8. Turing Machine

Turing formulated a model of algorithm or computation, that is widely accepted. The Church-Turing thesis states that any algorithmic procedure that can be carried out by human beings/computer can be carried out by a Turing machine. It has been universally accepted by computer scientists that the Turing machine provides an ideal theoretical model of a computer.

For formalizing computability, Turing assumed that, while computing, a person writes symbols on a one-dimensional tape which is divided into cells. One scans the cells one at a time and usually performs one of the three simple operations, namely

(i) writing a new symbol in the cell being currently scanned,
(ii) moving to the cell left of the present cell and
(iii) moving to the cell right of the present cell. With these observations in mind, Turing proposed his 'computing machine.'

8.1 Turing Machine Model

The Turing machine can be thought of as finite control connected to a R/W (read/write) head. It has one tape which is divided into a number of cells. The block diagram of the basic model for the Turing machine is given below.

![Diagram of Turing machine model](image)

Each cell can store only one symbol. The input to and the output from the finite state automaton are affected by the R/W head which can examine one cell at a time. In one move, the machine examines the present symbol under the R/W head on the tape and the present state of an automaton to determine

(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.

The above model can be rigorously defined as follows.
**Definition:** A Turing machine $M$ is a 7-tuple, namely $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$, where

- $Q$ is a finite nonempty set of states.
- $\Gamma$ is a finite nonempty set of tape symbols,
- $b$ is the blank.
- $\Sigma$ is a nonempty set of input symbols and is a subset of $\Gamma$ and $b \notin \Sigma$.
- $\delta$ 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.
- $q_0 \in Q$ is the initial state, and
- $F \subseteq Q$ is the set of final states.

### 8.2. Representation of Turing Machines

We can describe a Turing machine employing

(i) Transition diagram (transition graph).

(ii) Instantaneous descriptions using move-relations.

(iii) Transition table

#### 8.2.1. Representation by Instantaneous Descriptions

‘Snapshots’ of a Turing machine in action can be used to describe a Turing machine. These give 'instantaneous descriptions' of a Turing machine. An ID of a Turing machine is defined in terms of the entire input string and the current state.

**Definition:** An ID of a Turing machine $M$ is a string $\alpha \beta \gamma$, where $\beta$ is the present state of $M$, the entire input string is split as $\alpha \gamma$, the first symbol of $\gamma$ is the current symbol $a$ under the R/W head and $\gamma$ has all the subsequent symbols of the input string, and the string $\alpha$ is the substring of the input string formed by all the symbols to the left of $a$.

**Example:** A snapshot of Turing machine is shown in. Obtain the instantaneous description.

**Solution:** The present symbol under the R/W head is $a_1$, the present state is $q_3$ So $a_1$ is written to the right of $q_3$. The nonblank symbols to the left of $a_1$ form the string $a_4a_1a_2a_1a_2$, which is written to the left of $q_3$. The sequence of nonblank symbols to the right of $a_1$ is $a_4a_2$. Thus, the ID is as given in figure given below.
Note: (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) We observe that the blank symbol may occur as part of the left or right substring.

As in the case of pushdown automata, $\delta(q, x)$ induces a change in ID of the Turing machine. We call this change in ID a move.

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 $x_i$ is

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

After processing $x_i$, the resulting ID is

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

This change of ID is represented by

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

If $i = 1$, the resulting ID is $p\ y\ x_2x_3 \ldots x_n$.

If $\delta(q, x_i) = (p, y, R)$, then the change of ID is represented by

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

If $i = n$, the resulting ID is $x_1x_2 \ldots x_{n-1}y \ p \ b$.

We can denote an ID by $I_j$ for some $j$. $I_j \rightarrow I_k$ defines a relation among IDs. So the symbol $\rightarrow$ denotes the reflexive–transitive closure of the relation $\rightarrow$.

In particular, $I_j \rightarrow^n I_j$. Also, if $I_1 \rightarrow^n I_n$, then we can split this as $I_1 \rightarrow I_2 \rightarrow I_3 \rightarrow \ldots \rightarrow I_n$ for some IDs, $I_2, \ldots, I_{n-1}$.

Note: The description of moves by IDs is very much useful to represent the processing of input strings.

