TURING MACHINE

Module 4 – Part2

Introduction to TM

- Model of algorithm or computation
- A procedure that can be carried out by human beings/computer can be carried out by a Turing machine
- It provides an ideal theoretical model of a computer.
- Turing machines are useful in several ways.
  - As automaton
  - As computing function
  - Mathematical model for Partial Recursive function
  - determining the undecidability of certain languages
  - measuring the space and time complexity of problems.

Turing Machine Model

- TM has finite control connected to a R/W (read/write) head.
- It has one tape which is divided into a number of cells.

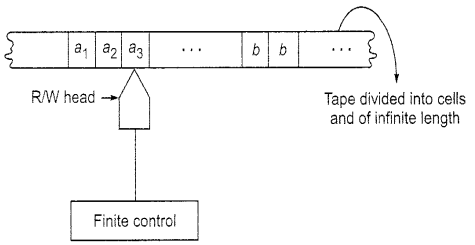


Fig. 9.1 Turing machine model.

TURING MACHINE MODEL

In one step TM does following opeartions

i. a new symbol to be written on the tape in the cell under the R/W head,
ii. a motion of the R/W head along the tape: either the head moves one cell left (L). or one cell right (R),
iii. the next state of the automaton, and
iv. whether to halt or not.

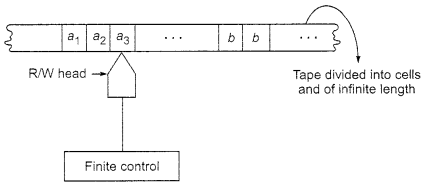


Fig. 9.1 Turing machine model.

Formal definition of TM

Definition 9.1 A Turing machine M is a 7-tuple, namely $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$, where

1. Q is a finite nonempty set of states,
2. Γ is a finite nonempty set of tape symbols,
3. $b \in \Gamma$ is the blank,
4. Σ is a nonempty set of input symbols and is a subset of Γ and $b \notin \Sigma$,
5. δ is the transition function mapping (q, x) onto (q', y, D) where D denotes the direction of movement of R/W head; $D = L$ or R according as the movement is to the left or right.
6. $q_0 \in Q$ is the initial state, and
7. $F \subseteq Q$ is the set of final states.

(1) The acceptability of a string is decided by the reachability from the initial state to some final state. So the final states are also called the accepting states.

(2) δ may not be defined for some elements of $Q \times \Gamma$.

Representation Of Turing Machines

(i) Instantaneous descriptions(ID) using move-relations.
(ii) Transition Table.
(iii) Transition Diagram (transition graph)

1. Instantaneous Descriptions

- An ID of a Turing machine M is a string αβγ, where β is the present state of M.
- The entire input string is split as αγ, the first symbol of γ is the current symbol a under the R/W head and γ has all the subsequent symbols of the input string
- The string α is the substring of the input string formed by all the symbols to the left of a.

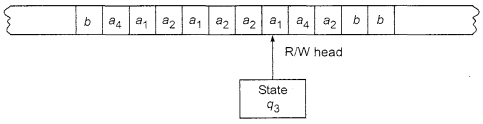


Fig. 9.2 A snapshot of Turing machine.

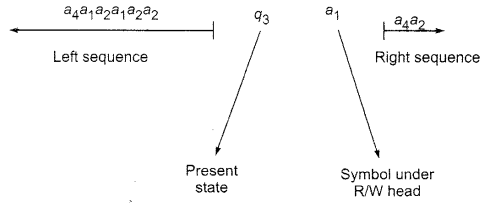


Fig. 9.3 Representation of ID.

(1) For constructing the ID, we simply insert the current state in the input string to the left of the symbol under the R/W head.
(2) Observe that the blank symbol may occur as part of the left or right substring.

