Logic and Computation: I

Part 1. Introduction to theory of computation

Kazuyuki Tanaka

Yanqi Lake Beijing Institute of Mathematical Sciences and Applications (BIMSA)

Oct. 27, 2022

北京雁栖湖应用数学研究院
YANQI LAKE BEIJING INSTITUTE OF MATHEMATICAL SCIENCES AND APPLICATIONS

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

Historical Introduction

- At the beginning of the 20th century, D. Hilbert took a strong interest in the mechanical processing of strings of symbols in the name of formalism. He then advocates "the decision problem (for validity or satisfiability of logical expressions) must be considered the main problem of mathematical logic".
- Hilbert's decision problem was upset by K. Gödel, A. Church and A. Turing in the way of formalizing the symbolic processing mathematically.
- 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 an introductory graduate-level course in **mathematical logic** and **theory of computation**. Its first part delivered in this semester covers the basic topics of the two fields and their interactions. So, advanced undergraduates are welcome to participate in this course from this semester.
2. Each week, there are two lectures, in Tuesday and Thursday. Every Thursday, we will assign simple homework problems or questionnaires to registered students, who are motivated to attend the class continuously. Normally, homeworks are due next Monday.
3. TA (Dr. Li) is in charge of the last ten minutes of each lecture to explain homework assignments and makes comments on submitted homeworks. She will also handle questions and comments from students via WeChat. We won't be able to accept questions during lectures, until the class members are fixed. Instead, we may have online office hours or extra lessons by appointment.
4. Lecture slides will be uploaded on the lecture page at BIMSA.

Introducing myself

Education

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

Teaching Jobs

- ⋆ 1986 ∼ 1991, Tokyo Institute of Technology
  Assistant Professor, Dept. of Info. Sci.;
  Visiting PennState.
- ⋆ 1991 ∼ 1997, Tohoku University
  Associate Professor, Dept. of Math.;
  Visiting Oxford.
- ⋆ 1997 ∼ 2022, Tohoku University
  Professor (2021 , Emeritus), Mathematical Institute and Research Alliance Center for Mathematical Sciences.
- ⋆ 2022 ∼ now, BIMSA, Research Fellow.

Speciality

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

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, and the decidability of Presburger arithmetic. We will use Ehrenfeucht-Fraïssé game as a basic tool of first-order logic, and apply it to prove Lindström's theorem.

Logic and Computation II

We will move on to Gödel's incompleteness theorem, second-order logic, infinite automata, determinacy of infinite games, etc.

Part 1. Schedule

- Oct.27, (1) Automata and monoids
- Nov. 1, (2) Turing machines
- Nov. 3, (3) Computable functions and primitive recursive functions
- Nov. 8, (4) Decidability and undecidability
- Nov.10, (5) Partial recursive functions and computable enumerable sets
- Nov.12, (6) Rice's theorem and many-one reducibility

Part 1 (1). Automata and Monoids

Introduction

- A (finite) automaton is a simplest computing machine with finitely many states. Other computing machines such as a Turing machine can be regarded as automata expanded functionally.

- 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. The set of words a machine $\mathcal{M}$ accepts is called the **language** accepted by $\mathcal{M}$, denoted $L(\mathcal{M})$, and $L(\mathcal{M}) \subset \Omega^*$.

- In automata theory, we study the class of languages $L(\mathcal{M})$ with automata $\mathcal{M}$. In theory of computation, larger classes of languages are also defined and studied for many kinds of functionally expanded machines $\mathcal{M}$.

Deterministic finite automaton

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

Definition

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**.

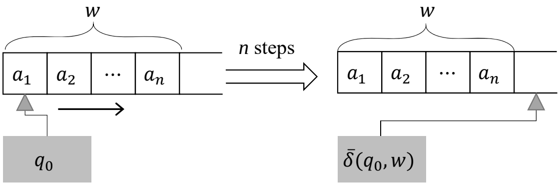
(4) $q_0 \in Q$ is an **initial state**.

