Logic and Computation I

Chapter 2. Propositional logic and computational complexity

Kazuyuki Tanaka

BIMSA

October 29, 2024

北京雁栖湖
应用数学研究院
Beijing Institute
of Mathematical
Sciences and Applications

Logic and Computation I

- **Part 1. Introduction to Theory of Computation**
- **Part 2. Propositional Logic and Computational Complexity**
- **Part 3. First Order Logic and Decision Problems**
- **Part 4. Modal logic**

Part 2. Schedule

- Oct.10, (1) Tautologies and proofs
- Oct.15, (2) The completeness theorem of propositional logic
- Oct.17, (3) SAT and NP-complete problems
- Oct.22, (4) NP-complete problems about graphs
- Oct.24, (5) Time-bound and space-bound complexity classes
- Oct.29, (6) Hierarchy theorems
- Oct.31, (7) PSPACE-completeness and TQBF

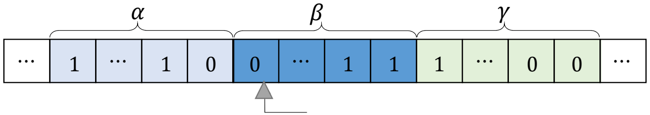
Recap

We assume that a Turing machine has one input tape and an arbitrary number of working tapes. We write $|x|$ for the length of the symbol string x.

- A TM runs in **time** $f(n)$ or is $f(n)$ **time(-bounded)**, if for every input x (except finitely many), all calculation processes end within $f(|x|)$ steps.
- A TM runs in **space** $f(n)$ or is $f(n)$ **space(-bounded)**, if for every input x (except finitely many), no calculation processes use more than $f(|x|)$ cells on each working tape.

The **Linear Speedup Theorem**

A language acceptable by a $f(n)$ time/space-bounded TM (det or non-det) is also acceptable by $\epsilon f(n)$ time/space-bounded TM, for any constant $\epsilon > 0$.



A number-theoretic function $f : \mathbb{N} \to \mathbb{N}$ used for bounding time and space is

- ▷ monotonically increasing,
- ▷ so simple (time-constructible and space-constructible) that it can be checked at any time during computation whether it is in the time or space bound.

By the latter condition, we may suppose that any computational process should halt within the time or space bound even if it is not accepted.

For $f, g : \mathbb{N} \to \mathbb{N}$, their growth rates (asymptotic behaviors) are compared as

- $f(n) = O(g(n)) \overset{\text{def}}{\Leftrightarrow}$ there exists some $c > 0$ and for sufficiently large n,
$$f(n) \leq c \cdot g(n).$$
- $f(n) = \Theta(g(n)) \overset{\text{def}}{\Leftrightarrow} f(n) = O(g(n))$ and $g(n) = O(f(n))$.

In the following, by "$O(f(n))$ time (space)", we mean "for some $g(n) = O(f(n))$, $g(n)$ time (space)".

Definition 2.33

For a function $f : \mathbb{N} \to \mathbb{N}$, we define the following four **complexity classes**.

$$\begin{aligned}
\mathrm{DTIME}(f(n)) &\overset{\text{def}}{=} \{L(M) \mid M \text{ is } O(f(n)) \text{ time deterministic TM}\},\\
\mathrm{NTIME}(f(n)) &\overset{\text{def}}{=} \{L(M) \mid M \text{ is } O(f(n)) \text{ time non-deterministic TM}\},\\
\mathrm{DSPACE}(f(n)) &\overset{\text{def}}{=} \{L(M) \mid M \text{ is } O(f(n)) \text{ space deterministic TM}\},\\
\mathrm{NSPACE}(f(n)) &\overset{\text{def}}{=} \{L(M) \mid M \text{ is } O(f(n)) \text{ space non-deterministic TM}\}.
\end{aligned}$$

By using $O(f(n))$ for bounding, we have a stable class that does not depend on the detailed definition of the Turing machine.

Definition 2.34 (Major Complexity Classes 1)

$$\begin{aligned}
\text{L (or LOGSPACE)} &\overset{\text{def}}{=} \mathrm{DSPACE}(\log n),\\
\text{NL (or NLOGSPACE)} &\overset{\text{def}}{=} \mathrm{NSPACE}(\log n),\\
\text{P} &\overset{\text{def}}{=} \mathrm{DTIME}(n^{O(1)}) = \bigcup_k \mathrm{DTIME}(n^k),\\
\text{NP} &\overset{\text{def}}{=} \mathrm{NTIME}(n^{O(1)}) = \bigcup_k \mathrm{NTIME}(n^k),\\
\text{PSPACE} &\overset{\text{def}}{=} \mathrm{DSPACE}(n^{O(1)}) = \bigcup_k \mathrm{DSPACE}(n^k),\\
\text{NPSPACE} &\overset{\text{def}}{=} \mathrm{NSPACE}(n^{O(1)}) = \bigcup_k \mathrm{NSPACE}(n^k).
\end{aligned}$$

