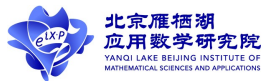
Logic and Computation: I

Chapter 2 Propositional logic and computational complexity

Kazuyuki Tanaka

BIMSA

December 6, 2022



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 2. Schedule

- Nov.17, (1) Tautologies and proofs
- Nov.22, (2) The completeness theorem of propositional logic
- Nov.24, (3) SAT and NP-complete problems
- Nov.29, (4) NP-complete problems about graph
- Dec. 1, (5) Time-bound and space-bound complexity classes
- Dec. 6, (6) PSPACE-completeness and TQBF

PSPACE-completeness and TQBF

Recap

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

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

- For major functions f, we have

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

- Savitch's theorem: for any $S(n) \geq \log n$, $\mathrm{NSPACE}(S(n)) \subseteq \mathrm{DSPACE}(S(n)^2)$.
- Immermann-Szelepcsényi's theorem: for any $S(n)(\geq \log n)$, $\mathrm{NSPACE}(S(n))$ is closed under complement.

Hierarchy theorems

- In the first half of this lecture, 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 (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 $\{0, 1\}$.
- 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 $\sharp(x) = i$.

Proof (continued) 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 (1) of the last lecture, if M_i runs for a sufficiently long time $2^{kS(n)}$, it is already in a roop. So, M stops and accepts x.
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.

- By way of contradiction, we assume there exists an M_i mimicking M in space $o(S(n))$.
- 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. □

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

Theorem (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 $O(T'(n) \log T'(n))$.

Homework

Prove the Time Hierarchy Theorem.

From the above hierarchy theorems, we have the following.

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

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 (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 (due to Cook), 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.

Theorem (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.

As already mentioned,

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

From today's theorems,

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

For each of the above four relations, there are two complexity classes are sandwiched between them. E.g., P and NP are sandwiched between $\mathsf{NL} \subsetneq \mathsf{PSPACE}$. 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.

PSPACE and TQBF

- PSPACE, along with NP, is a complexity class that captures many natural decision problems.
- R. Karp introduced the notion of PSPACE completeness as well as NP completeness.
- A **PSPACE complete problem** is a PSPACE problem that any PSPACE problem is polynomial-time reducible to it.
- Let TQBF (true quantified Boolean formula) denote the problem to decide whether a QBF (quantified Boolean formula) is true.
- Next we will introduce Stockmeyer et al.'s result on the PSPACE-completeness of TQBF .

- QBF(quantified Boolean formula) can be regarded as the first-order theory of the simple Boolean algebra $\mathbf{2} = \{0, 1\}$.
- When we discussed SAT as a NP-complete problem, we should have taken care to handle subscripts for variables $x_0, x_1, x_2, ...$, but PSPACE is a much broader class so that we may ignore such a coding issue.
- Recall that a Boolean expression is built from variables $x_0, x_1, x_2, ...$ and constants $0, 1$ by operations $\wedge, \vee, \neg$ as usual.
- A **QBF** (quantified Boolean formula) is built similarly, in addition, by the quantifiers $\forall, \exists$. $\exists x A(x)$ is equivalent to $A(0) \vee A(1)$ and $\forall x A(x)$ is $A(0) \wedge A(1)$.
- The **TQBF** problem is the decision problem of whether a given QBF is true (1), which is also called simply QBF in some books.

Example 1: An equivalent transformation of QBF

$$\forall x \exists y A(x, y) \leftrightarrow \exists y A(0, y) \wedge \exists y A(1, y) \leftrightarrow (A(0, 0) \vee A(0, 1)) \wedge (A(1, 0) \vee A(1, 1)).$$

- Any QBF can be transformed into a prenex normal form (i.e., a boolean expression prefixed by a sequence of quantifiers) by the following rule.
$(\exists y A) \wedge B \Rightarrow \exists y(A \wedge B)$, $(\exists y A) \vee B \Rightarrow \exists y(A \vee B)$ (B does not contain y freely),
$\neg\exists x A \Rightarrow \forall x \neg A$, $\neg\forall x A \Rightarrow \exists x \neg A$.

- Note that the elimination of quantifiers, as in Example 1, increases the length of the expression exponentially, whereas the transformation to the prenex normal form does not change the length of the expression.

- TQBF is to decide the truth value of a formula A in the form $Q_1 x_1 Q_2 x_2 ... Q_n x_n B(x_1, ..., x_n)$ (where $B(x_1, ..., x_n)$ is a Boolean expression).

- Now, if all Q_i are $\exists$, then
A is true $\Leftrightarrow$ B is satisfiable.

- Therefore, TQBF includes SAT as a special case.

- If all Q_i are $\forall$, then
A is true $\Leftrightarrow$ B is valid (tautology).

Theorem

TQBF is PSPACE-complete.

Proof.