Moves In TM

- Suppose $\delta(q, x_i) = (p, y, L)$. The input string to be processed is $x_1x_2 \ldots x_n$, and the present symbol under the R/W head is x_i. So the ID before processing is

$$x_1x_2 \ldots x_{i-1}qx_i \ldots x_n$$

- After processing x_i, ID is

$$x_1 \ldots x_{i-2}\, px_{i-1}yx_{i+1} \ldots x_n$$

- Change of ID is represented by

$$x_1x_2 \ldots x_{i-1}\, q\, x_i \ldots x_n \vdash x_i \ldots x_{i-2}\, px_{i-1}\, yx_{i+1} \ldots x_n$$

- If i = 1, the resulting ID is p y x_2 x_3 .....x_n

- If $\delta(q, X_i) = (p, y, R)$ then change of ID is represented by:

$$x_1x_2 \ldots x_{i-1}q\, x_i \ldots x_n \vdash x_1x_2 \ldots x_{i-1}\, y\, px_{i+1} \ldots x_n$$

- If i = n, the resulting ID is $x_1\, x_2 \ldots x_{n-1}\, y\, p\, b$.

2. Representation By Transition Table

TABLE 9.1 Transition Table of a Turing Machine

Present state	Tape symbol		
	b	0	1
$\rightarrow q_1$	$1Lq_2$	$0Rq_1$	
q_2	bRq_3	$0Lq_2$	$1Lq_2$
q_3		bRq_4	bRq_5
q_4	$0Rq_5$	$0Rq_4$	$1Rq_4$
(q_5)	$0Lq_2$		

TABLE 9.1 Transition Table of a Turing Machine

Present state	Tape symbol		
	b	0	1
$\rightarrow q_1$	$1Lq_2$	$0Rq_1$	
q_2	bRq_3	$0Lq_2$	$1Lq_2$
q_3		bRq_4	bRq_5
q_4	$0Rq_5$	$0Rq_4$	$1Rq_4$
(q_5)	$0Lq_2$		

$q_1 00b \vdash 0q_1 0b \vdash 00q_1 b \vdash 0q_2 01 \vdash q_2 001$
$\vdash q_2 b001 \vdash bq_3 001 \vdash bbq_4 01 \vdash bb_0 q_4 1 \vdash bb_0 1q_4 b$
$\vdash bb010q_5 \vdash bb01q_2 00 \vdash bb0q_2 100 \vdash bbq_2 0100$
$\vdash bq_2 b0100 \vdash bbq_3 0100 \vdash bbbq_4 100 \vdash bbb_1 q_4 00$
$\vdash bbb10q_4 0 \vdash bbb100q_4 b \vdash bbb1000q_5 b$
$\vdash bbb100q_2 00 \vdash bbb10q_2 000 \vdash bbb1q_3 0000$
$\vdash bbbq_5 10000 \vdash bbq_2 b10000 \vdash bbbq_3 10000 \vdash bbbbq_5 0000$

3. Transition Diagram

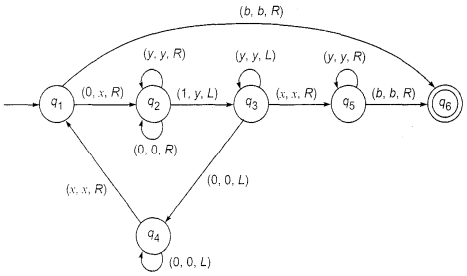


Fig. 9.4 Transition system for *M*.

- Process the string 0011.

3. Transition Diagram

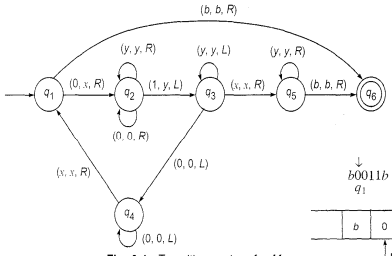


Fig. 9.4 Transition system for *M*.

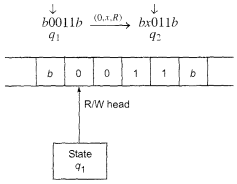


Fig. 9.5 TM processing 0011.

