Logic and Computation I

Part 1. Introduction to theory of computation

Kazuyuki Tanaka

Beijing Institute of Mathematical Sciences and Applications (BIMSA)

Sep. 10, 2024

The aim of this course is to gain a broader view on logic and computation, and explore the dynamic interaction between them.

- Topics presented in this semester include: Computable functions, undecidability, propositional logic, NP-completeness, first-order logic, Gödel's completeness theorem, Gödel's incompleteness theorems, modal logic and its decidability.
- Topics in the second semester: Second-order logic, modal μ-calculus, infinite automata, determinacy of infinite games, admissible recursion, etc.

Historical Introduction

- At the beginning of the 20th century, D. Hilbert emphasized the importance of first-order logic as a general framework of mathematics. He then advocates "the decision problem (for validity or satisfiability of first-order formulas) must be considered the main problem of mathematical logic".
- Hilbert's aim was soon upset by K. Gödel, A. Church and A. Turing by developing mathematics of symbolic manipulation.
- In particular, Turing's mathematical model of symbolic computation, now known as Turing machine, had a great influence on the birth of computers, and is still used as a theoretical platform for algorithm analysis.
- There is no boundary between logic and computation. Let us explore their dynamic interaction.


D. Hilbert


K. Gödel


A. Church


A. Turing

Outline of the Course

1. This is a two-semester course in **mathematical logic** and **theory of computation**. In this semester, it covers the basic topics of the two fields and their important interactions so that advanced undergraduates can participate without any particular knowledge. Then in the next semester, we discuss more advanced topics emphasizing on decidability and definability.
2. We meet at 15:20-16:55, every Tuesday and Thursday in room A3-2-301, BIMSA, though the lectures are also offered online.
3. TA's are Dr. W.Li (chief) and Mr. K.Duo. They will handle homeworks, questions and comments from students via WeChat. We will not assign homeworks regularly in this semester, but motivated students are encouraged to solve many problems given in the lectures and submit your solutions to us.
4. Lecture slides will be uploaded on the course announcement page at BIMSA. More information will be given at our WeChat group page.

Introducing myself

Kazuyuki Tanaka

My specialty is logic, especially theory of definabilty and computability. I have mainly contributed to second-order arithmetic and reverse mathematics, and supervised fifteen doctoral students in this area. See https://sendailogic.com/tanaka/.

Education

- ⋆ Tokyo Institute of Technology
  Information Science, Bachelor, Master
- ⋆ University of California, Berkeley
  Mathematics,
  Ph.D. (Advisor: Leo Harrington)

Teaching Jobs

- ⋆ 1986 $\sim$ 1991, Tokyo Inst. Tech.
  Assistant Professor, Dept. of Info. Sci.;
  Visiting PennState.
- ⋆ 1991 $\sim$ 1997, Tohoku University
  Associate Professor, Dept. of Math.;
  Visiting Oxford.
- ⋆ 1997 $\sim$ 2022, Tohoku University
  Professor, Math Institute.
- ⋆ 2022 $\sim$ now, BIMSA, Professor.

Logic and Computation I (Syllabus)

- **Part 1. Introduction to Theory of Computation**
  Fundamentals on theory of computation and computability theory (recursion theory) of mathematical logic, as well as the connection between them. This part is the basis for the following lectures.
- **Part 2. Propositional Logic and Computational Complexity**
  The basics of propostional logic (Boolean algebra) and complexity theory including some classical results, such as the Cook-Levin theorem.
- **Part 3. First Order Logic and Decision Problems**
  The basics of first-order logic, Gödel's completeness theorem, Gödel's incompleteness theorems and the Ehrenfeucht-Fraïssé theorem and Lindström's theorem.
- **Part 4. Modal logic**
  Kripke models, canonical modal logics, standard translation, bisimulation, decidability results, and epistemic logic.
  In "Logic and Computation II", second-order logic and modal μ-calculus.

