Logic and Computation I

Chapter 1. Introduction to theory of computation
Turing machine

Kazuyuki Tanaka

BIMSA

September 12, 2024

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 recursion theory as well as the connection between them.
- **Part 2. Propositional Logic and Computational Complexity**
  The basics of propositional logic 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 and incompleteness theorems. Ehrenfeucht-Fraïssé games and Lindström's theorem.
- **Part 4. Modal logic**
  Kripke models, standard translation, decidability 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

Recapitulation: DFA and NFA

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

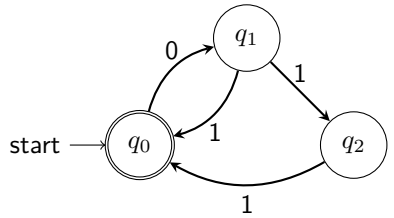
- Q' is a finite set of states.
- Ω' is a finite set of symbols.
- $\delta' : Q' \times \Omega' \to \mathcal{P}(Q')$ or equiv. $\delta' \subset Q' \times \Omega' \times Q'$ is a transition relation.
- $Q_0 \subset Q'$ is a set of initial states.
- $F' \subset Q'$ is a set of final states.

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

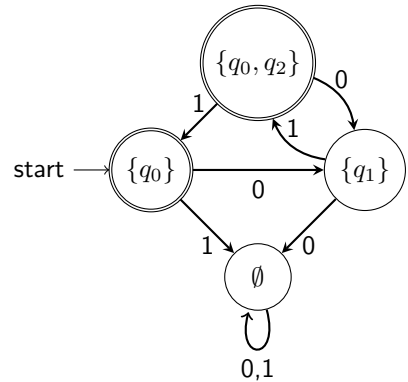
- Q is a finite set of states.
- Ω is a finite set of symbols.
- $\delta : Q \times \Omega \to Q$ is a transition function.
- $q_0 \in Q$ is an initial state.
- $F \subset Q$ is a set of final states.

The equivalence of DFA and NFA

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



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



Both recognize the same regular language

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

Redundant states are omitted.

.

Regular languages and regular expressions

Theorem 1.4

L is regular (accepted by a finite automaton) iff L is recognized by a finite monoid.

Theorem 1.8 (Kleene)

The class of regular languages is the smallest class that satisfies the conditions

$(r1)$ $\varnothing$ 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 also closed under $\cap$ and ^{c}.

Proof.

- Goal: Given any $\mathcal{M} = (Q, \Omega, \delta, q_0, F)$, find a regular expression for $L(\mathcal{M})$.
- 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.

§1.2 Turing machines

Alan Turing (1912 – 1954)

In his 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 1.9

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

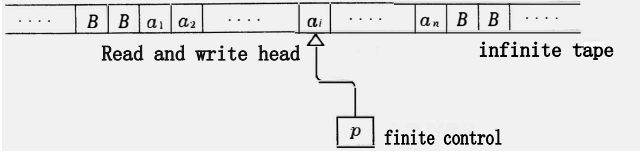
(1) Q is a non-empty finite set, whose elements are called **states**.

(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 a **transition function**.

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

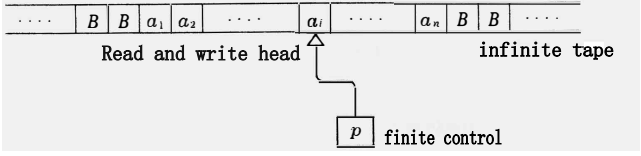
(5) $F \subset Q$ is a set of **final states**.



- 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 writes 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 in the current state 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 a sequence of computation $\alpha_0 \triangleright \alpha_1 \triangleright \cdots \triangleright \alpha_n$ as $\alpha_0 \triangleright^* \alpha_n \ (n \geq 0)$.

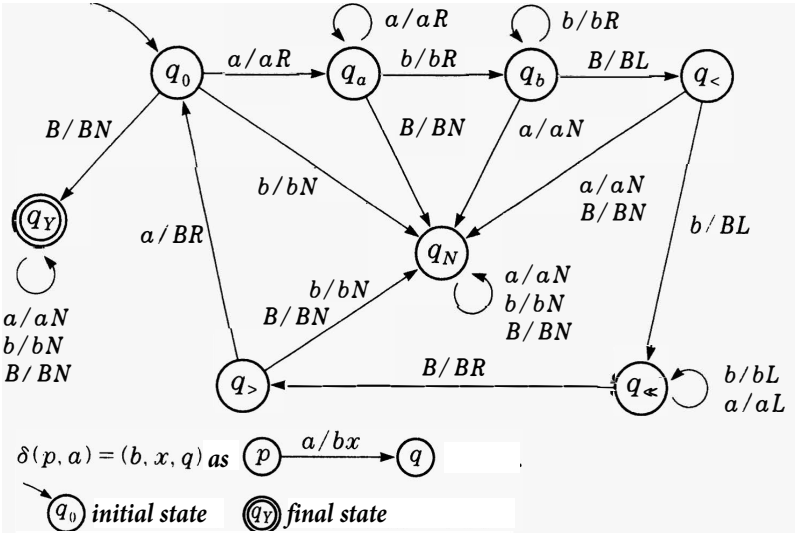
- We say $\mathcal{M}$ **accepts** a word $a_1 \cdots a_n \in (\Omega - \{B\})^*$ if there exists $b_1 \cdots b_m$ and $q \in F$ such that $q_0 a_1 \cdots a_n \triangleright^* b_1 \cdots b_i q b_{i+1} \cdots b_m$. That is, some final state $q \in F$ is visited in the computation.
- The language of the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

Recall Example 2 of §1.1

$L = \{a^n b^n : n \geq 0\}$ is not regular.