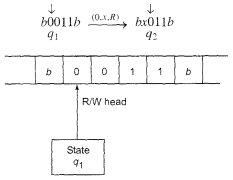
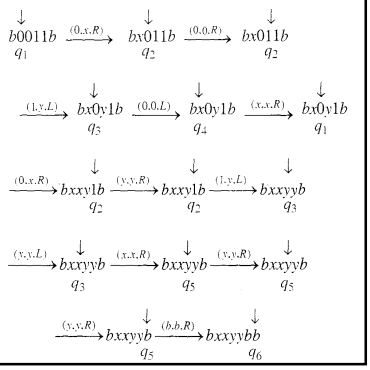


Fig. 9.5 TM processing 0011.



Language Acceptability By Turing Machines

- Consider the Turing machine M = (Q,∑,Γ,δ,q_o,b, F).
- A string w in ∑* is said to be accepted by M if
  $q_o w \vdash \alpha_1 p \alpha_2$ for some p ∈ F and α_1, α_2 ∈ Γ*
- M does not accept w if the machine M either halts in a non accepting state or does not halt.

- Consider the Turing machine *M described by the transition table given in* Table 9.2. Describe the processing of (a) 011 (b) 0011, (c) 001 using IDs. Which of the above strings are accepted by *M?*

TABLE 9.2 Transition Table for Example 9.4

Present state	Tape symbol				
	0	1	x	y	b
→ q_1	xRq_2				bRq_5
q_2	$0Rq_2$	yLq_3		yRq_2	
q_3	$0Lq_4$		xRq_5	yLq_3	
q_4	$0Lq_4$		xRq_1		
q_5				yRq_5	bRq_6
(q6)					

Solution

(a) $q_1011 \vdash xq_211 \vdash q_3xy1 \vdash xq_5y1 \vdash xyq_51$

As $\delta(q_5, 1)$ is not defined, *M* halts; so the input string 011 is not accepted.

(b) $q_10011 \vdash xq_2011 \vdash x0q_211 \vdash xq_30y1 \vdash q_4x0y1 \vdash xq_10y1.$
$\vdash xxq_2y1 \vdash xxyq_21 \vdash xxq_3yy \vdash xq_3xyy \vdash xxq_5yy$
$\vdash xxyq_5y \vdash xxyyq_5b \vdash xxyybq_6$

M halts. As q_6 is an accepting state, the input string 0011 is accepted by *M*.

(c) $q_1001 \vdash xq_201 \vdash x0q_21 \vdash xq_30y \vdash q_4x0y$
$\vdash xq_10y \vdash xxq_2y \vdash xxyq_2$

M halts. As q_2 is not an accepting state, 001 is not accepted by *M*.

Design Of Turing Machines

Design Of Turing Machines

Guidelines:

The fundamental objective in scanning a symbol by the R/W head is to 'know' what to do in the future.

The machine must remember the past symbols scanned. The Turing machine can remember this by going to the next unique state.

The number of states must be minimized.

This can be achieved by changing the states only when there is a change in the written symbol or when there is a change in the movement of the R/W head.

- Example 1: Design a Turing machine to recognize all strings consisting of an even number of 1's.
- **Solution**

(a) q_1 is the initial state. M enters the state q_2 on scanning 1 and writes b.

(b) If M is in state q_2 and scans 1, it enters q_1, and writes b.

(c) q_1 is the only accepting state.

- So M accepts a string if it exhausts all the input symbols and finally is in state q_1. Symbolically,

M = ({q1, q2}, {1, b}, {1, b}, δ, q1, b, {q1})

TABLE 9.3 Transition Table for Example 9.5

Present state	1
$\rightarrow(q_1)$	bq_2R
q_2	bq_1R

Let us obtain the computation sequence of 11. Thus, $q_1 11 \vdash bq_2 1 \vdash bbq_1$. As q_1 is an accepting state, 11 is accepted. $q_1 111 \vdash bq_2 11 \vdash bbq_1 1 \vdash bbbq_2$. M halts and as q_2 is not an accepting state, 111 is not accepted by M.

Example 2:Design a Turing machine over {1. *b} which can compute a concatenation* function over *L = {1}. If a pair of words (w1, w2) is the input, the output has to be w1w2.*

- Solution: w1=11 and w2=111 are separated by blank character b.