Part 1. Schedule

- Sep.10, (1) Automata and monoids
- Sep.12, (2) Turing machines
- Sep.17, a holiday
- Sep.19, (3) Computable functions and primitive recursive functions
- Sep.24, (4) Decidability and undecidability
- Sep.26, (5) Partial recursive functions and computable enumerable sets
- Oct. 1 and 3, holidays
- Oct. 8, (6) Rice's theorem and many-one reducibility

§1.1 Automata and Monoids

Words and languages

- A (finite) automaton is a simplest computing machine with finitely many states. Other computing machine such as a Turing machine can be regarded as a functionally-expanded automaton.
- Let Ω be a finite set of symbols. By a **word** over Ω, we mean a finite sequence of symbols from Ω. Then by Ω^n, we denote the set of words with length n. And put
$$\Omega^* = \bigcup_{i \geq 0} \Omega^i.$$
For instance, $\{0,1\}^2 = \{00, 01, 10, 11\}$, $\{0,1\}^0 = \{\varepsilon\}$ with ε an empty word. By a **language** in Ω, we mean a subset of Ω^*.
- In automata theory, we study the class of languages (of words) accepted by automata. In theory of computation, larger classes of languages are also defined and studied for many kinds of functionally-expanded machines.

Deterministic finite automaton

We first introduce *deterministic* finite automata. Later, we also define *non-deterministic* ones, and then show that the two types of automata have the same power of computation.

Definition 1.1

A **deterministic finite automaton** (DFA) is a 5-tuple $\mathcal{M} = (Q, \Omega, \delta, q_0, F)$,

(1) Q is a non-empty finite set, whose elements are called **states**.

(2) Ω is a non-empty finite set, whose elements are called **symbols**.

(3) $\delta : Q \times \Omega \to Q$ is a **transition function**.

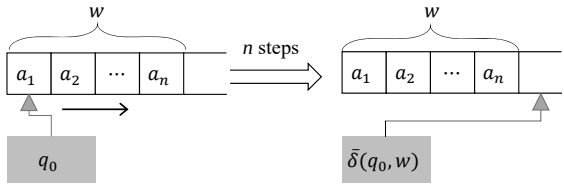
(4) $q_0 \in Q$ is an **initial state**.

(5) $F \subset Q$ is a set of **final states**.

Language accepted by DFA

- $\mathcal{M}$ reads a symbol on the input tape under the head, and it changes its state according to δ and moves the head to the right next symbol.
- For convenience, we extend δ to $\bar{\delta} : Q \times \Omega^* \to Q$ inductively as follows:

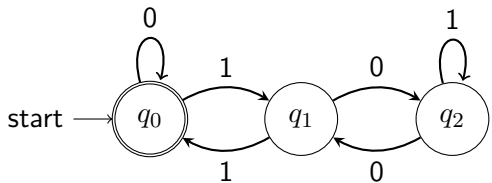
$$\begin{cases} \bar{\delta}(q, \varepsilon) = q, \\ \bar{\delta}(q, aw) = \bar{\delta}(\delta(q, a), w) \quad (a \in \Omega,\ w \in \Omega^*). \end{cases}$$



- If $\bar{\delta}(q_0, w) \in F$, we say that w is **accepted** by $\mathcal{M}$.
- The language accepted by $\mathcal{M}$: $L(\mathcal{M}) \equiv \{w \in \Omega^* : \bar{\delta}(q_0, w) \in F\}$.
- $L(\mathcal{M})$ with an automaton $\mathcal{M}$ is called **regular** or Chomsky type-3.

Example 1

Consider a DFA $\mathcal{M} = (Q, \Omega, \delta, q_0, F)$, where $Q = \{q_0, q_1, q_2\}$, $\Omega = \{0, 1\}$, $F = \{q_0\}$ and δ is illustrated in the following diagram:



Then, the language accepted by $\mathcal{M}$ is

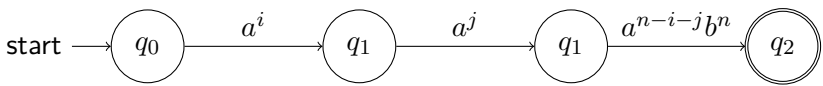
$$\begin{aligned} L(\mathcal{M}) &= \{\varepsilon, 0, 00, \dots, 11, 1001, 10101, \dots\} \\ &= \{x \in \Omega^* : x \text{ is the binary representation of a multiple of 3 or } \varepsilon\} \end{aligned}$$

Example 2

Counter-example

$L = \{a^n b^n : n \geq 1\}$ is not regular.

