Logic and Computation: I

Chapter 1. Introduction to theory of computation
Turing machine

Kazuyuki Tanaka

BIMSA

November 1, 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.

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.
- **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. Ehrenfeucht-Fraïssé game and Lindström's theorem.

Logic and Computation II

Gödel's incompleteness theorem, second-order logic, infinite automata, infinite games, descriptive set theory, admissible ordinals, 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

Recapitulation: DFA and NFA

NFA $\mathcal{N} = (Q', \Omega', \delta', Q_0, F')$

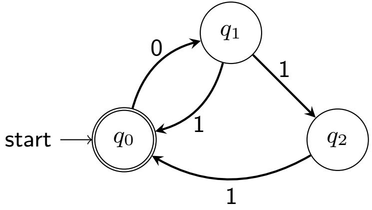
- Q' is a finite set of states.
- Ω' is a finite set of symbols.
- $\delta' : Q' \times \Omega' \to \mathcal{P}(Q')$ or equiv. $\delta' \subset Q' \times \Omega' \times Q'$ is a transition relation.
- $Q_0 \subset Q'$ is a set of initial states.
- $F' \subset Q'$ is a set of final states.

DFA $\mathcal{M} = (Q, \Omega, \delta, q_0, F)$

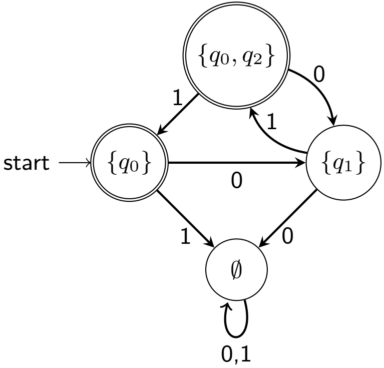
- Q is a finite set of states.
- Ω is a finite set of symbols.
- $\delta : Q \times \Omega \to Q$ is a transition function.
- $q_0 \in Q$ is an initial state.
- $F \subset Q$ is a set of final states.

The equivalence of DFA and NFA

NFA $\mathcal{N} = (Q, \Omega, \delta, q_0, F)$



DFA $\mathcal{M} = (Q', \Omega', \delta', Q_0, F')$



Redundant states are omitted.

Both recognize regular language

$$L = (01 + 011)^*$$

.

Monoids, regular languages and regular expressions

Theorem

L is regular language iff L is recognized by a finite monoid.

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 $A^* = \{w_1 w_2 \cdots w_n : w_i \in A\}$.

The regular languages are closed under $\cap$, $\cup$, $\cdot$, ^{c} and *.

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 $Q = \{q_0, q_1, \dots, q_n\}$. 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 $\{q_0, q_1, \dots, q_k\}$ (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 $L = \bigcup_{q_j \in F} L^n_{0,j}$. Thus L can also be described by regular expression.

Turing machine

The birth of Turing machine

Alan Turing (1912 – 1954)

In the paper "On Computable numbers, with an application to the Entscheidungsproblem", Turing reformulated K. Gödel's arithmetic-based arguments as symbol processing arguments, which produces a simple model of computation, now known as Turing machine.

- The importance of Entscheidungsproblem (decision problem) was emphasized by D. Hilbert in 1928. Turing claimed that his "universal computing machine" can perform any conceivable mathematical computation algorithmically. He further proved that the halting problem for Turing machines is undecidable.
- Compared with finite automata with a limited amount of memory, the Turing machine has infinite and unrestricted memory and is the model of general purpose computers.

Definition

(Deterministic) Turing machine (TM) is a 5-tuple $\mathcal{M} = (Q, \Omega, \delta, q_0, F)$,

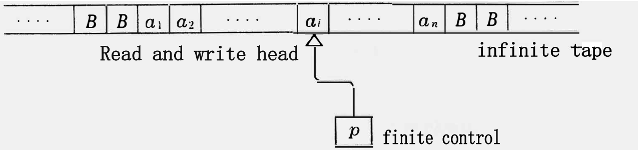
(1) Q is a non-empty finite set, whose elements are called **states**.

(2) Ω is a non-empty finite set, whose elements are called **symbols**. The black symbol $B \in \Omega$.

(3) $\delta : Q \times \Omega \to \Omega \times \{R, L, N\} \times Q$ is called **transition function**.

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

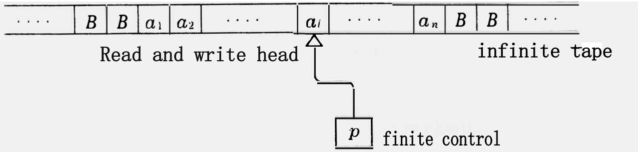
(5) $F \subset Q$ is a set of **final states**.



