Chap. 8 Introduction to Turing Machine

8.2 The Turing Machine

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

1. *Q: finite set of* ***states****,*
2. *T: a finite set of* ***input symbols****,*
3. *Γ: a finite set of* ***tape symbols****,$(T \subseteq \Gamma)$*
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, $B \in \Gamma$, $B \notin T$,*
7. *F: a set of* ***final*** *or* ***accepting*** *states, $F \subseteq Q$.*

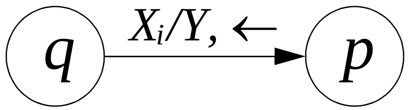
Instantaneous Description *for Turing Machine*

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

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

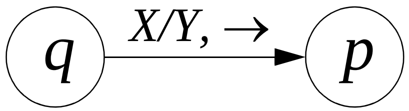
Current tape symbol is X *and assume* $\alpha = \alpha' Z$.

If $(p, \mathbf{Y}, L) \in \delta(q, \mathbf{X_i})$



$(X_1 X_2 \ldots X_{i-1}, q, \mathbf{X_i} X_{i+1} \ldots X_n) \vdash_M^{X_i/Y, \leftarrow} (X_1 X_2 \ldots X_{i-2}, p, \mathbf{X_{i-1}} \mathbf{Y} X_{i+1} \ldots X_n)$

If $(p, \mathbf{Y}, R) \in \delta(q, \mathbf{X_i} X)$ *or*



Then

$(X_1 X_2 \ldots X_{i-1}, q, \mathbf{X_i} X_{i+1} \ldots X_n) \vdash_M^{X_i/Y, \rightarrow} (X_1 X_2 \ldots X_{i-1} \mathbf{Y}, p, \mathbf{X_{i+1}} \ldots X_n)$

If $q \rightarrow^{X/Y, \leftarrow} p$, $(\alpha, q, X\beta) = (\alpha' Z, q, X\beta) \vdash_M^{X_i/Y, \leftarrow} (\alpha', p, \mathbf{Z} Y \beta)$.

If $q \rightarrow^{X/Y, \rightarrow} p$, $(\alpha, q, X\beta) \vdash_M^{X_i/Y, \rightarrow} (\alpha Y, p, \beta)$.

TM also is a finite automaton with read/write tape

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

*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 Chomsky's hierarchy

*We say TM **halt**, if* $\nexists\delta(q, X)$*(and does **not accept**) or* $q \in F$*(and **accepts**).*

*TM may **runs 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*

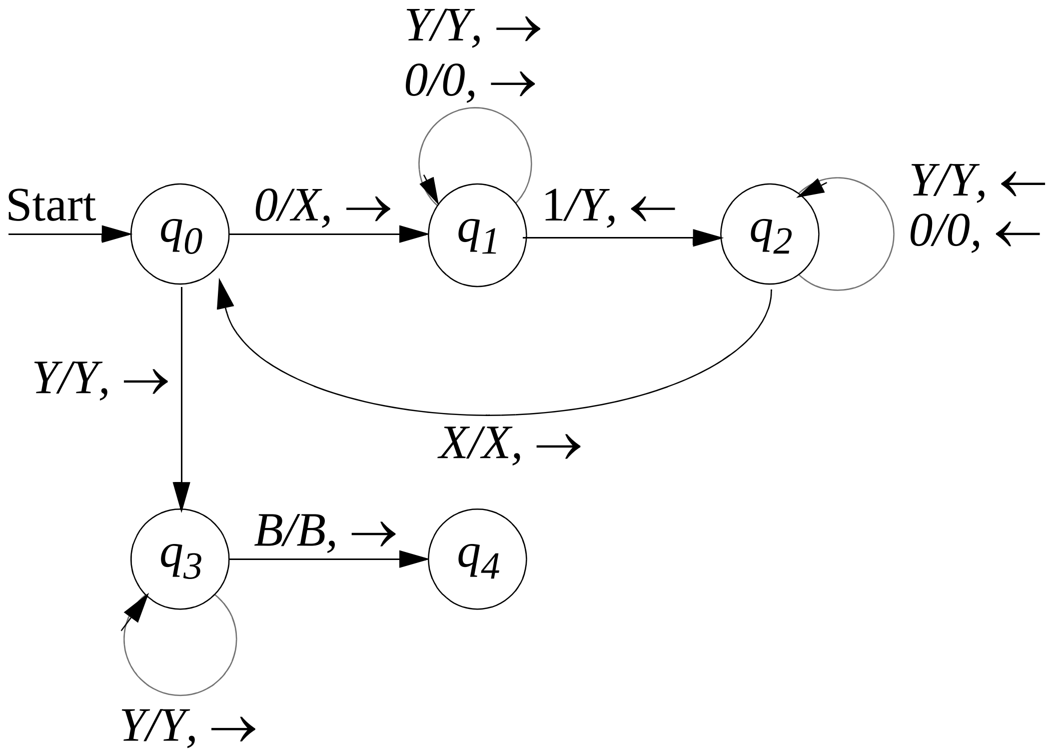
*or TM **runs forever**(does not halts) for x.*

Example 8.2 $L = \{0^n1^n \mid n \geq 1\}$.