Example 4

- The problem **STCon(nect)** is to determine whether there is a path from a vertex s to a vertex t in a directed graph G.
- Non-deterministic space complexity: $\mathrm{STCon} \in$ NL.
    - Let n be the number of vertices of G. Extend a path from s non-deterministically, and accept it when it reaches t. Because it does not need to record the history, $O(\log n)$ space is enough for keeping the information on the current vertex, the next one you will visit, and the number of steps you have taken.
- The above NL algorithm can been roughly seen as a non-deterministic linear time one.
- For deterministic case, STCon is in P and $\mathrm{DSPACE}(\log^2 n)$, which is shown by Theorem 2.36 and 2.37 (Savitch's Theorem).

Theorem 2.35

For any space-constructible function $S(n) \geq \log n$,

$$\begin{aligned} \mathrm{DSPACE}(S(n)) &\subseteq \mathrm{DTIME}(2^{O(S(n))}) = \bigcup_k \mathrm{DTIME}(2^{kS(n)}), \\ \mathrm{NSPACE}(S(n)) &\subseteq \mathrm{NTIME}(2^{O(S(n))}) = \bigcup_k \mathrm{NTIME}(2^{kS(n)}). \end{aligned}$$

In particular, L $\subseteq$ P, and NL $\subseteq$ NP.

Proof idea.

- Let M be a machine running in space $S(n)$. Then, the number of its configurations is $\leq |Q|nS(n)|\Omega|^{S(n)} \leq c^{S(n)}$ for a constant c. So, a computational process longer than $c^{S(n)}$ includes a repetition of the same configuration. Hence, we may only consider computational time shorter than $c^{S(n)}$, i.e., $2^{O(S(n))}$.

Theorem 2.36

For any $T(n)$ and $S(n) \geq \log n$,

$$\begin{array}{rcll} \mathrm{NTIME}(T(n)) & \subseteq & \mathrm{DSPACE}(T(n)), & (\diamondsuit) \\ \mathrm{NSPACE}(S(n)) & \subseteq & \mathrm{DTIME}(2^{O(S(n))}) & (\clubsuit) \end{array}$$

In particular, NP $\subseteq$ PSPACE, and NL $\subseteq$ P.

Proof.

$(\diamondsuit)$ Given a $T(n)$-time NTM M, let a DTM M' perform a depth-first search on its computation tree. To exhaust all path searches, M' does not need to remember the history of configurations of M, but it is sufficient to memorize that of transition branches, which can be done in $T(n)$ space. Also, for the simulation of M, $T(n)$ space is enough.

$(\clubsuit)$ Given a $S(n)$-space NTM M, let a DTM M' enumerate reachable configs. of M by some deterministic way (e.g., width-first) until a final config. is reached. Note that when you rewrite a list of $2^{O(S(n))}$ configurations $2^{O(S(n))}$ times, the computation steps are $2^{O(S(n))} \cdot 2^{O(S(n))} = 2^{O(S(n))}$ (cf. Theorem 2.35).

Theorem 2.37 (Savitch's theorem)

For any $S(n) \geq \log n$,

$$\mathrm{NSPACE}(S(n)) \subseteq \mathrm{DSPACE}(S(n)^2).$$

In particular, PSPACE = NPSPACE, and EXPSPACE = NEXPSPACE.
By Example 4, $\mathrm{STCon} \in \mathrm{DSPACE}(\log^2(n))$.

Proof.

- By the proof of (♣) in Theorem 2.36, for a $S(n)$-space NTM M , there exists some constant c such that M can be mimicked by a $c^{S(n)}$-time DTM. This simulation needs $c^{S(n)}$ space, but can be improved as below.
- By $\mathrm{Reach}(\alpha, \beta, k)$, we mean the existence of a transition from a configuration α to a configuration β within $k(\leq c^{S(n)})$ steps in a $S(n)$-space NTM M.

Proof(continued). $\mathrm{Reach}(\alpha, \beta, k)$ is determined recursively as follows.

- ▷ If $k = 0$, check whether $\alpha = \beta$.
- ▷ If $k = 1$, check whether it can move from α to β in one step.
- ▷ If $k \geq 2$, check whether there is a computational configuration γ that satisfies both $\mathrm{Reach}\left(\alpha, \gamma, \frac{k}{2}\right)$ and $\mathrm{Reach}\left(\gamma, \beta, \frac{k}{2}\right)$. If so, $\mathrm{Reach}(\alpha, \beta, k)$ also holds. If $\frac{k}{2}$ is not an integer, one side is rounded up and the other rounded down.
- ▷ For $\frac{k}{2} \geq 2$, first seek a γ' s.t. $\mathrm{Reach}\left(\alpha, \gamma', \frac{k}{2^2}\right)$ and $\mathrm{Reach}\left(\gamma', \gamma, \frac{k}{2^2}\right)$. Later, seek a γ' s.t. $\mathrm{Reach}\left(\gamma, \gamma', \frac{k}{2^2}\right)$ and $\mathrm{Reach}\left(\gamma', \beta, \frac{k}{2^2}\right)$.