Example 3

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



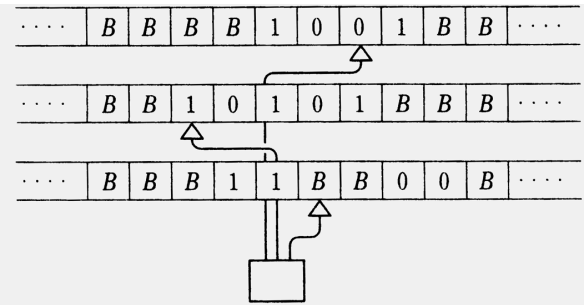
Multitape TM

Definition 1.10

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

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

Example: A 3-tape TM is illustrated as below.

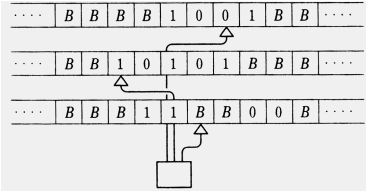


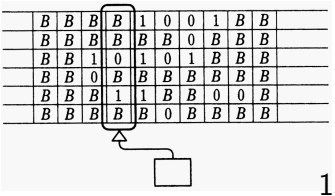
Theorem 1.11

A language accepted by a multitape TM is 0-type.

Proof.

- Let L be a language accepted by a k-tape TM $\mathcal{M} = (Q, \Omega, \delta, q_0, F)$. We construct a single tape TM $\mathcal{M}'$ that can accept L by simulating $\mathcal{M}$.
- The tape of $\mathcal{M}'$ consists of k tracks, each of which is used to simulate one tape of $\mathcal{M}$. In addition, $\mathcal{M}'$ needs another k tracks to record the head position of each tape of $\mathcal{M}$. Thus, a cell of the tape of $\mathcal{M}'$ contains k symbols of $\mathcal{M}$ together with k symbols from $\{0, B\}$, where 0 denotes the head position. So, we set the alphabet Ω' of $\mathcal{M}'$ as $(\Omega \times \{0, B\})^k$.





- To simulate one step of $\mathcal{M}$, $\mathcal{M}'$ first needs to recognize all the k symbols at the head positions of $\mathcal{M}$, and then following δ, it rewrites its tape contents and changes the state.
- We omit the details, but it is not difficult to see that $\mathcal{M}'$ accepts L. □

Theorem 1.12

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

Proof.

Closed under $\cup$.

- Let A, B be 0-type languages accepted by TM's $\mathcal{M}$, $\mathcal{M}'$, respectively.
- We construct a 2-tape TM $\mathcal{N}$ which accepts $A \cup B$ as follows.
- First, $\mathcal{N}$ copies the input on its 1st tape to the 2nd tape, then $\mathcal{N}$ mimics $\mathcal{M}$ on the 1st tape and $\mathcal{M}'$ on the 2nd tape simultaneously.
- If either $\mathcal{M}$ or $\mathcal{M}'$ enters a final state, so does $\mathcal{N}$.

We can similarly show that it is also closed under $\cap$. □

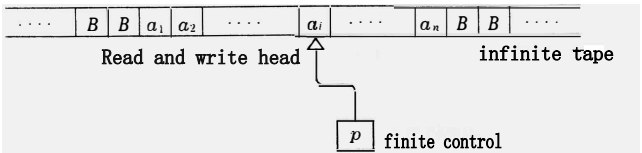
Definition 1.13

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

(1) Q, Ω, F are same as the 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 a **transition relation**.



Theorem 1.14

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

Proof.

- Let $\mathcal{M}$ be a nondeterministic TM.
- It is enough to build a deterministic 3-tape TM $\mathcal{M}'$ to simulate $\mathcal{M}$, since the language accepted by a multitape TM is type-0 by Theorem 1.11.
- Let l be the maximum number of branches at each point of the computation, that is, $l = \max\{|\ \delta(q, a)\ | : q \in Q \text{ and } a \in A\}$.
- 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 roles of the three tapes of $\mathcal{M}'$

- 1st tape for input, which will be read many times but never rewritten.
- 2nd tape for recording a finite sequence of symbols $x_1, \ldots, x_l$ to instruct which branch should be chosen at each point. Such a sequence 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 instruction and its order of the sequence 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 language as $\mathcal{M}$. □

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

Theorem 1.15

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 a machine enters 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 called a 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 a (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 equivalent single tape deterministic ones.

Theorem 1.16

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 type-0 language.

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 $f(u)$.

□

Highlights

- TM is a more expressive power than FA.
  That is, the class of type-3 languages is properly included in the class of type-0 languages.
- The class of Type-0 languages is closed under $\cap$, $\cup$, * (Kleene star operation), and $\cdot$ (concatenation).
  Question: Is the class of 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.

Quiz 1

- For any string $w = w_1 w_2 \cdots w_n$, the reverse of w is defined as $w^R = w_n \cdots w_2 w_1$.
- Let $L^R \equiv \{w^R : w \in L\}$.

Quiz

(1) Is LL^R regular for any regular L ?

(2) Is $L' = \{ww^R : w \in L\}$ regular for any regular L ?

(3) Is $L'' = \{w \in \{9\}^* : w$ appears in $\pi = 3.141592 \cdots$ as a subsequence.$\}$ is type-0 or regular or neither?

Quiz

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

Quiz 1, AU2024

Thank you for your attention!