating algorithm to decide the language

$$\mathrm{HALFCYCLE} := \{\langle G\rangle \mid \text{ The longest cycle in the directed graph } G \text{ is of length } \lfloor n(G)/2\rfloor\}.$$

3. Prove that $\mathrm{coAP} = \mathrm{AP}$, $\mathrm{coAL} = \mathrm{AL}$ and $\mathrm{coAPSPACE} = \mathrm{APSPACE}$. Prove also that $\mathrm{co}\Sigma_i\mathrm{P} = \Pi_i\mathrm{P}$.

*** 4.** Show that $\mathrm{P} = \mathrm{NP}$, if and only if $\mathrm{P} = \mathrm{PH}$.

5. Define complete problems in the class PH by usual poly-time reductions. Show that the existence of a PH-complete problem implies:

(a) There exists an $i_0 \in \mathbb{N}$ such that $\Sigma_i\mathrm{P} = \Sigma_{i_0}\mathrm{P}$ for all $i \geqslant i_0$.

(b) There exists a $j_0 \in \mathbb{N}$ such that $\Pi_j\mathrm{P} = \Pi_{j_0}\mathrm{P}$ for all $j \geqslant j_0$.

In particular, if $\mathrm{PH} = \mathrm{PSPACE}$, the polynomial hierarchy collapses to only finitely many different levels. (PSPACE has complete problems.) By Theorem 4.9 $\mathrm{PH} \subseteq \mathrm{PSPACE}$. It is not known whether these two classes are equal.

6. Recall that $\mathrm{AP} = \mathrm{PSPACE}$, whereas by the remarks in the last exercise we do not know whether $\mathrm{PH} = \mathrm{PSPACE}$. But then what is wrong in the following proof of the assertion $\mathrm{AP} = \mathrm{PH}$?

> We already know that $\mathrm{PH} \subseteq \mathrm{PSPACE} = \mathrm{AP}$. In order to prove the reverse inclusion, choose any $L \in \mathrm{AP}$. By definition L has a poly-time alternating algorithm. Suppose that this algorithm makes at most i nondeterministic choices. If the first choice is existential, we have $L \in \Sigma_i\mathrm{P}$; if it is universal, $L \in \Pi_i\mathrm{P}$. But $\mathrm{PH} = \bigcup_{i\in\mathbb{N}} \Sigma_i\mathrm{P} = \bigcup_{i\in\mathbb{N}} \Pi_i\mathrm{P}$. Thus in all cases we have $L \in \mathrm{PH}$.

4.3 Relativization

So far we have used the diagonalization method for locating undecidable and intractable problems. All these proofs are based on simulating other machines and reversing the decision of the simulated machine on some input after the simulation stops. Relativization provides strong evidence that such a simulation technique is unlikely to settle the $\mathrm{P} = \mathrm{NP}$? question.

We give a conventional TM (deterministic or nondeterministic) a divine power that comes in the form of a black-box. The black-box stands for a language L. Whenever the TM queries the black-box about a string α, it obtains in a single step of computation the answer whether $\alpha \in L$. How such a black-box can be implemented is not an issue here; it is treated as an *oracle*.[4.1] We take interest in the question if this 'free' information about L allows the TM to accept other languages more easily than before.

4.11 Definition An oracle is a language. An oracle Turing machine M^L is a TM with an oracle tape corresponding to the oracle L. Whenever M writes a string α on this tape, it gets the decision whether $\alpha \in L$ in a single computation step. We define P^L to be the class of problems solvable by poly-time TMs given the oracle for L. NP^L and coNP^L can be analogously defined.

4.12 Example The following two examples use the SAT oracle.

(1) $\mathrm{NP} \subseteq \mathrm{P}^{\mathrm{SAT}}$.

(2) NONMIN-FORMULA $:= \{\langle\phi\rangle \mid \phi$ is not a minimal Boolean formula$\}$ is in $\mathrm{NP}^{\mathrm{SAT}}$.

4.13 Theorem (1) There exists an oracle A for which $\mathrm{P}^A = \mathrm{NP}^A$.

(2) There exists an oracle B for which $\mathrm{P}^B \neq \mathrm{NP}^B$.

Assume that one could prove $\mathrm{P} = \mathrm{NP}$ or otherwise by a simulation argument using diagonalization. Both the simulating and the simulated machine could use the same oracle L, thereby proving the same relation about the relative classes P^L and NP^L. But that cannot be consistent with both the (provable) assertions of the previous theorem.

Exercises for Section 4.3

1. Show that:

(a) If $L \in \mathrm{P}$, then $\mathrm{P}^L = \mathrm{P}$.

(b) If $\mathrm{P}^L = \mathrm{NP}$ for some oracle L, then $\mathrm{NP} = \mathrm{coNP}$.

2. Prove that the language HALFCYCLE of Exercise 4.2.2 is in $\mathrm{NP}^{\mathrm{SAT}}$.

3. The concept of oracle TMs can be extended by giving the oracle the capability to answer in a single computation step a query about *any* arbitrary language in an entire complexity class $\mathcal{C}$. We define the classes $\mathrm{P}^{\mathcal{C}}$, $\mathrm{NP}^{\mathcal{C}}$, etc. relative to the oracle for the class $\mathcal{C}$. For example, P^{NP} stands for all problems that are solvable in deterministic polynomial time relative to the oracle for the class NP. More precisely, $\mathrm{P}^{\mathcal{C}} := \bigcup_{A \in \mathcal{C}} \mathrm{P}^A$, $\mathrm{NP}^{\mathcal{C}} := \bigcup_{A \in \mathcal{C}} \mathrm{NP}^A$, etc. Prove that:

(a) $\mathrm{P}^{\mathrm{NP}} = \mathrm{P}^{\mathrm{coNP}} = \mathrm{P}^{\mathrm{SAT}} = \mathrm{P}^{\mathrm{UNSAT}}$.

** **(b)** $\mathrm{NP}^{\mathrm{NP}} = \mathrm{NP}^{\mathrm{SAT}} = \Sigma_2\mathrm{P}$ and $\mathrm{coNP}^{\mathrm{NP}} = \Pi_2\mathrm{P}$. (Compare Exercises 4.2.2 and 4.3.2.)

(In general, it can be shown that $\Sigma_i\mathrm{P} = \mathrm{NP}^{\Sigma_{i-1}\mathrm{P}}$ and $\Pi_i\mathrm{P} = \mathrm{coNP}^{\Sigma_{i-1}\mathrm{P}}$ for every $i \geqslant 1$. These relations provide an alternate way of defining the polynomial hierarchy, namely, in terms of oracle TMs.)

[4.1] According to the Merriam-Webster online dictionary, an 'oracle' is a person through whom a deity (God) is believed to speak.