**8.2.2. Representation by Transition Table**

We give the definition of $\delta$ in the form of a table called the transition table. If $\delta(q, \ a) = (\gamma, \ \alpha, \ \beta)$, we write $\alpha\beta\gamma$ under the $\alpha$-column and in the $q$-row. So if we get $\alpha\beta\gamma$ in the table, it means that $\alpha$ is written in the current cell, $\beta$ gives the movement of the head (L or R) and $\gamma$ denotes the new state into which the Turing machine enters.
Consider, for example, a Turing machine with five states $q_1, \ldots, q_5$, where $q_1$ is the initial state and $q_5$ is the (only) final state. The tape symbols are 0, 1 and $b$. The transition table given in Table 9.1 describes $\delta$.

**TABLE 9.1 Transition Table of a Turing Machine**

<table>
<thead>
<tr>
<th>Present state</th>
<th>Tape symbol</th>
<th>b</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>$1Lq_2$</td>
<td>0Rq_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td>$bRq_3$</td>
<td>0Lq_2</td>
<td>1Lq_2</td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td></td>
<td>bRq_4</td>
<td>bRq_5</td>
<td></td>
</tr>
<tr>
<td>$q_4$</td>
<td>0Rq_5</td>
<td>0Rq_4</td>
<td>1Rq_4</td>
<td></td>
</tr>
<tr>
<td>$q_5$</td>
<td></td>
<td>0Lq_2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**8.2.3. Representation by Transition Diagram**

The states are represented by vertices. Directed edges are used to represent transition of states. The labels are triples of the form $(\alpha, \beta, \gamma)$, where $\alpha, \beta \in \Gamma$ and $\gamma \in \{L, R\}$. When there is a directed edge from $q_i$ to $q_j$ with label $(\alpha, \beta, \gamma)$, it means that

$\delta(q_i, \alpha) = (q_j, \beta, \gamma)$

During the processing of an input string, suppose the Turing machine enters $q_i$ and the R/W head scans the (present) symbol $\alpha$. As a result, the symbol $\beta$ is written in the cell under the R/W head. The R/W head moves to the left or to the right, depending on $\gamma$, and the new state is $q_j$.

Every edge in the transition system can be represented by a 5-tuple $(q_i, \alpha, \beta, \gamma, q_j)$. So each Turing machine can be described by the sequence of 5-tuples representing all the directed edges. The initial state is indicated by $\rightarrow$ and any final state is marked with $\circ$.

Example:
8.3. Language acceptability by Turing Machines

Let us consider the Turing machine \( M = (Q, \Sigma, \Gamma, \delta, q_0, b, F) \). A string \( w \) in \( \Sigma^* \) is said to be accepted by \( M \) if \( q_0w \xrightarrow{\alpha_1p\alpha_2} \) for some \( p \in F \) and \( \alpha_1, \alpha_2 \in \Gamma^* \).

\( M \) does not accept \( w \) if the machine \( M \) either halts in a nonaccepting state or does not halt.

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

<table>
<thead>
<tr>
<th>Present state</th>
<th>0</th>
<th>1</th>
<th>( \pi )</th>
<th>( y )</th>
<th>( b )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_1 )</td>
<td>( \pi )R( q_2 )</td>
<td></td>
<td></td>
<td></td>
<td>( b )R( q_5 )</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>0R( q_2 )</td>
<td>( y )L( q_3 )</td>
<td></td>
<td>( y )R( q_2 )</td>
<td></td>
</tr>
<tr>
<td>( q_3 )</td>
<td>0L( q_4 )</td>
<td></td>
<td>( \pi )R( q_5 )</td>
<td>( y )L( q_3 )</td>
<td></td>
</tr>
<tr>
<td>( q_4 )</td>
<td>0L( q_4 )</td>
<td></td>
<td>( \pi )R( q_1 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( q_5 )</td>
<td></td>
<td></td>
<td></td>
<td>( y )( y )R( q_5 )</td>
<td>( b )R( q_5 )</td>
</tr>
</tbody>
</table>

**Solution:**

(a) \( q_1 \)011 \( \rightarrow \) \( q_2 \)11 \( \rightarrow \) \( q_2 \)\( xy \)1 \( \rightarrow \) \( q_3 \)\( xy \)1 \( \rightarrow \) \( \pi \)\( q_5 \)1

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

(b) \( q_1 \)0011 \( \rightarrow \) \( q_2 \)011 \( \rightarrow \) \( q_2 \)\( x0 \)11 \( \rightarrow \) \( q_2 \)\( x0 \)\( y \)1 \( \rightarrow \) \( q_2 \)\( x0 \)\( y \)1 \( \rightarrow \) \( q_2 \)1y1.