- The difference with DFA lies in the transition function

$$\delta : Q \times \Omega \to \Omega \times \{R, L, N\} \times Q.$$

- $\delta(p, a_i) = (b, x, q)$ means that at state p, if $\mathcal{M}$ reads symbol a_i on the tape, then
  - the head write b to replace a_i,
  - according to $x = R,\ L,\ N$,
    the head moves to the right or move left or keep still,
    the control state changes to q



- A configuration of TM, denoted $a_1 \cdots a_{i-1} p a_i \cdots a_n$, describes:
  - A string $a_1 \cdots a_n \in \Omega^*$ is written on the tape. There are no (non-blanck) symbols outside of $a_1 \cdots a_n$ on the tape while the blanck B may be included in the sequence,
  - the head is pointed at a_i on the tape,
  - the current control state is p.

We say configuration α yields configuration α', denoted as $\alpha \triangleright \alpha'$, if there is a legal transition from configuration α to configuration α' as follows:

1) if $\delta(p, a_i) = (a_i', L, q)$,

$$a_1 \cdots a_{i-1} p a_i \cdots a_n \triangleright a_1 \cdots a_{i-2} q a_{i-1} a_i' a_{i+1} \cdots a_n \ (i > 1),$$
$$p a_1 a_2 \cdots a_n \triangleright q B a_1' a_2 \cdots a_n.$$

2) if $\delta(p, a_i) = (a_i', N, q)$,

$$a_1 \cdots a_{i-1} p a_i \cdots a_n \triangleright a_1 \cdots a_{i-1} q a_i' a_{i+1} \cdots a_n.$$

3) if $\delta(p, a_i) = (a_i', R, q)$,

$$a_1 \cdots a_{i-1} p a_i \cdots a_n \triangleright a_1 \cdots a_{i-1} a_i' q a_{i+1} \cdots a_n \ (i \leq n),$$
$$a_1 \cdots a_{n-1} a_n p \triangleright a_1 \cdots a_{n-1} a_n' B q.$$

We write the sequence of computation $\alpha_0 \triangleright \alpha_1 \triangleright \cdots \triangleright \alpha_n$ as $\alpha_0 \triangleright^* \alpha_n \ (n \geq 0)$.

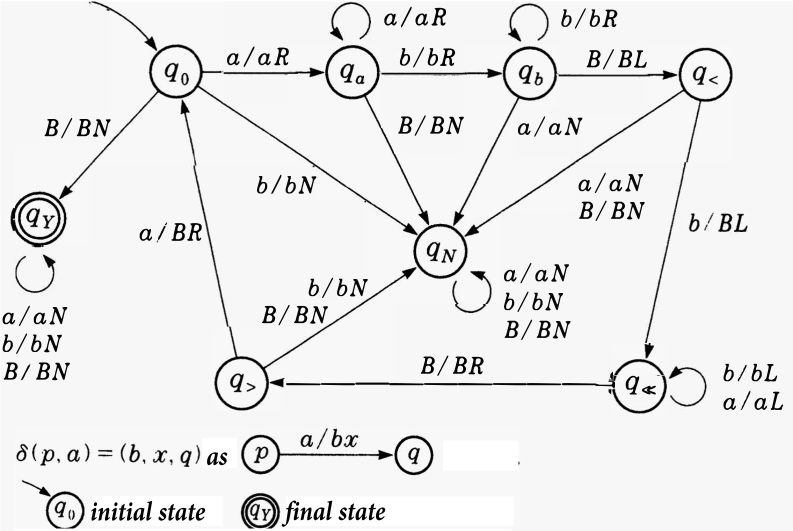
- We say $\mathcal{M}$ **accepts** $a_1 \cdots a_n \in (\Omega - \{B\})^*$ if there exists $b_1 \cdots b_m$ and $q \in F$ such that $q_0 a_1 \cdots a_n \rhd^* b_1 \cdots b_i q b_{i+1} \cdots b_m$. That is, some final state $q \in F$ is visited in the computation.
- The languages (of the strings) accepted by $\mathcal{M}$ is denoted as $L(\mathcal{M})$.
- The languages accepted by a TM is also called **type-0** languages.
- Regular languages are also type-0 languages (since a Turing machine is an extension of finite automata). But there are also non-regular type-0 languages.