- Assume L were regular and accepted by a DFA $\mathcal{M} = (Q, \Omega, \ldots)$.
- Take an $n >\mid Q \mid$. When $\mathcal{M}$ reads $a^n = \underbrace{aaa\cdots a}_{n \text{ copies of } a}$, there exists at least one state being visited more than once (Pigeon-hole principle). In the following diagram, q_1 appears twice, where $0 \leq i < n$ and $0 < j < n$:



- Thus if $\mathcal{M}$ accepts $a^n b^n$, $\mathcal{M}$ also accepts $a^{n-j}b^n$, which contradicts with the assumption that $\mathcal{M}$ accepts L.

Regular languages

Lemma 1.2

Any regular language in Ω is accepted by a DFA on Ω.

Proof.

- Let $\mathcal{M} = (Q, \Omega', \delta, q_0, F)$ be a DFA that accepts a regular language $L \subset \Omega^*$.
- We construct a DFA $\mathcal{M}' = (Q', \Omega, \delta', q_0, F)$ from $\mathcal{M}$ by removing symbols in $\Omega' - \Omega$ as follows:
  - $Q' = Q \cup \{q'\}$, where $Q \cap \{q'\} = \varnothing$.
  - $\delta' : Q' \times \Omega \to Q'$ such that
    if $q \in Q$ and $a \in \Omega \cap \Omega'$, $\delta'(q, a) = \delta(q, a)$;
    if $q = q'$ or $a \in \Omega - \Omega'$, $\delta'(q, a) = q'$
- $\mathcal{M}$ does not accept a sting including a symbol in $\Omega' - \Omega$, thus $L(\mathcal{M}') = L(\mathcal{M})$.

Regular languages

Theorem 1.3

The class of regular languages is closed under the set operations $\cap$, $\cup$ and ^{c}.

Proof.

- Closed under ^{c}.
  For a DFA $\mathcal{M} = (Q, \Omega, \delta, q_0, F)$, we can define
  $$\overline{\mathcal{M}} = (Q, \Omega, \delta, q_0, Q - F)$$
  such that $L(\overline{\mathcal{M}}) = \Omega^* - L(\mathcal{M}) = (L(\mathcal{M}))^c$.
- Closed under $\cup$.
  Given $\mathcal{M}_i = (Q_i, \Omega, \delta_i, q_0^i, F_i)$ $(i = 1, 2)$, we can construct
  $$\mathcal{M} = (Q_1 \times Q_2, \Omega, \delta, (q_0^1, q_0^2), F)$$
  such that $\delta((q^1, q^2), a) = (\delta_1(q^1, a), \delta_2(q^2, a))$ and $F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$.
  Then $L(\mathcal{M}) = L(\mathcal{M}_1) \cup L(\mathcal{M}_2)$.
- The closedness under $\cap$ can be proved similarly. □

Recall: Monoids and Homomorphisms

Let M be a set and $\circ$ be a binary operation $M \times M \to M$.

- The structure $(M, \circ)$ is called a semigroup if $\circ$ is associative: $u \circ (v \circ w) = (u \circ v) \circ w$, for all $u, v, w \in M$.
- The structure $(M, \circ, e)$ is called a monoid if $(M, \circ)$ is a semigroup and $e \in M$ satisfies $e \circ w = w \circ e = w$ for all $w \in M$.
- Example. Let Q be a set, $M = \{f : Q \to Q\}$ and $\circ$ the composition of functions, $\mathtt{id}$ be the identity function. Then, $(M, \circ, \mathtt{id})$ is a monoid.
- Example. $(\Omega^*, \cdot, \varepsilon)$ is a monoid, where $\cdot$ is the concatenation of two words.
- Let $(M_i, \circ_i, e_i)$ $(i = 1, 2)$ be two monoids. A function $f : M_1 \to M_2$ is called a (monoid) homomorphism if $f(u \circ_1 v) = f(u) \circ_2 f(v)$ for all $u, v \in M_1$ and $f(e_1) = e_2$.

Monoids and regular languages

Theorem 1.4

The following statements are equivalent.

(1) $L \subset \Omega^*$ is regular.

(2) There is a finite monoid M and monoid homomorphism $\phi : \Omega^* \to M$ such that $L = \phi^{-1}\phi(L)$.

We say a monoid M recognizes L if the above theorem holds.