By repeating this recursive process, we obtain a binary tree whose height is about $\log_2 k = O(S(n))$. In each branching, $O(S(n))$ space is necessary to memorize the configuration. In a total pass, it can be executed in $O(S(n)^2)$ space.

□

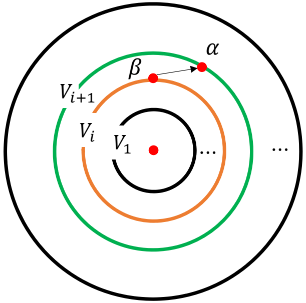
Theorem 2.38 (Immermann- Szelepcsényi theorem)

For any $S(n)(\geq \log n)$, $\mathrm{NSPACE}(S(n))$ is closed under complement.

Proof.

- Suppose a NTM M accepts a language A with $S(n)$ space. We will construct a NTM $\overline{M}$ that accepts the complement A^c with $S(n)$ space.
- A configuration of M can be represented by a string of length $S(n)$, and so the total number is $c^{S(n)}$ for some constant c.
- Consider the directed graph G with the configurations as vertices and the transition relations as directed edges. It is sufficient to determine whether there is a path from the initial state to a final state in the graph G. We may assume that M has a unique accepting configuration, by making M erase its worktape and return its heads to the starting positions after the computation.
- To get the answer Yes for the existence of such a path, it is nondeterministically computable in $\log(c^{S(n)}) = O(S(n))$ space as shown in Example 4 (STCon).

- To get No, it seems to require a huge amount of spaces to examine all possible paths, but there is a surprisingly simple process.
- Let V_i be the set of vertices reachable within i steps from the initial configuration. Let $m(i) = |V_i|$. By counting $m(i)$, we can check whether all paths are examined.
- To compute $m(i+1)$, we check whether $\alpha \in V_{i+1}$ for each vertex α in some order. If yes, increment a counter by one. The final value of the counter is $m(i+1)$.



- To check $\alpha \in V_{i+1}$, non-deterministically choose an element β of V_i ($m(i)$ times) and check if there is an edge from β to α or $\alpha = \beta$.
- Finally, for some $i \le c^{S(n)}$, $m(i+1) = m(i)$. If the final configuration does not appear before i, then M will not accept the input, so $\overline{M}$ will accept it. $\square$

Since $\mathrm{NSPACE}(n)$ matches the class of context-sensitive languages, this also solved the long-standing open question: Is the complement of a context-sensitive language is still context-sensitive?

Highlights of the above results are:

$$\mathsf{PSPACE} = \mathsf{NPSPACE}, \qquad \mathsf{EXPSPACE} = \mathsf{NEXPSPACE}$$

and

$$\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP} \subseteq \mathsf{NEXP} \subseteq \mathsf{EXPSPACE}$$

Among them, topic on which is/are a proper inclusion relation will be discussed in the last half of this lecture.

§2.6. Hierarchy theorems

- We will show that some complexity classes are not equivalent, that is, the existence of a hierarchy of complexity classes.
- As in the previous lectures, $T(n)$ is time-constructible with $T(n) > n$, and $S(n)$ is space-constructible with $S(n) \geq \log n$.

Theorem 2.39 (Space Hierarchy Theorem)

Let $S(n) \geq \log n$ be space constructible. Then for any $S'(n) = o(S(n))$, there exists a problem in $\mathrm{DSPACE}(S(n))$ but not in $\mathrm{DSPACE}(S'(n))$.

Proof.

- Prove by a diagonalization argument.
- Let $M_0, M_1, \ldots$ enumerate the deterministic Turing machines with alphabet $\{0, 1\}$.

Proof (continued)

- For a binary string $x \in \{0,1\}^*$, let $\sharp(x)$ be the natural number represented by x as the binary representation ignoring 0's at its head. Therefore, for any natural number i, there exists an arbitrarily long sequence x such that $\sharp(x) = i$.

  Now, construct a machine M with $O(S(n))$ space that cannot be imitated in the $o(S(n))$ space. For a binary string x of length n,

1. Mark $S(n)$ cells on the working tape ($\because S(n)$ is constructible),
2. If $i = \sharp(x)$, imitate M_i with input x in space $S(n)$.
3. By the imitation, if M is going to run over space $S(n)$, it stops and accepts x.
4. As in the proof of theorem 2.35, if M_i runs for a sufficiently long time $2^{kS(n)}$, it is already in a roop. So, M stops and accepts x (in $O(S(n))$ space).
5. If M_i accepts/rejects x in space $S(n)$, then M rejects/accepts x (respectively).

