FORMAL LANGUAGES & AUTOMATA THEORY

UNIT- V TURNING MACHINE

7 TURING MACHINES

After going through this chapter, you should be able to understand :

- Turing Machine
- Design of TM
- Computable functions
- Recursively Enumerable languages
- Church's Hypothesis & Counter machine
- Types of Turing Machines

7.1 INTRODUCTION

The Turing machine is a generalized machine which can recognize all types of languages viz, regular languages (generated from regular grammar), context free languages (generated from context free grammar) and context sensitive languages (generated from context sensitive grammar). Apart from these languages, the Turing machine also accepts the language generated from unrestricted grammar. Thus, Turing machine can accept any generalized language. This chapter mainly concentrates on building the Turing machines for any language.

7.2 TURING MACHINE MODEL

The Turing machine model is shown in below figure . It is a finite automaton connected to read - write head with the following components :

- Tape
- Read - write head
- Control unit

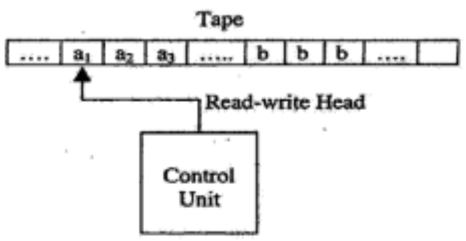


FIGURE : Turing machine model

Tape : It is a temporary storage and is divided into cells. Each cell can store the information of only one symbol. The string to be scanned will be stored from the left most position on the tape. The string to be scanned should end with infinite number of blanks.

Read - write head : The read - write head can read a symbol from where it is pointing to and it can write into the tape to where the read - write head points to.

Control Unit : The reading / writing from / to the tape is determined by the control unit. The different moves performed by the machine depends on the current scanned symbol and the current state. The read - write head can move either towards left or right i.e., movement can be on both the directions. The various moves performed by the machine are :

1. Change of state from one state to another state
2. The symbol pointing to by the read - write head can be replaced by another symbol.
3. The read - write head may move either towards left or towards right.

The Turing machine can be represented using various notations such as

- Transition table
- Instantaneous description
- Transition diagram

7.2.1 Transition Table

The table below shows the transition table for some Turing machine. Later sections describe how to obtain the transition table.

δ	Tape Symbols (Γ)				
States	a	b	X	Y	B
q_0	(q_1, X, R)	-	-	(q_3, Y, R)	-
q_1	(q_1, a, R)	(q_2, Y, L)	-	(q_1, Y, R)	-
q_2	(q_2, a, L)	-	(q_0, X, R)	(q_2, Y, L)	-
q_3	-	-	-	(q_3, Y, R)	(q_4, B, R)
q_4	-	-	-	-	-

Note that for each state q, there can be a corresponding entry for the symbol in Γ. In this table the symbols a and b are input symbols and can be denoted by the symbol Σ. Thus $\Sigma \subseteq \Gamma$ excluding the symbol B. The symbol B indicates a blank character and usually the string ends with infinite number of B's i. e., blank characters. The undefined entries indicate that there are no - transitions defined or there can be a transition to dead state. When there is a transition to the dead state, the machine halts and the input string is rejected by the machine. It is clear from the table that

$$\delta : Q \times \Gamma \ to \ (Q \times \Gamma \times \{ L, R \})$$

where $Q = \{q_0, q_1, q_2, q_3, q_4\}$; $\Sigma = \{ a, b \}$

$\Gamma = \{ a, b, X, Y, B \}$

q_0 is the initial state; B is a special symbol indicating blank character

$F = \{q_4\}$ which is the final state.

Thus, a Turing Machine M can be defined as follows.

Definition : The Turing Machine $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ where

Q is set of finite states

Σ is set of input alphabets

Γ is set of tape symbols

δ is transition function $Q \times \Gamma \ to \ (Q \times \Gamma \times \{L, R\})$

q_0 is the initial state

B is a special symbol indicating blank character

$F \subseteq Q$ is set of final states.

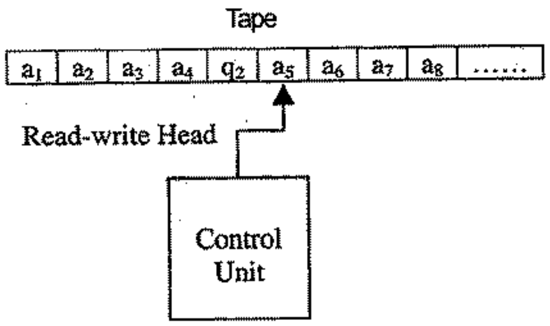
7.2.2 Instantaneous description (ID)

Unlike the ID described in PDA, in Turing machine (TM), the ID is defined on the whole string (not on the string to be scanned) and the current state of the machine.

Definition :

An ID of TM is a string in $\alpha q \beta$, where q is the current state, $\alpha \beta$ is the string made from tape symbols denoted by Γ i. e., α and $\beta \in \Gamma^*$. The read - write head points to the first character of the substring β. The initial ID is denoted by $q\alpha\beta$ where q is the start state and the read - write head points to the first symbol of α from left. The final ID is denoted by $\alpha\beta qB$ where $q \in F$ is the final state and the read - write head points to the blank character denoted by B.