Proof.
$(1) \Rightarrow (2)$

- Let L be a regular language and $\mathcal{M} = (Q, \Omega, \delta, q_0, F)$ be a DFA that accepts L.
- For each $w \in \Omega^*$, a mapping $f_w : Q \to Q$ is defined by $f_w(q) = \overline{\delta}(q, w)$.
- We obtain a finite monoid $M = \{f_w : w \in \Omega^*\}$ with $f_u \circ f_v(q) = f_v(f_u(q))$ and $\mathtt{id} = f_\varepsilon$.
- Noticing $f_u \circ f_v = f_{uv}$, we can show that $\phi(w) = f_w$ is a monoid homomorphism from Ω^* to M.
- If $f_w = f_{w'}$ and $w \in L$, then $w' \in L$. So $L = \phi^{-1}\phi(L)$.

Proof. (Continued)
(2) $\Rightarrow$ (1)

- Let M be a finite monoid and a monoid homomorphism $\phi : \Omega^* \to M$. Assume $L = \phi^{-1}\phi(L)$.
- A DFA $\mathcal{M} = (Q, \Omega, \delta, q_0, F)$ is constructed as follows:
  - $Q = M$,
  - $\delta(q, a) = q \circ \phi(a)$,
  - q_0 is the identity element of M
  - $F = \phi(L)$.

  Thus $\bar{\delta}(q_0, w) = \phi(w)$. We have

$$\bar{\delta}(q_0, w) \in F = \phi(L) \Leftrightarrow w \in \phi^{-1}\phi(L) = L.$$

- $\mathcal{M}$ recognizes L.

Nondeterministic finite automata

Definition 1.5

A **nondeterministic finite automaton** (NFA) is a 5-tuple $\mathcal{M} = (Q, \Omega, \delta, Q_0, F)$,

(1) Q is a non-empty finite set, whose elements are called **states**.

(2) Ω is a non-empty finite set, whose elements are called **symbols**.

(3) $\delta : Q \times \Omega \to \mathcal{P}(Q)$ is a **transition relation**.

(4) $Q_0 \subset Q$ is a set of **initial states**.

(5) $F \subset Q$ is a set of **final states**.

$\mathcal{P}(Q)$ denotes the power set of Q.

Language accepted by NFA

- Similar to DFA, the transition relation δ of NFA can also be extended as $\bar{\delta} : Q \times \Omega^* \to \mathcal{P}(Q)$,

$$\begin{cases} \bar{\delta}(q, \varepsilon) = \{q\}, \\ \bar{\delta}(q, aw) = \bigcup_{p \in \delta(q,a)} \bar{\delta}(p, w), \end{cases}$$

  and $\bar{\delta}(A, w) = \bigcup_{q \in A} \bar{\delta}(q, w)$.

- If $\bar{\delta}(q_0, w) \cap F \neq \varnothing$, we say that w is accepted by $\mathcal{M}$.
- The language accepted by $\mathcal{M}$:

$$L(\mathcal{M}) = \{w \in \Omega^* : \bar{\delta}(Q_0, w) \cap F \neq \varnothing\}.$$

NFA vs. DFA

Theorem 1.6

The language accepted by NFA is regular. That is, for any NFA $\mathcal{M}$, there is a DFA $\mathcal{M}'$ such that $L(\mathcal{M}) = L(\mathcal{M}')$.

Proof. For a NFA $\mathcal{M} = (Q, \Omega, \delta, Q_0, F)$, construct a DFA $\mathcal{M}' = (Q', \Omega, \delta', q_0', F')$ as follows:

$$\begin{aligned} Q' &= \mathcal{P}(Q), \\ \delta'(A, a) &= \bigcup_{q \in A} \delta(q, a) \text{ with } A \in Q', \\ q_0' &= Q_0, \\ F' &= \{A \in Q' : A \cap F \neq \varnothing\}. \end{aligned}$$

Then $\overline{\delta'}(q_0', w) = \overline{\delta}(Q_0, w)$, and thus $L(\mathcal{M}') = L(\mathcal{M})$. □

Lemma 1.7

The following holds for regular languages in Ω.

$(r1)$ $\varnothing$ is regular.

$(r2)$ For any $a \in \Omega$, $\{a\}$ is regular.

$(r3)$ If A, $B \subset \Omega^*$ are regular, so is $A \cup B$.