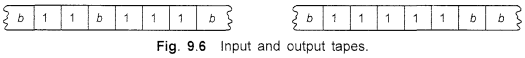


Fig. 9.6 Input and output tapes.

- Main task is to remove the symbol *b*.

(a)The separating symbol b is found and replaced by 1.

(b) The rightmost 1 is found and replaced by a blank *b*.

(c) The R/W head returns to the starting position.

TABLE 9.5 Transition Table for Example 9.6

Present state	Tape symbol	
	1	*b*
$\rightarrow q_0$	$1Rq_0$	$1Rq_1$
q_1	$1Rq_1$	bLq_2
q_2	bLq_3	—
q_3	$1Lq_3$	bRq_f
(q_f)	—	—

TABLE 9.4 Computation for $11b111$

$q_0 11b111 \vdash 1q_0 1b111 \vdash 11q_0 b111 \vdash 111q_1 111$
$\vdash 1111q_1 11 \vdash 11111q_1 1 \vdash 111111q_1 b \vdash 11111q_2 1b$
$\vdash 1111q_3 1bb \vdash 111q_3 11bb \vdash 11q_3 111bb \vdash 1q_3 1111bb$
$\vdash q_3 11111bb \vdash q_3 b11111bb \vdash bq_f 11111bb$

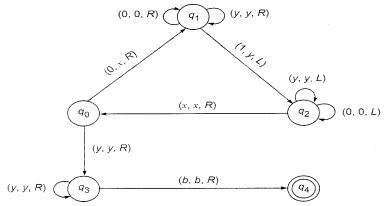
Ex 3: Design a TM to accept L= {0^n1^n: n>=1}

- ***Solution***

(a) If the leftmost symbol in the given input string w is 0, replace it by *x* and move right till we encounter a leftmost 1 in w. Change it to *y and* move backwards.

(b) Repeat (a) with the leftmost 0. If we move back and forth and no 0 or 1 remains, move to a final state.

(c) For strings not in the form 0^n1^n, the resulting state has to be nonfinal.



$q_0 0011 \vdash xq_1011 \vdash x0q_111 \vdash xq_20y1$
$\vdash q_2x0y1 \vdash xq_00y1 \vdash xxq_1y1 \vdash xxyq_11$
$\vdash xxq_2yy \vdash xq_2xyy \vdash xxq_0yy \vdash xxyq_3y$
$\vdash xxyyq_3 = xxyyq_3b \vdash xxyybq_4b$

Hence 0011 is accepted by *M*.

$q_0010 \vdash xq_110 \vdash q_2xy0 \vdash xq_0x0 \vdash xyq_30$

As $\delta(q_3, 0)$ is not defined, *M* halts. So 010 is not accepted by *M*.

Ex 4: design a TM to accept L= {$1^n2^n3^n$:n>=1}

- Before designing the required Turing machine M, let us evolve a procedure for processing the input string 112233. After processing, we require the ID to be of the form bbbbbbq. The processing is done by using five steps:
- Step 1 : q1 is the initial state. The R/W head scans the leftmost 1, replaces 1 by b, and moves to the right. M enters q2.
- step 2: On scanning the leftmost 2, the R/W head replaces 2 by b and moves to the right. M enters q3.
- Step 3 :On scanning the leftmost 3. the R/W head replaces 3 by b, and moves to the right. M enters q4.
- Step 4: After scanning the rightmost 3, the R/W heads moves to the left until it finds the leftmost 1. As a result, the leftmost 1, 2 and 3 are replaced by b.
- Step 5 :Steps 1-4 are repeated until all 1's, 2's and 3's are replaced by blanks.

TABLE 9.6 Transition Table for Example 9.7

Present state	Input tape symbol			
	1	2	3	b
$\rightarrow q_1$	bRq_2			bRq_1
q_2	$1Rq_2$	bRq_3		bRq_2
q_3		$2Rq_3$	bRq_4	bRq_3
q_4			$3Lq_5$	bLq_7
q_5	$1Lq_5$	$2Lq_5$		bLq_5
q_6	$1Lq_5$			bRq_1
(q_7)				

- Write instantaneous descriptions for for 1223, 1233.