Example

$L = \{a^n b^n : n \geq 0\}$.

Example

$L = \{a^n b^n : n \geq 0\}$ is a type-0 language.



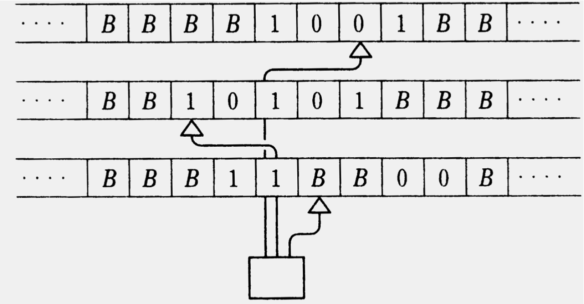
k-tape TM

Definition

A k-**tape** TM is a 5-tuple $\mathcal{M} = (Q, \Omega, \delta, q_0, F)$, where the transition function is

$$\delta : Q \times \Omega^k \to \Omega^k \times \{R, L, N\}^k \times Q.$$

A k-tape TM is also called multitape TM.

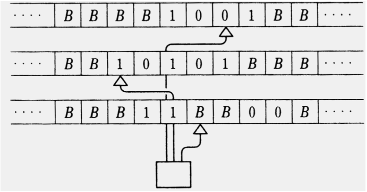


Theorem

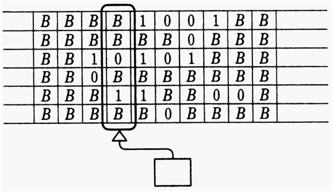
The languages accepted by a multitape TM is 0-type.

Proof.

- Assume L is accepted by a k-tape TM $\mathcal{M} = (Q, \Omega, \delta, q_0, F)$.
  Goal: construct a single tape TM $\mathcal{M}'$ that can simulate $\mathcal{M}$.
- Divide the single tape of $\mathcal{M}'$ into k track and each track is used to simulate one tape of $\mathcal{M}$.
- In addition, $\mathcal{M}'$ needs another k tapes to record each head position of $\mathcal{M}$.
- The alphabet for $\mathcal{M}'$ is $\Omega' = (\Omega \times \{0, B\})^k$.



$\mathcal{M}$



$\mathcal{M}'$

Theorem

The class of 0-type languages is closed under $\cap$ and $\cup$.

Proof.

$\cup$ case.

- Assume 0-type languages A, B are accepted by Turing machines $\mathcal{M}$, $\mathcal{M}'$.
- To accept $A \cup B$, we construct a 2-tape TM $\mathcal{N}$ as follows.
- $\mathcal{N}$ copy the input of its 1st tape to the 2nd tape, then $\mathcal{M}$ works on the 1st tape and $\mathcal{M}'$ on the 2nd tape simultaneously.
- If either $\mathcal{M}$ or $\mathcal{M}'$ enters the final configuration, so is $\mathcal{N}$.

$\cap$ case can be proved in a similar way. □

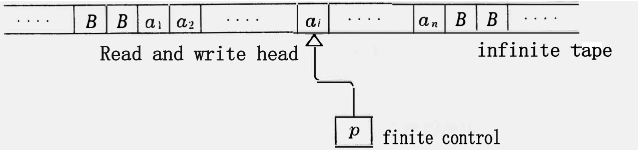
Definition

(Nondeterministic) Turing machine (TM) is a 5-tuple $\mathcal{M} = (Q, \Omega, \delta, q_0, F)$,

(1) Q, Ω, F are same as deterministic case,

(2) $Q_0 \subset Q$ is a set of **initial state**.

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



Theorem

The language accepted by a nondeterministic TM is 0-type.

Proof.

- Let $\mathcal{M}$ be a nondeterministic TM.
- Goal: build a deterministic 3-tape TM $\mathcal{M}'$ to simulate $\mathcal{M}$, since already know that the language accepted by $\mathcal{M}'$ is type-0.
- Let l be the maximum number of branches that occur at each point of the computation, that is, $l = \max\{|\ \delta(q,a)\ |: q \in Q$ and $a \in A\}$.
- Then, each computation process can be uniquely represented by a finite sequence of l symbols $x_1, \ldots, x_l$, because it is determined by which branch is chosen at each point (Not all strings over $x_1, \ldots, x_l$ have corresponding computational processes).