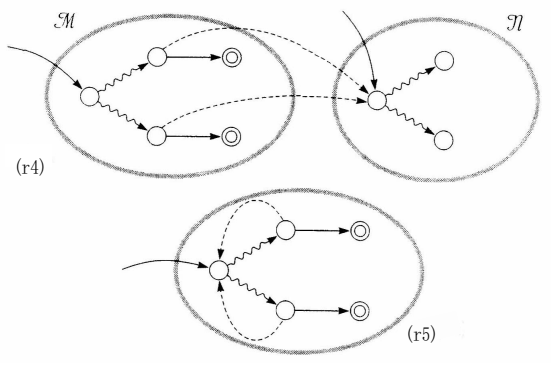
$(r4)$ If A, $B \subset \Omega^*$ are regular, so is $A \cdot B = \{v \cdot w : v \in A, w \in B\}$.

$(r5)$ If A is regular, so is $A^* = \{w_1 w_2 \cdots w_n : w_i \in A\}$.

Proof. $(r1)$ and $(r2)$ are obvious. $(r3)$ was shown by Theorem 1.3

To show $(r4)$: Suppose A, $B \subset \Omega^*$ are accepted by $\mathcal{M}$, $\mathcal{N}$, respectively.
We combine these NFA's sequentially and nondeterministically as in the figure (r4).
In other words, the input is nondeterministically split into two parts such that the former can be accepted by $\mathcal{M}$ while the latter accepted by $\mathcal{N}$.

To show $(r5)$: Similarly, in the figure (r5) below, the input is nondeterministically split into many parts each of which can be accepted by $\mathcal{M}$.



Regular expression

- By the previous theorem, the languages in Ω, obtained from $\{a\}$ $(a \in \Omega)$ by way of operations $\cup$, $\cdot$, *, are regular.
- For simplicity, we write $\{a\}$ as a, $\cup$ as $+$ and omit $\cdot$.
  E.g., $\{a\} \cdot (\{a\} \cup \{b\})^*$ is written as $a(a+b)^*$.
- Such expressions for regular languages are called regular expressions.
- S.C. Kleene showed that the the class of regular languages coincides with the class of languages described by regular expressions.

Theorem 1.8 (Kleene)

The class of regular languages is the smallest class that satisfies the conditions $(r1)$, $(r2)$, $(r3)$, $(r4)$ and $(r5)$.

Proof.

- Goal: for any $\mathcal{M} = (Q, \Omega, \delta, q_0, F)$, $L(\mathcal{M})$ can be described by a regular expression.
- Let $Q = \{q_0, q_1, \ldots, q_n\}$. The language accepted by $\mathcal{M}_{i,j} = (Q, \Omega, \delta, q_i, \{q_j\})$ is denoted as $L_{i,j}$.
- If only the states of $\{q_0, q_1, \ldots, q_k\}$ (except for the initial and final states) are visited while $\mathcal{M}_{i,j}$ is processing, we denote the language as $L^k_{i,j}$. Moreover, for the sake of convenience, we set (for $k = -1$) $L^{-1}_{i,j} = \{a : \delta(q_i, a) = q_j\}$.
- We next show that for any i, j, $L^k_{i,j}$ can be described by a regular expression by induction on $k \geq -1$.
  - $L^{-1}_{i,j} \subseteq \Omega$ is finite set of symbols, so it can be described by a regular expression.
  - For $k \geq 0$,
    $$L^k_{i,j} = L^{k-1}_{i,j} + L^{k-1}_{i,k}(L^{k-1}_{k,k})^* L^{k-1}_{k,j}$$
    which can be described by a regular expression.
- Finally $L = \bigcup_{p_j \in F} L^n_{0,j}$. Thus L can also be described by a regular expression.

Highlights

- Any nondeterminstic FA can be rebuilt into a determinictic FA.
  Question: How about functionally-expanded automata. [Yes for Turing machines. No for push-down automata.]
- L can be accepted by an automaton iff L has a regular expression.
  Question: A regular expression can be viewed as a generative grammar. Can you rewrite $a(a+b)^*$ as transformational rules? Also, can you find transformational rules which produce a non-regular language.

Reference

J.E. Hopcroft, R. Motwani and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*, 2nd edition, Addison-Wesley 2001.

Appendix – Chomsky hierarchy

Grammar Type	Grammar	Machine
Type 0	Unrestricted	Turing machines
Type 1	Context-sensitive	linear bounded automata
Type 2	Context-free	pushdown automata
Type 3	Regular	finite state automata

Thank you for your attention!