- First, we show that TQBF is a PSPACE problem.
- For simplicity, consider the prenex normal form $A \equiv \forall x_1 \exists x_2 \forall x_3 B(x_1, x_2, x_3)$ ($B(x_1, x_2, x_3)$ is a Boolean expression).
- To find the value of A by substituting $0, 1$ for the variables in $B(x_1, x_2, x_3)$ appropriately, we need to memorize the current assignment for x_1, x_2, x_3 during the computation. So, this can be performed in the space of the length of a given formula.

By $B(b_1^a, b_2^e, b_3^a)$, we denote an expression obtained from $B(x_1, x_2, x_3)$ by substituting (b_1^a, b_2^e, b_3^a) for x_1, x_2, x_3, respectively. Here, a value substituted for a variable of $\forall$ is denoted as b^a, and the value for $\exists$ is denoted as b^e.

- The computation proceeds as follows.
  1. Examine $B(0^a, 0^e, 0^a)$. If true (value 1), check $B(0^a, 0^e, 1^a)$. If both are true, go to 3; otherwise, go to 2.
  2. Examine $B(0^a, 1^e, 0^a)$. If true, check $B(0^a, 1^e, 1^a)$. If both are true, go to 3; otherwise, reject (A is false).
  3. Examine $B(1^a, 0^e, 0^a)$. If true, check $B(1^a, 0^e, 1^a)$. If both are true, accept (A is true); otherwise, go to 4.
  4. Examine $B(1^a, 1^e, 0^a)$. If true, check $B(1^a, 1^e, 1^a)$. If both are true, accept (A is true); otherwise, reject (A is false).

- It is easy to generalize the above computation to any QBF. This is a computation in DSPACE(n).

- Next we show that TQBF is PSPACE-hard.
- Let L be a PSPACE problem and M a deterministic machine that accepts L in $S(n)$ space.
- Recall that the NP-hard proof of Cook theorem. Let $\varphi(t, \alpha)$ represent that α is the computational configuration at time t. More precisely, $\varphi(t, \alpha)$ is expressed as a conjunction of variables $x_{t,i,a}$, which represents that the ith ($\leq p(n)$) symbol in the computation configuration at time $t \leq p(n)$ is a. The transition of the configuration is represented by the relation of formula $\varphi(t, \_)$ and $\varphi(t+1, \_)$.
- Then, we can express that M accepts input w in polynomial time $p(n)$ by a Boolean expression Φ_w of length about $p(|w|)^3$.
- Since $S(n)$-space is converted to $2^{O(S(n))}$-time, the size of Φ_w is also $2^{O(S(n))}$. It is not polynomial-time reduction.
- This can be improved by the same trick as used for Savitch's theorem. Let $\mathrm{Reach}(\alpha, \beta, t)$ be a formula expressing that configuration β can be reached from configuration α within t steps. More precisely, a configuration is a sequence of variables $x_{i,a}$ ($i < S(n)$), and so there are $2S(n)$ variables in $\mathrm{Reach}(\alpha, \beta, t)$. Then we will express the computation as the relation between $\mathrm{Reach}(\_, \_, t)$ and $\mathrm{Reach}(\_, \_, t/2)$.

- Definition of $\mathrm{Reach}(\alpha, \beta, t)$.
- $\mathrm{Reach}(\alpha, \beta, 0)$ is equivalent to $\alpha = \beta$, which is more precisely a conjunction of equalities between variables. Let $\mathrm{Reach}(\alpha, \beta, 1)$ denotes that β is reachable from α in one step. The length of this formula is $O(S(n)^2)$.
- For $t \geq 2$, if $\mathrm{Reach}(\alpha, \beta, t)$ is defined as $\exists\gamma(\mathrm{Reach}(\alpha, \gamma, t/2) \wedge \mathrm{Reach}(\gamma, \beta, t/2))$, the size of the final formula is about size t, that is, $2^{S(n)}$.
- By using $\forall, \exists$, it is also recursively defined as

$$\exists\gamma\forall\delta_1\forall\delta_2((\delta_1 = \alpha \wedge \delta_2 = \gamma) \vee (\delta_1 = \gamma \wedge \delta_2 = \beta) \to \mathrm{Reach}(\delta_1, \delta_2, t/2)).$$

  Without the recursive part, the length is $O(S(n))$.
- To reach $\mathrm{Reach}(\_, \_, 1)$, we repeat the recursion $S(n)$ times, and so the final length is $O(S(n)^2)$.
- Thus $L \leq_p \mathrm{TQBF}$, and TQBF is PSPACE complete. □

Homework

Prove $\mathrm{DSPACE}(n) \neq \mathsf{P}$ and $\mathrm{DSPACE}(n) \neq \mathsf{NP}$.

Summary

- As already mentioned,

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

  From today's theorems,

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

- TQBF is PSPACE-complete.

Further readings

M. Sipser, *Introduction to the Theory of Computation*, 3rd ed., Course Technology, 2012.

Thank you for your attention!