Proof.(Continued)

The construction of $\mathcal{M}'$: the roles of the three tapes

- 1st tape: only for input, which will be read many times but never rewritten.
- 2nd tape: for recording which branch is chosen at each point. A finite sequence of symbols $x_1, \ldots, x_l$ can be regarded as a natural number in the $l+1$-base and thus such sequences are linearly ordered.
- 3rd tape: for performing the computation process of $\mathcal{M}$ according to the branching information written on the 2nd tape.

Proof.(Continued)

$\mathcal{M}'$ mimics $\mathcal{M}$

(1) $\mathcal{M}'$ writes the first string on the 2nd tape.

(2) $\mathcal{M}'$ copies the input from the 1st tape to the 3rd tape.

(3) $\mathcal{M}'$ mimics $\mathcal{M}$ on the 3rd tape according to the branching information on the 2nd tape.

(4) If $\mathcal{M}$ accepts the input along this computation, $\mathcal{M}'$ also accepts the input.

(5) If it fails to proceed the computation or ends with a non-final state, then change the contents of the 2nd tape to the next string and go back to (2).

- Note that $\mathcal{M}'$ is always deterministic.
- It is clear from the construction that $\mathcal{M}'$ accepts the same languages as $\mathcal{M}$. □

Then by similar arguments for regular languages, we can also prove the following theorem by using the nondeterminism of TM.

Theorem

The class of 0-type languages is closed under $\cdot$ and *

- Up to now, the TM and variants of TM we considered are devices that can decide whether an input is accepted or not.
- Notice that when the machine enter a final state, it leaves a string on the tape. If we regard such a string as an output of this TM for a given input, we can naturally define a function from strings to strings.
- This is the so-called Turing definable function.

Remark

- Such a function is partially defined, since the TM does not always terminate.
- To make the output unique, we define the output of (deterministic) TM as the string on the tape when the TM enters a final state for the first time, because it might enter a final state more than once.
- For a multitape TM and a nondeterministic TM, the output should be considered to be the output of an equivalent single tape deterministic ones.

Theorem

Let $\sharp$ be a new symbols not included in Ω. The following are equivalent:
(1) A function $f : A \to \Omega^*$ $(A \subset \Omega^*)$ can be defined by a TM with output.
(2) $\{u\sharp f(u) : u \in A\}$ is a 0-type langauge.

Proof.
$(1) \Rightarrow (2)$.
Assume a partial function $f : \Omega^* \to \Omega^*$ is definable by a TM $\mathcal{M}$. We define a 2-tape $\mathcal{M}'$ working as follows:

- It can check the string on the 1st tape is in the form of $u\sharp v$
- Then $\mathcal{M}'$ copies u to the 2nd tape and works on the 2nd tape to simulate $\mathcal{M}$.
- If $\mathcal{M}$ enters a final state, it checks whether the string on the 2nd tape is the same as v on the 1st tape. If yes, then $\mathcal{M}'$ also enters a final state.

$(2) \Rightarrow (1)$.

Assume a TM $\mathcal{M}'$ that accepts $\{u\sharp f(u) : u \in A\}$. Next, we consider a nondeterministic $\mathcal{M}$ (with output).

- $\mathcal{M}$ has 2 tapes.
- $\mathcal{M}$ non-deterministically produces a string $v \in \Omega^*$ on the 2nd tape.
- After the input string u on the 1st tape, write $\sharp$ and copy v after $\sharp$. Then mimic $\mathcal{M}'$ on the 1st tape.
- When it reaches a final state, it empties the 1st tape, copies the contents of the 2nd tape again, and then $\mathcal{M}$ enters a final state.
- The nondeterminism lies in writing an arbitrary string on the 2nd tape, which is equivalent to enumerating all the possible $f(u)$.

□

Summary

- TM is more expressive than FA.
  the class of type-3 languages $\subset$ the class of type-0 languages
- Type-0 languages are closed under $\cap$, $\cup$,* (Kleene star operation), $\cdot$(concatenation).
  Question: Are type-0 languages closed under ^{c}(complementation)?
  Answer: No.
- Turing definable functions

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

Quiz 1

- For any string w, the reverse of w is written as w^R, e.g., $w = w_1 w_2 \cdots w_n$ and $w^R = w_n \cdots w_2 w_1$.
- $L^R = \{w^R : w \in L\}$.

Quiz

(1) Assume L is regular. Which of FA and/or TM can accept LL^R?

(2) Which of FA and/or TM can accept $L' = \{ww^R : w \in \{0,1\}^*\}$?

(2) Which of FA and/or TM can accept $L'' = \{0^n 1^n : n \geq 0\}$?

Quiz

Please scan the following QR code to submit your answer now.

Thank you for your attention!