

Chap. 8 Introduction to Turing Machine

8.2 The Turing Machine

A Turing Machine(TM) M is a 7-tuples, $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ where

1. Q : finite set of states,
2. Σ : a finite set of input symbols,
3. Γ : a finite set of tape symbols, ($\Sigma \subseteq \Gamma$)
4. $\delta: Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L, R\}}$ or $\delta \subseteq (Q \times \Gamma) \times (Q \times \Gamma \times \{L, R\})$.
 $(p, Y, D) \in \delta(q, X)$ where $p, q \in Q, X, Y \in \Gamma, D \in \{L, R\}$
 in state q with tape symbol X ,
 move to state p , tape symbol is replaced to Y ,
 tape head moves to left(L) or right(R).
5. $q_0 \in Q$, initial state,
6. B : blank symbol, $B \in \Gamma, B \notin \Sigma$,
7. F : a set of final or accepting states, $F \subseteq Q$.

Instantaneous Description for Turing Machine

$$(\alpha, q, X\beta) \in \Gamma^* \times Q \times \Gamma^*.$$

Tape string $\alpha X\beta$ is surrounded by infinite blanks.

Current tape symbol is X and assume $\alpha = \alpha'Z$.

$$\text{if } (p, Y, R) \in \delta(q, X), \quad (\alpha, q, X\beta) \vdash_M (\alpha Y, p, \beta),$$

$$\text{if } (p, Y, L) \in \delta(q, X), \quad (\alpha, q, X\beta) = (\alpha'Z, q, X\beta) \vdash_M (\alpha', p, ZY\beta).$$

TM also is a finite automaton with read/write tape

$$\text{if } q \xrightarrow{X/Y, R} p, \quad (\alpha, q, X\beta) \Rightarrow_M (\alpha Y, p, \beta),$$

$$\text{if } q \xrightarrow{X/Y, L} p \quad (\alpha, q, X\beta) = (\alpha'Z, q, X\beta) \Rightarrow_M (\alpha', p, ZY\beta).$$

$$L(M) = \{w \in \Sigma^* \mid (\varepsilon, q_0, w) \vdash_M^* (\alpha, f, \beta), f \in F\}$$

L is recursively enumerable, if there is a TM M such that $L = L(M)$.

Class of recursively enumerable (RE) languages

Type 0 languages in Chomsk's hierarchy

We say **TM halt**, if $\exists \delta(q, X)$ (and does **not accept**) or $q \in F$ (and **accepts**).
 TM may **run forever** (does **not halt**) for some $x \notin L(M)$

Three cases

- i) **TM halts and accepts** x $x \in L(M)$
- ii) **TM halts and does not accept** x $x \notin L(M)$
- iii) **TM runs forever** for x $x \notin L(M)$

If $x \in L(M)$, TM always **halts and accepts** x .

If $x \notin L(M)$, TM may **halt and does not accept** x
 or **run forever** (does not halts) for x .

Example 8.2 $L = \{0^n 1^n \mid n \geq 1\}$.

Example 8.4 $m \dot{-} n = \max(m - n, 0)$ monus or proper subtraction

$0^m 10^n \Rightarrow 0^{m \dot{-} n}$. (TM as a **function**)

8.3 Programming Techniques for Turing Machines

8.3.1 Storage in the state

state *exercising control*
 storing symbols

$$Q \Rightarrow Q \times \Gamma$$

Example 8.6 $01^* + 10^*$

$$Q = \{q_0, q_1\} \times \{0, 1, B\}$$

8.3.2 Multiple track

$\delta \subseteq (Q \times \Gamma^n) \times (Q \times \Gamma^n \times \{L, R\})$ *one control and storing n symbols*

Example 8.7 $L = \{wcw \mid w \in \{0, 1\}^+\}$

8.3.3 Subroutine

Example 8.8 Multiply $0^m 10^n 1 \Rightarrow 0^{mn}$.

8.4 Extension to the Basic Turing Machine

Multitape Turing Machine

$$\delta \subseteq (Q \times \Gamma^n) \times (Q \times \Gamma^n \times \{L, R, S\}^n).$$

S: no head move

Theorem 8.9 *Every language accepted by multitape TM is recursively enumerable. $O(n^2)$.*

proof *If multi tape TM M has k tapes, single tape TM N with $2k$ tracks*

*k tracks: **simulate** the contents of k tapes.*

*k tracks: mark the **head position** of k tapes.*

One move in $M =$ two sweeps of in N . $O(n^2)$

left to right sweep

***update** tape contents and head position*

count number of heads to be updated(right bound)

right to left sweep

restore the head position of TM N to the leftmost head position

Nondeterministic Turing Machine

$$\delta: Q \times \Gamma \rightarrow 2^Q \times \Gamma \times \{L, R\}.$$

Theorem 8.11 *If M_N is a nondeterministic TM, then*

there is a deterministic TM M_D such that $L(M_N) = L(M_D)$.

proof *Every nondeterministic moves of M_N , path in the decision tree.*

Assume the degree of the tree $\leq k$.