\( \rightarrow \) \( xyq_2y \) \( \rightarrow \) \( xyq_2y \) \( \rightarrow \) \( xyq_3yy \) \( \rightarrow \) \( q_4x0y \) \( \rightarrow \) \( xyq_4y \) \( \rightarrow \) \( xyq_5yb \) \( \rightarrow \) \( xyvyq_6 \).

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

(c) \( q_1 \)001 \( \rightarrow \) \( q_2 \)01 \( \rightarrow \) \( \pi \)\( q_2 \)1 \( \rightarrow \) \( q_3 \)\( 0y \) \( \rightarrow \) \( q_4x0y \)

\( \rightarrow \) \( q_2 \)\( 0y \) \( \rightarrow \) \( \pi \)\( q_2 \)\( y \) \( \rightarrow \) \( xyq_2 \).

\( M \) halts. As \( q_2 \) is not an accepting state, 001 is not accepted by \( M \).

8.4 Design of Turing Machines

The basic guidelines for designing a Turing machine is given below.

1. 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.

2. 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.
We shall explain the design by a simple example.

**Example-1:**

Design a Turing machine to recognize all strings consisting of an even number of 1's.

**Solution:**

The construction is made by defining moves in the following manner:
(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,

\[
M = (\{q_1, q_2\}, \{1, b\}, \{1, b\}, \delta, q, b, \{q_1\})
\]

where \( \delta \) is defined by Table 9.3.

<table>
<thead>
<tr>
<th>Present state</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_2 )</td>
<td>( bq_2 R )</td>
</tr>
<tr>
<td></td>
<td>( bq_1 R )</td>
</tr>
</tbody>
</table>

Transition diagram

Let us obtain the computation sequence of 11. Thus, \( q_1 11 \leftarrow bq_2 1 \leftarrow bbq_1 \).
As \( q_1 \) is an accepting state, 11 is accepted. \( q_1 111 \leftarrow bq_2 11 \leftarrow bbq_1 1 \leftarrow bbbq_2 \).
\( M \) 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 \( \Sigma = \{1\} \). If a pair of words \((w_1, w_2)\) is the input, the output has to be \( w_1w_2 \).

**Solution**

Let us assume that the two words, \( w_1 \) and \( w_2 \) are written initially on the input tape separated by the symbol \( b \). For example, if \( w_1 = 11, w_2 = 111 \) then the input and output tapes are as shown in Figure.

![Input and output tapes](Fig. 9.6)
Let us assume that the two words $w_1$ and $w_2$ are written initially on the input tape separated by the symbol $b$. For example, if $w_1 = 11$, $w_2 = 111$, then the input and output tapes are as shown in Fig. 9.6.

We observe that the main task is to remove the symbol $b$. This can be done in the following manner:

1. The separating symbol $b$ is found and replaced by 1.
2. The rightmost 1 is found and replaced by a blank $b$.
3. The R/W head returns to the starting position.

We can construct the transition table as follows:

<table>
<thead>
<tr>
<th>Present state</th>
<th>Tape symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$\rightarrow q_0$</td>
<td>1*R$q_0$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>1*R$q_1$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>$b$L$q_3$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>1*L$q_3$</td>
</tr>
<tr>
<td>$q_f$</td>
<td>$-$</td>
</tr>
</tbody>
</table>

The transition diagram is given here.

A computation for $11b111$ is illustrated below.

$q_011b111 \rightarrow 1q_01b111 \rightarrow 11q_0b111 \rightarrow 111q_111$  
$\rightarrow 1111q_111 \rightarrow 11111q_11b \rightarrow 111111q_21b$  
$\rightarrow 111111q_31bb \rightarrow 111111bb \rightarrow 11q_3111bb$  
$\rightarrow q_31111bb \rightarrow q_3b11111bb \rightarrow b_q11111bb$

For the input string $1b1$, the computation sequence is given as

$q_01b1 \rightarrow 1q_0b1 \rightarrow 11q_11 \rightarrow 111q_11b \rightarrow 11q_2b \rightarrow 1q_31bb$  
$\rightarrow q_31bb \rightarrow q_3b11bb \rightarrow b_q11bb$. 
Example-3: Design TM that accepts $\{0^n1^n \mid n \geq 1 \}$

Solution:

We require the following moves:
(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.

Keeping these ideas in our mind, we construct a TM $M$ as follows:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, b, F)$$

where

$$Q = \{q_0, q_1, q_2, q_3, q_f\}$$

$$F = \{q_f\}$$

$$\Sigma = \{0, 1\}$$

$$\Gamma = \{0, 1, x, y, b\}$$

The transition diagram is given in Fig. 9.7. $M$ accepts $\{0^n1^n \mid n \geq 1 \}$.