This machine M operates in $O(S(n))$ space.

Proof (continued)

- By way of contradiction, we assume there exists an M_i mimicking M in space $S'(|x|) = o(S(n))$.
- Consider a sufficiently long input x ($S'(|x|) < S(|x|)$) such that $\sharp(x) = i$.
- By the definition of M, M and M_i give different results for input x in space $S(|x|)$, which is a contradiction. $\square$

Note that $S'(n)$ is not assumed to be space constructible in the theorem.

Theorem 2.40 (Time Hierarchy Theorem)

Let $T(n)$ be time constructible and $T(n) > n$. For any $T'(n)$ such that

$$T'(n)\log T'(n) = o(T(n)),$$

there exists a problem in $\mathrm{DTIME}(T(n))$ but not in $\mathrm{DTIME}(T'(n))$.

This proof is similar to that of Space Hierarchy Theorem. Note that a universal machine for the $T'(n)$-time machines operates in time $O(T'(n)\log T'(n))$.

Exercise 2.6.1

Prove the Time Hierarchy Theorem.

From the above hierarchy theorems, we have the following.

$$\mathrm{L} \subsetneq \mathrm{PSPACE} \subsetneq \mathrm{EXPSPACE}, \quad \mathrm{P} \subsetneq \mathrm{EXP}.$$

For nondeterministic classes, we also have hierarchy theorems. Since their arguments are much more complex, we only state two major theorems and key ideas of the poofs.

Theorem 2.41 (Ibarra, 1972)

For any real number $r > s \geq 1$,

$$\mathrm{NSPACE}(n^s) \subsetneq \mathrm{NSPACE}(n^r).$$

Proof by contradiction. For instance, suppose $\mathrm{NSPACE}(n^4) = \mathrm{NSPACE}(n^3)$. Then by the padding method (we will show in the next slide), we also have $\mathrm{NSPACE}(n^5) = \mathrm{NSPACE}(n^4)$ and then $\mathrm{NSPACE}(n^6) = \mathrm{NSPACE}(n^5)$, etc. Thus, $\mathrm{NSPACE}(n^7) = \mathrm{NSPACE}(n^3)$. However,

$$\begin{aligned} \mathrm{NSPACE}(n^3) &\subseteq \mathrm{DSPACE}(n^6) \text{(from Savitch's theorem)} \\ &\subsetneq \mathrm{DSPACE}(n^7) \text{(from the space hierarchy theorem)} \\ &\subseteq \mathrm{NSPACE}(n^7) \end{aligned}$$

which is a contradiction.

The padding method

.

Let A be a language accepted by an n^5 space NTM M. Then, let

$$A' = \{x\,\sharp^m \mid x \in A,\ |x|^5 = |x\,\sharp^m|^4\},$$

where $\sharp$ is a new symbol not belonging to M
Define a NTM M' to operate on input $x\,\sharp^m$ as follows.

(i) Check if $|x|^5 = |x\,\sharp^m|^4$. If no, reject.

(ii) If Yes then mimic M's moves on input x

Since M operates on x in space $|x|^5$, M' operates in space $|x|^5 = |x\sharp^m|^4$, and so $A' = L(M') \in \mathrm{NSPACE}(n^4)$. Hence also $A \in \mathrm{NSPACE}(n^4)$
Further generalizations are left to the audience.

Theorem 2.42 (Cook 1973)

For any real number $r > s \geq 1$,

$$\mathrm{NTIME}(n^s) \subsetneq \mathrm{NTIME}(n^r).$$

The proof is more cumbersome because there is no counterpart of Savitch's theorem for time complexity classes. It uses a technique of so-called lazy diagonalization.

Summary

As already mentioned,

$$\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP} \subseteq \mathrm{NEXP} \subseteq \mathrm{EXPSPACE}.$$

From today's theorems,

$$\mathrm{NL} \subsetneq \mathrm{PSPACE}, \quad \mathrm{P} \subsetneq \mathrm{EXP}, \quad \mathrm{NP} \subsetneq \mathrm{NEXP}, \quad \mathrm{PSPACE} \subsetneq \mathrm{EXPSPACE}.$$

For each of the above four relations, there are two complexity classes are sandwiched between them. E.g., P and NP are sandwiched between $\mathrm{NL} \subsetneq \mathrm{PSPACE}$. So, in such consecutive four classes, at lease one of adjacent pairs must be a proper inclusion. However, it is widely open to decide which pair is proper.

Further readings

D.C. Kozen, *Theory of Computation*, Springer, 2006.

Thank you for your attention!