Let M_D has a two tapes.

tape 1: sequence of choices in the decision tree

tape 2: simulate the content of M_N .

systematically simulate all moves of M_N .

NP

$O(k^n)$

8.5 Restricted Turing Machine

8.5.1 Turing Machine with semi-infinite tapes

Theorem 8.12

A two tracks of semi-infinite tapes simulates a two-way infinite tape.

8.5.2 Multistack machine

a read only input tape

multiple stacks

$$\delta: Q \times \Sigma \times \Gamma^n \rightarrow Q \times \Gamma^* \times \dots \times \Gamma^*.$$

Theorem 8.13 *If L is accepted by a TM, L is accepted by two-stack machine.*

proof *Left of head one stack*
Right of head another stack

8.5.3 Counter Machine

stack machine

stack alphabet = $\{Z_0, X\}$

Z_0 : bottom stack marker

X : # of B 's represents a number

we can test if number is zero

we can not directly test if two numbers are same

Theorem 8.14 *A three-counter machine can simulate TM*

proof *two-stack machine = TM*

Suppose stack vocabulary has $r-1$ symbols.

stack contents: $X_1, X_2, \dots, X_n \leftrightarrow i = X_n r^{n-1} + X_{n-1} r^{n-2} + \dots + X_1$.

r -ry number with LSB on top of the stack

two counter = two stack contents (i)

1. pop X : $i \rightarrow i/r$

$i \rightarrow i - r; \text{ counter } 3 \rightarrow \text{ counter } 3 + 1$

$\text{if } i = 0 \text{ then counter } 3 = i/r \text{ (pop } X)$

2. $\text{change } X \text{ to } Y: i \rightarrow i + Y - X$

3. $\text{push } Y: i \rightarrow ir + Y$

$i \rightarrow i - 1; \text{ counter } 3 \rightarrow \text{ counter } 3 + r$

$\text{if } i = 0 \text{ then counter } 3 = ir$

$\text{counter } 3 \rightarrow \text{ counter } 3 + Y (= ir + Y, \text{ push } Y)$

Theorem 8.15 *A two-counter machine can simulate TM*

Three counter: i, j, k

$m = 2^i 3^j 5^k$.

increase i, j, k : multiply m by 2, 3, or 5

decrease i, j, k : divide m by 2, 3, or 5

test if $i = 0$: divisible by 2: if decrees until zero even numbers or not.

Turing Machine is a number.

Enumeration machine

work tape: move in either direction, read/write any symbol in Γ .

output tape: move right only, write symbols in Σ and # (separator).

1. *Generate $\forall x \in \Sigma^*$ in systematic way (lexicographic order)*
2. *simulates x for M*
3. *If M accepts x , output x*

If x_i runs forever, $x_{i+1} \notin O(M)$, but $x_{i+1} \in L(M)$

time sharing

Instead of simulating M on the input string one at a time, working a few steps in one string and moves to another

pair generator

$(0, 0), (0, 1), (1, 0), (0, 2), (1, 1), (2, 0), \dots, (i, j), \dots$

Simulate for x_i for j -steps

8.6 Turing Machine and Computer

Simulating Turing machine by computer

infinite tape
storage device

Turing Machine(multi tape) as a computer

Arithmetic and Logic Unit

fa(decoder) with multi tapes(registers, PC, ...)

memory

multi tape

input devices

input tapes

output devices

ouput tapes

Turing machine as a program(operational semantics)

A Turing Machine(TM) M is a transition(rewriting) system,

$M = (C, \rightarrow)$ where

1. C is a set of **configurations**(state of memories, numbers),
2. $\rightarrow \subseteq C \times C$.
 $c, c' \in C, c \rightarrow c'$.

The transition system is **deterministic(monogenic)**, if

$\forall c, c_1, c_2 \in C, \text{ if } c \rightarrow c_1 \wedge c \rightarrow c_2, \text{ then } c_1 = c_2.$

Let $I, T \subseteq C$ be two sets of **initial** and **terminate(final)** configurations.

$i \in I, t \in T$, when $i \rightarrow^* t$, (i, t) is the **run** of the transition system

It is usual to arrange that if $t \in T$, then $\exists t' \in C . \exists . t \rightarrow t'$.

If $c \notin T \wedge \exists c' \in C . \exists . c \rightarrow c'$, the configuration c is said to be **stuck**