- Processing of 112233 is given by:

$q_1112233 \vdash bq_212233 \vdash b1q_22233 \vdash b1bq_3233 \vdash b1b2q_333$
$\vdash b1b2bq_43 \vdash b1b2q_3b3 \vdash b1bq_52b3 \vdash b1q_5b2b3 \vdash bq_51b2b3$
$\vdash q_6b1b2b3 \vdash bq_11b2b3 \vdash bbq_2b2b3 \vdash bbbq_22b3$
$\vdash bbbbq_3b3 \vdash bbbbbq_33 \vdash bbbbbbq_4b \vdash bbbbbq_7bb$

Thus,

$q_1112233 \overset{*}{\vdash} q_7bbbbbb$

Description Of Turing Machines

- Formal description(partial): not defined for all not defined for all *(q, x) by spelling out the current state, the input symbol, the resulting state, the* tape symbol replacing the input symbol and the movement of R/W head to the left or right.
- Implementation description: describes the movement of the head, the symbol stored etc. in English.
- Ex: a single instruction like 'move to right till the end of the input string' requires several moves. A single instruction in the implementation description is equivalent to several moves of a standard TM. At a higher level we can give instructions in English language even without specifying the state or transition function. This is called a *high-level description.*

Techniques for TM Construction

- High-level conceptual tools to make the construction of TMs easier.

There are 4 techniques:

1. Turing machine with stationary head
2. Storage in the state
3. Multiple track turing machine
4. Subroutines

1. Stationary Head

- In the definition of a TM we defined *δ(q, a) as (q', y, D) where D = L or R.*
- Suppose, we want to include the option that the head can continue to be in the same cell for some input symbol. Then we define *δ (q, a) as (q', y, S).*
- This means that the TM, on reading the input symbol a, changes the state to q' and writes y in the current cell in place of a and continues to remain in the same cell.
- In terms of IDs,

  wqax |- wq'yx

- Of course, this move can be simulated by the standard TM with two moves namely

$$wqax \vdash wyq''x \vdash wq'yx$$

That is, $\delta(q, a) = (q', y, S)$ is replaced by $\delta(q, a) = (q'', y, R)$ and $\delta(q'', X) = (q', y, L)$ for any tape symbol X.

Thus in this model $\delta(q, a) = (q', y, D)$ where $D = L, R$ or S.

2.Storage in the State

- State is used in FA or PDA or TM, to 'remember' things.
- We can use a state to store a symbol as well. So the state becomes a pair (q, a) where q is the state and a is the tape symbol stored in (q, a). So the new set of states becomes Q x Γ.
- **EXAMPLE:**
- Construct a TM that accepts the language 0 1* + 1 0*.

- ***Solution***
- We have to construct a TM that remembers the first symbol and checks that it does not appear afterwards in the input string. So we require two states: *q1, q2.*
- The tape symbols are 0, 1 and *b. So the TM, having the "storage facility in* state " is

$$M = (\{q_0, q_1\} \times \{0, 1, b\}, \{0, 1\}, \{0, 1, b\}, \delta, [q_0, b], \{[q_1, b]\})$$

- We describe *δ by its implementation description.*

1. In the initial state, M is in q_0 and has b in its data portion. On seeing the first symbol of the input sting w, M moves right, enters the state q_0 and the first symbol, say *a*, it has seen.

2. M is now in [q1], a).
- (i) If its next symbol is b, M enters [q1, b), an accepting state.
- (ii) If the next symbol is a, M halts without reaching the final state (i.e. δ is not defined).
- (iii) If the next symbol is a` (a` =0 if a =1 and a` =1 if a =0), M moves right without changing state.

