Logic and Computation I

Chapter 2. Propositional logic and computational complexity

Kazuyuki Tanaka

BIMSA

October 31, 2024

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, (5-2) Relations between Time and space complexity classes
- Oct.31, (6) Hierarchy theorems, (7) PSPACE-completeness and TQBF

Recap

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

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

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

§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.

§2.7. PSPACE and TQBF

- PSPACE, along with NP, captures many natural decision problems.
- R. Karp introduced the notion of PSPACE completeness.
  A problem is said to be PSPACE **complete**, if it is PSPACE and any PSPACE problem is polynomial-time reducible to it.
- A **QBF** (quantified Boolean formula) is built from variables $x_0, x_1, x_2, \ldots$ and constants $0, 1$ by way of operations $\wedge, \vee, \neg$ and quantifiers $\forall, \exists$, where $\exists x A(x)$ is equivalent to $A(0) \vee A(1)$, and $\forall x A(x)$ is $A(0) \wedge A(1)$.
- **TQBF** (true QBF) denotes a problem to decide whether a QBF (quantified Boolean formula) is true. It is also called simply QBF in some books.
- Stockmeyer et al. show that TQBF is PSPACE-complete.

- A QBF can be seen as a first-order formula about the simple Boolean algebra $\mathbf{2} = \{0, 1\}$.

Example 5: An equivalent transformation of QBF

$$\begin{aligned}\forall x \exists y A(x, y) &\leftrightarrow \exists y A(0, y) \land \exists y A(1, y) \\ &\leftrightarrow (A(0,0) \lor A(0,1)) \land (A(1,0) \lor A(1,1)).\end{aligned}$$

- Any QBF can be transformed into a prenex normal form (i.e., a boolean expression prefixed by a sequence of quantifiers) by the following rules:

$$\begin{aligned}&(\exists y A) \land B \Rightarrow \exists y(A \land B),\ (\exists y A) \lor B \Rightarrow \exists y(A \lor B) \quad (y \text{ is not free in } B),\\ &(\forall y A) \land B \Rightarrow \forall y(A \land B),\ (\forall y A) \lor B \Rightarrow \forall y(A \lor B) \quad (y \text{ is not free in } B),\\ &\neg\exists x A \Rightarrow \forall x \neg A, \quad \neg\forall x A \Rightarrow \exists x \neg A.\end{aligned}$$

- Note that the elimination of quantifiers, as in Example 5, increases the length of the expression exponentially, whereas the transformation to the prenex normal form does not change the length of the expression.

- We consider **TQBF** as a problem to decide the truth value of a formula A in the form $Q_1 x_1 ... Q_n x_n B(x_1, ..., x_n)$ (with a Boolean expression $B(x_1, ..., x_n)$).
- 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).
- 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.

Theorem 2.43

TQBF is PSPACE-complete.

Proof.

- First, we show that TQBF is a PSPACE problem.

TQBF is PSPACE

- 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.

Proof continued

To evaluate $\forall x_1 \exists x_2 \forall x_3 B(x_1, x_2, x_3)$, 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).

TQBF is PSPACE-hard

To 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 proof of NP-hardness in Cook's theorem.

- Let $\varphi(t, \alpha)$ represent that α is M's 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 i-th $(\leq p(n))$ symbol in the computation configuration at time $t(\leq p(n))$ is a.
- The transition of the configuration is represented by the relation between $\varphi(t, \_)$ and $\varphi(t+1, \_)$.
- Then, we can express that M accepts an 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.

Proof continued

Let $\mathrm{Reach}(\alpha, \beta, t)$ be a formula expressing that configuration β can be reached from configuration α within t steps. Since a configuration is a sequence of variables $x_{i,a}$ $(i < S(n))$, there are $2S(n)$ variables in $\mathrm{Reach}(\alpha, \beta, t)$.

$\mathrm{Reach}(\alpha, \beta, t)$ is defined recursively as follows.

- $\mathrm{Reach}(\alpha, \beta, 0)$ 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 $t \geq 2$, $\mathrm{Reach}(\alpha, \beta, t) \equiv \exists\gamma(\mathrm{Reach}(\alpha, \gamma, t/2) \land \mathrm{Reach}(\gamma, \beta, t/2))$ would make the size of the final formula about $2^{\log(t)} S(n) \approx 2^{S(n)}$, since γ indeed consists of $S(n)$ variables.
- Instead, we define it by using $\forall, \exists$ as

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

Then, the length of the final formula is about $kS(n)\log(t)$, that is, $O(S(n)^2)$.

Thus $L \leq_p \mathrm{TQBF}$, and TQBF is PSPACE complete.

Exercise 2.7.1

Prove $\mathrm{DSPACE}(n) \neq$ P and $\mathrm{DSPACE}(n) \neq$ 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!