(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:

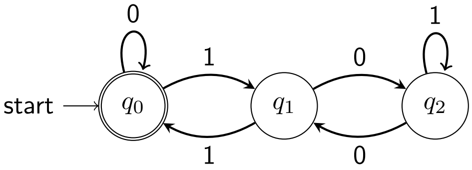
$$\begin{cases} \bar{\delta}(q, \varepsilon) = q, \\ \bar{\delta}(q, aw) = \bar{\delta}(\delta(q, a), w) \quad (a \in \Omega,\ w \in \Omega^*). \end{cases}$$



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

Example 1

Considering the following DFA $\mathcal{M} = (Q, \Omega, \delta, q_0, F)$, where $Q = \{q_0, q_1, q_2\}$, $\Omega = \{0, 1\}$, $F = \{q_0\}$,



the languages accepted by $\mathcal{M}$ is

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

Example 2

Counterexample

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

- Assume L is regular and accepted by a DFA $\mathcal{M} = (Q, \Omega, \ldots)$.
- Assume $n >| Q |$. 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 (Pigeonhole principle). In the following diagram, q_1 appears twice, where $0 \leq i < n$ and $0 < j < n$:

$$\text{start} \longrightarrow q_0 \xrightarrow{a^i} q_1 \xrightarrow{a^j} q_1 \xrightarrow{a^{n-i-j}b^n} q_2$$

- 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

The regular languages on Ω is accepted by DFA on Ω.

Proof.

- Let $\mathcal{M} = (Q, \Omega', \delta, q_0, F)$ be a DFA that accepts the regular language $L \subset \Omega^*$.
- Construct $\mathcal{M}' = (Q', \Omega, \delta', q_0, F)$:
  - $Q' = Q \cup \{q'\}$, 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 $a \in \Omega - \Omega'$, $\delta'(q, a) = q'$
- The DFA $\mathcal{M}'$ is obtained from $\mathcal{M}$ by removing symbols in $\Omega' - \Omega$.
- $\mathcal{M}$ does not accept a sting including a symbol in $\Omega' - \Omega$, thus $L(\mathcal{M}') = L(\mathcal{M})$.

$\square$

Regular languages

Theorem

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

Proof.

- closeness 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 $L(\overline{\mathcal{M}}) = \Omega^* - L(\mathcal{M}) = (L(\mathcal{M}))^c$.
- closeness 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 $L(\mathcal{M}) = L(\mathcal{M}_1) \cup L(\mathcal{M}_2)$.
- $\cap$ can be proved similarly. $\square$

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

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

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**.

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

Language accepted by NFA

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

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

  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

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

The following holds for regular languages over 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 $A \cdot B = \{v \cdot w : v \in A, w \in B\}$.

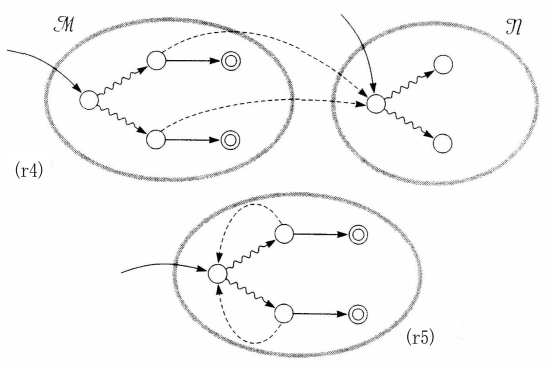
$(r5)$ If A is regular, so is $A^* = \{w_1 w_2 \cdots w_n : w_i \in A\}$.

Proof idea for (r4) and (r5)

To show (r4):

- An input is accepted by an NFA if the input can be split into two parts such that the former can be accepted by $\mathcal{M}$ while the latter accepted by $\mathcal{N}$.

- Nondeterminism is necessary: an automata should nondeterministically guess where to divide the input.



Regular expression

- By the previous theorem, for all $a \in \Omega$, starting from $\{a\}$ we can inductively define a class over Ω that is closed under the operations, $\cup$, $\cdot$, *, which is the so-called regular expression.
- For simplicity, we write $\{a\}$ as a, $\cup$ as $+$ and omit $\cdot$. e.g., $\{a\} \cdot (\{a\} \cup \{b\})^*$ is written as $a(a+b)^*$.
- Regular expression has a wide application in computer science, such as text processing.

S.C. Kleene showed that the the class of regular languages coincides with the class of regular expressions.

Theorem (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 regular expression.
- Finally $L = \bigcup_{p_j \in F} L^n_{0,j}$. Thus L can also be described by regular expression.

Summary

- 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 is a regular language iff there is a regular expression R such that $L(R) = L$.
  Question: A regular expression can be viewed as a generative grammar. Can you rewrite $a(a+b)^*$ as transformational rules?

Further readings

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!