$q_0 \rightarrow^{0/X, \rightarrow} q_1$, $q_1 \rightarrow^{0/0, \rightarrow} q_1$, $q_1 \rightarrow^{Y/Y, \rightarrow} q_1$, $q_1 \rightarrow^{1/Y, \leftarrow} q_2$,

$q_2 \rightarrow^{0/0, \leftarrow} q_2$, $q_2 \rightarrow^{Y/Y, \leftarrow} q_2$, $q_2 \rightarrow^{X/X, \rightarrow} q_0$,

$q_0 \rightarrow^{Y/Y, \rightarrow} q_3$, $q_3 \rightarrow^{Y/Y, \rightarrow} q_3$, $q_3 \rightarrow^{B/B, \rightarrow} q_4$.



$(B, q_0, 0011) \Rightarrow^{\rightarrow} (X, q_1, 011) \Rightarrow^{\rightarrow} (X0, q_1, 11) \Rightarrow^{\leftarrow} (X, q_2, 0Y1)$
$\Rightarrow^{\leftarrow} (B, q_2, X0Y1) \Rightarrow^{\rightarrow} (X, q_0, 0Y1) \Rightarrow^{\rightarrow} (XX, q_1, Y1) \Rightarrow^{\rightarrow} (XXY, q_1, 1)$
$\Rightarrow^{\leftarrow} (XX, q_2, YY) \Rightarrow^{\rightarrow} (X, q_2, XYY) \Rightarrow^{\rightarrow} (XX, q_0, YY) \Rightarrow^{\rightarrow} (XXY, q_3, Y)$
$\Rightarrow^{\rightarrow} (XXY, q_3, Y) \Rightarrow^{\rightarrow} (XXYYB, q_4, B)$ ***Accept*** *for 0011.*

$(B, q_0, 0010) \Rightarrow^{\rightarrow} (X, q_1, 010) \Rightarrow^{\rightarrow} (X0, q_1, 10) \Rightarrow^{\leftarrow} (X, q_2, 0Y0)$
$\Rightarrow^{\leftarrow} (B, q_2, X0Y0) \Rightarrow^{\rightarrow} (X, q_0, 0Y0) \Rightarrow^{\rightarrow} (XX, q_1, Y0) \Rightarrow^{\rightarrow} (XXY, q_1, 0)$
$\Rightarrow^{\leftarrow} (XXY0, q_2, B)$ ***Fail*** *for 0010.*

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

$0^m10^n \Rightarrow 0^{m \dot{-} n}$.*(TM as a* ***function****)*

8.3 Programming Techniques for Turing Machines

8.3.1 Storage in the state

state ***exercising*** *control*

storing *symbols* ***Fig*. 8.13**

$Q \Rightarrow Q \times \Gamma$

Exa*. 8.6(p. 338)* $01^* + 10^*$

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

8.3.2 Multiple track(*Fig*. 8.13 *in p. 338)*

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

Exa*. 8.7(p. 339)* $L = \{wcw \mid w \in \{0, 1\}^+\}$

8.3.3 Subroutine

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

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

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

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.*

$O(k^n)$

NP

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

$\delta: Q \times T \times \Gamma^n \to Q \times \Gamma^* \times \ldots \times \Gamma^*$.

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

r-nary number with LSB on top of the stack

two counter = two stack contents (i)

1. pop X: $i \rightarrow i/r$ counter3 := 0;
   while i = 0 **do** i := i - r; counter3 := counter3 +1 **od**
   /* counter3 = i/r (pop X) */
2. change X to Y: i := i + Y - X
3. push Y: $i \rightarrow ir + Y$ counter3 := 0;
   while i = 0 **do** $i \rightarrow i - 1$; counter3 := counter3 + r **od**
   /* counter3 = ir */
   counter3 := counter3 + Y (= ir + Y, push Y)

Theorem 8.15 A two-counter machine can simulate TM

Three counters i, j, k are reperented by one counter $m = 2^i 3^j 5^k$.

1. increase i, j, k: multiply m by 2, 3, or 5
2. test if i, j, k = 0: divisible by 2, 3, 5: if decrease by 2, 3, 5 until zero
3. decrease i, j, k: divide m by 2, 3, or 5

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)*

   $\varepsilon, (a_1, a_2, \ldots, a_n), (a_1a_1, a_1a_2, \ldots, a_na_n), \ldots$
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

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

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 mauti 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*

$^{\forall}c, c_1, c_2 \in C$, *if* $c \rightarrow c_1 \wedge c \rightarrow c_2$, *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* $^{\exists}t' \in C$.∋. $t \rightarrow t'$.

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