The moves for 0011 and 010 are given below.

$q_0011 \rightarrow xq_1011 \rightarrow xoq_111 \rightarrow xq_20y1$

$\rightarrow q_2x0y1 \rightarrow qx_0y1 \rightarrow xxq_1y1 \rightarrow xxyq_1$

$\rightarrow xxq_2yy \rightarrow qx_2xyy \rightarrow xxq_0yy \rightarrow xxyq_3y$

$\rightarrow xxyq_3 = xxyyq_3b \rightarrow xxyybq_4b$

Hence 0011 is accepted by $M$.

$q_0010 \rightarrow xq_110 \rightarrow q_2xy0 \rightarrow xq_0y0 \rightarrow xyq_50$

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

---

Prepared by Harivinod N
www.techjourney.in
Example-3: Design TM that accepts $\{1^n2^n3^n \mid n \geq 1\}$

Solution:

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 $bhbhbhb$. The processing is done by using five steps:

**Step 1** $q_1$ is the initial state. The R/W head scans the leftmost 1, replaces 1 by $b$, and moves to the right. $M$ enters $q_2$.

**Step 2** On scanning the leftmost 2, the R/W head replaces 2 by $b$ and moves to the right. $M$ enters $q_3$.

**Step 3** On scanning the leftmost 3, the R/W head replaces 3 by $b$, and moves to the right. $M$ enters $q_4$.

**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>
<thead>
<tr>
<th>Present state</th>
<th>Input tape symbol</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\rightarrow q_1$</td>
<td>$bRq_2$</td>
<td>$bRq_3$</td>
<td>$bRq_4$</td>
<td>$bRq_1$</td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td>$1Rq_2$</td>
<td>$bRq_3$</td>
<td>$bRq_4$</td>
<td>$bRq_2$</td>
<td></td>
</tr>
<tr>
<td>$q_3$</td>
<td>$2Rq_3$</td>
<td>$bRq_4$</td>
<td>$bRq_3$</td>
<td>$bRq_3$</td>
<td></td>
</tr>
<tr>
<td>$q_4$</td>
<td>$3Lq_5$</td>
<td>$bLq_5$</td>
<td>$bLq_5$</td>
<td>$bLq_7$</td>
<td></td>
</tr>
<tr>
<td>$q_5$</td>
<td>$1Lq_6$</td>
<td>$2Lq_5$</td>
<td>$bLq_5$</td>
<td>$bLq_5$</td>
<td></td>
</tr>
<tr>
<td>$q_6$</td>
<td>$1Lq_6$</td>
<td>$2Lq_5$</td>
<td>$bLq_5$</td>
<td>$bLq_5$</td>
<td></td>
</tr>
<tr>
<td>$q_7$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Transition diagram
The change of IDs due to processing of $112233$ is given as:

$q_1112233 \rightarrow b q_211233 \rightarrow b l q_22233 \rightarrow b 1 b q_3233 \rightarrow b 1 b 2 q_333$

$\rightarrow b 1 b 2 b q_43 \rightarrow b 1 b 2 q_5 b 3 \rightarrow b 1 b q_52 b 3 \rightarrow b 1 q_5 b 2 b 3 \rightarrow b q_5 1 b 2 b 3$

$\rightarrow q_6 b 1 b 2 b 3 \rightarrow b q_6 1 b 2 b 3 \rightarrow b b q_6 2 b 3 \rightarrow b b b q_6 2 b 3$

$\rightarrow b b b q_6 3 b 3 \rightarrow b b b b b b q_6 3 b \rightarrow b b b b b b q_7 b b$

Thus,

$q_1112233 \rightarrow q_7 b b b b b$

It can be seen from the table that strings other than those of the form $0^n1^n2^n$ are not accepted.

**Exercise:** Compute the computation sequence for strings like $1223, 1123, 1233$ and then see that these strings are rejected by $M$.

### 8.5 Description of Turing Machines

In the examples discussed so far, the transition function $\delta$ was described as a partial function $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is 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. We can call this a formal description of a TM.

Just as we have the machine language and higher-level languages for a computer, we can have a higher level of description, called the *implementation description*. In this case we describe the movement of the head, the symbol stored etc. in English. For example, 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*.

In the remaining sections of this chapter and later chapters, we give implementation description or high-level description.