3. Step 2 is repeated until M reaches [q1, b) or halts (δ is not defined for an input symbol in w).

3. MULTIPLE TRACK TURING MACHINE

- In a multiple track TM, a single tape is assumed to be divided into several tracks.
- So, tape alphabet Γ is required to consist of k-tuples of tape symbols, *k being the number* of tracks.
- Hence the only difference between the standard TM and the TM with multiple tracks is the set of tape symbols.
- Standard Turing machine:

  Tape symbols are - elements of Γ;
- TM with multiple track:

  Tape symbols are- Γ^k

4. SUBROUTINES

- Subroutine- some task has to be done repeatedly.
- TM program for the subroutine has an initial state and a 'return' state. After reaching the return state, there is a temporary halt.
- For using a subroutine, new states are introduced. When there is a need for calling the subroutine, moves are effected to enter the initial state for the subroutine (when the return state of the subroutine is reached) and to return to the main program of TM.

Ex: Design a TM for performing multiplication of two positive integers.

Solution

The input (m, n), m, n being given, the positive integers are represented by 0^m10^n. M starts with 0^m10^n in its tape. At the end of the computation, 0^{mn}(mn in unary representation) surrounded by b's is obtained as the ouput.

The major steps in the construction are as follows:

1. 0^m10^n1 is placed on the tape (the output will be written after the rightmost 1).
2. The leftmost 0 is erased.
3. A block of n 0's is copied onto the right end.
4. Steps 2 and 3 are repeated m times and 10^m10^{mn} is obtained on the tape.
5. The prefix 10^m1 of 10^m10^{mn} is erased, leaving the product mn as the output.

For every 0 in 0^m, 0^n is added onto the right end. This requires repetition of step 3. We define a subroutine called COPY for step 3.

For the subroutine COPY, the initial state is q_1 and the final state is q_5. δ is given by the transition table (see Table 9.7).

TABLE 9.7 Transition Table for Subroutine COPY

State	Tape symbol			
	0	1	2	b
q_1	$q_2 2R$	$q_4 1L$	—	—
q_2	$q_2 0R$	$q_2 1R$	—	$q_3 0L$
q_3	$q_3 0L$	$q_3 1L$	$q_1 2R$	—
q_4	—	$q_5 1R$	$q_4 0L$	—
q_5	—	—	—	—

The Turing machine M has the initial state q_0. The initial ID for M is $q_0 0^m10^n1$. On seeing 0, the following moves take place (q_6 is a state of M).

$q_0 0^m10^n1 \vdash bq_6 0^{m-1}10^n1 \overset{*}{\vdash} b0^{m-1}q_6 10^n1 \vdash b0^{m-1}1q_1 0^n1$. q_1 is the initial state of COPY. The TM M_1 performs the subroutine COPY. The following moves take place for M_1: $q_1 0^n1 \vdash 2q_2 0^{n-1}1 \overset{*}{\vdash} 20^{n-1}1q_3 b \vdash 20^{n-1}q_3 10 \overset{*}{\vdash} 2q_1 0^{n-1}10$. After exhausting 0's, q_1 encounters 1. M_1 moves to state q_4. All 2's are converted back to 0's and M_1 halts in q_5. The TM M picks up the computation by starting from q_5. The q_0 and q_6 are the states of M. Additional states are created to check whether each 0 in 0^m gives rise to 0^n at the end of the rightmost 1 in the input string. Once this is over, M erases 10^n1 and finds 0^{mn} in the input tape.

M can be defined by

$$M = (\{q_0, q_1, \ldots, q_{12}\}, \{0, 1\}, \{0, 1, 2, b\}, \delta, q_0, b, \{q_{12}\})$$

where δ is defined by Table 9.8.

TABLE 9.8 Transition Table for Example 9.10

	0	1	2	b
q_0	$q_6 bR$	—	—	—
q_6	$q_6 0R$	$q_7 1R$	—	—
q_5	$q_7 0L$	—	—	—
q_7	—	$q_8 1L$	—	—
q_8	$q_8 0L$	—	—	$q_{10} bR$
q_8	$q_8 0L$	—	—	$q_9 bR$
q_{10}	—	$q_{11} bR$	—	—
q_{11}	$q_{11} bR$	$q_{12} bR$	—	—

Thus M performs multiplication of two numbers in unary representation.