### 8.6 Techniques for TM Construction

In this section we give some high-level conceptual tools to make the construction of TMs easier. The Turing machine defined earlier is called the standard Turing machine.
8.6.1. Turing Machine with Stationary Head

In the definition of a TM we defined $\delta(q, a)$ as $(q', y, D)$ where $D = L$ or $R$. So the head moves to the left or right after reading an input symbol. 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 $\delta(q, a)$ 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,

$$wq\delta x \leftarrow wq'yx$$

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

$$wq\delta x \leftarrow w\delta'q'x \leftarrow wq'yx$$

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 $S$.

8.6.2. Storage in the State

We are using a state, whether it is of a FSM 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 (in the usual sense) and $a$ is the tape symbol stored in $(q, a)$. So, the new set of states becomes $Q \times \Gamma$.

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, $q_0, q_1$. 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 $\delta$ 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 string $w$, $M$ moves right, enters the state $q_1$ and the first symbol, say $a$, it has seen.

2. $M$ is now in $[q_1, a]$. (i) If its next symbol is $b$, $M$ enters $[q_1, b]$, an accepting state. (ii) If the next symbol is $a$, $M$ halts without reaching the final state (i.e. $\delta$ is not defined). (iii) If the next symbol is $\overline{a}$ ($\overline{a} = 0$ if $a = 1$ and $\overline{a} = 1$ if $a = 0$), $M$ moves right without changing state.

3. Step 2 is repeated until $M$ reaches $[q_1, b]$ or halts ($\delta$ is not defined for an input symbol in $w$).

8.6.3. Multiple Track Turing Machine

In the case of TM defined earlier, a single tape was used. In a multiple track TM, a single tape is assumed to be divided into several tracks. Now the tape alphabet 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. In the case of the standard Turing machine, tape symbols are elements of \( \Gamma \); in the case of TM with multiple track, it is \( \Gamma^k \). The moves are defined in a similar way.

### 8.6.4. Subroutines

We know that subroutines are used in computer languages, when some task has to be done repeatedly. We can implement this facility for TMs as well.

First a TM program for the subroutine is written. This will have 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 affected 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.

We use this concept to design a TM for performing multiplication of two positive integers.

**Example:** Design a TM which can multiply 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, \(0^m(0^m10^n)\) in unary representation) surrounded by \(b\)'s is obtained as the output.

The major steps in the construction are as follows:

1. \(0^m10^n\) 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 \(10^m\) of \(10^m10^{mn}\) is erased, leaving the product \(mn\) as the output.

For every 0 in \(0^n\), \(0^i\) 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 \(q_5\). \(\delta\) is given by the transition table (see Table 9.7).

<table>
<thead>
<tr>
<th>State</th>
<th>Tape symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>(q_1)</td>
<td>(q_22R)</td>
</tr>
<tr>
<td>(q_2)</td>
<td>(q_70R)</td>
</tr>
<tr>
<td>(q_3)</td>
<td>(q_70L)</td>
</tr>
<tr>
<td>(q_4)</td>
<td>(_)</td>
</tr>
<tr>
<td>(q_5)</td>
<td>(_)</td>
</tr>
</tbody>
</table>
The Turing machine $M$ has the initial state $q_0$. The initial ID for $M$ is $q_00^n10^n1$. On seeing 0, the following moves take place ($q_6$ is a state of $M$).

$q_00^n10^n1 \rightarrow b q_60^{n-1}10^n1 \rightarrow b 0^{n-1}q_610^n1 \rightarrow b 0^{n-1}1q_10^n1$. $q_1$ is the initial state of COPY. The TM $M_1$ performs the subroutine COPY. The following moves take place for $M_1$: $q_10^n1 \rightarrow 2q_20^{n-1}1 \rightarrow 20^{n-1}1q_3b \rightarrow 20^{n-1}q_310 \rightarrow 2q_10^{n-1}10$. 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 $0^n$ 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 $0^{nm}$ 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 $\delta$ is defined by Table 9.8.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$q_6bR$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_6$</td>
<td>$q_6bR$</td>
<td>$q_1R$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_2$</td>
<td>$q_3bL$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_7$</td>
<td></td>
<td>$q_41L$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_8$</td>
<td>$q_5bL$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_9$</td>
<td></td>
<td>$q_6bR$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_{10}$</td>
<td></td>
<td>$q_7bR$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$q_{11}$</td>
<td>$q_8bR$</td>
<td>$q_{12}bR$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Transition Diagram

Thus, $M$ performs multiplication of two numbers in unary representation.