Example : Consider the snapshot of a Turing machine



In this machine, each $a_i \in \Gamma$ (i. e., each a_i belongs to the tape symbol). In this snapshot, the symbol a_5 is under read - write head and the symbol towards left of a_5 i. e., q_2 is the current state. Note that, in the Turing machine, the symbol immediately towards left of the read - write head will be the current state of the machine and the symbol immediately towards right of the state will be the next symbol to be scanned. So, in this case an ID is denoted by

$$a_1a_2a_3a_4q_2a_5a_6a_7a_8.......$$

where the substring $a_1a_2a_3a_4$ towards left of the state q_2 is the left sequence, the substring $a_5a_6a_7a_8.....$ towards right of the state q_2 is the right sequence and q_2 is the current state of the machine. The symbol a_5 is the next symbol to be scanned.

Assume that the current ID of the Turing machine is $a_1a_2a_3a_4q_2a_5a_6a_7a_8......$ as shown in snapshot of example.

Suppose, there is a transition $\delta(q_2, a_5) = (q_3, b_1, R)$

It means that if the machine is in state q_2 and the next symbol to be scanned is a_5, then the machine enters into state q_3 replacing the symbol a_5 by b_1 and R indicates that the read - write head is moved one symbol towards right. The new configuration obtained is

$$a_1a_2a_3a_4b_1q_3a_6a_7a_8.....$$

This can be represented by a move as $a_1a_2a_3a_4q_2a_5a_6a_7a_8.... \vdash a_1a_2a_3a_4b_1q_3a_6a_7a_8.....$

Similarly if the current ID of the Turing machine is $a_1a_2a_3a_4q_2a_5a_6a_7a_8.....$

and there is a transition

$$\delta(q_2, a_5) = (q_1, c_1, L)$$

means that if the machine is in state q_2 and the next symbol to be scanned is a_5, then the machine enters into state q_1 replacing the symbol a_5 by c_1 and L indicates that the read - write head is moved one symbol towards left. The new configuration obtained is

$$a_1a_2a_3q_1a_4c_1a_6a_7a_8......$$

This can be represented by a move as $a_1a_2a_3a_4q_2a_5a_6a_7a_8....\ |\!-\ a_1a_2a_3q_1a_4c_1a_6a_7a_8....$

This configuration indicates that the new state is q_1, the next input symbol to be scanned is a_4. The actions performed by TM depends on

1. The current state.
2. The whole string to be scanned
3. The current position of the read - write head

The action performed by the machine consists of

1. Changing the states from one state to another
2. Replacing the symbol pointed to by the read - write head
3. Movement of the read - write head towards left or right.

7.2.3 The move of Turing Machine M can be defined as follows

Definition : Let $M = (Q,\Sigma,\Gamma,\delta,q_0,B,F)$ be a TM. Let the ID of M be $a_1a_2a_3......a_{k-1}qa_ka_{k+1}......a_n$ where $a_j \in \Gamma$ for $1 \le j \le n-1$, $q \in Q$ is the current state and a_k as the next symbol to scanned. If there is a transition $\delta(q, a_k) = (p, b, R)$

then the move of machine M will be $a_1a_2a_3......a_{k-1}qa_ka_{k+1}.....a_n\ |\!-a_1a_2a_3......a_{k-1}bpa_{k+1}.....a_n$

If there is a transition $\delta(q, a_k) = (p, b, L)$

then the move of machine M will be

$$a_1a_2a_3......a_{k-1}qa_ka_{k+1}....a_n\ |\!-a_1a_2a_3......a_{k-2}pa_{k-1}ba_{k+1}......a_n$$

7.2.4 Acceptance of a language by TM

The language accepted by TM is defined as follows.

Definition :

Let $M = (Q,\Sigma,\Gamma,\delta,q_0,B,F)$ be a TM. The language L(M) accepted by M is defined as

$L(M) = \{w \,|q_0w|\!-\,*\alpha_1\ p\ \alpha_2$ where $w \in \Sigma^*, p \in F$ and $\alpha_1, \alpha_2 \in \Gamma^*\}$

i.e., set of all those words w in $\Sigma*$ which causes M to move from start state q_0 to the final state p. The language accepted by TM is called recursively enumerable language.

The string w which is the string to be scanned, should end with infinite number of blanks. Initially, the machine will be in the start state q_0 with read - write head pointing to the first symbol of w from left. After some sequence of moves, if the Turing machine enters into the final state and halts, then we say that the string w is accepted by Turing machine.

7.2.5 Differences between TM and PDA

Push Down Automa :

1. A PDA is a nondeterministic finite automaton coupled with a stack that can be used to store a string of arbitrary length.
2. The stack can be read and modified only at its top.
3. A PDA chooses its next move based on its current state, the next input symbol and the symbol at the top of the stack.
4. There are two ways in which the PDA may be allowed to signal acceptance. One is by entering an accepting state, the other by emptying its stack.
5. ID consisting of the state, remaining input and stack contents to describe the "current condition" of a PDA.
6. The languages accepted by PDA's either by final state or by empty stack, are exactly the context - free languages.
7. A PDA languages lie strictly between regular languages and CSL's.

Turing Machines :

1. The TM is an abstract computing machine with the power of both real computers and of other mathematical definitions of what can be computed.
2. TM consists of a finite - state control and an infinite tape divided into cells.
3. TM makes moves based on its current state and the tape symbol at the cell scanned by the tape head.
4. The blank is one of tape symbols but not input symbol.
5. TM accepts its input if it ever enters an accepting state.
6. The languages accepted by TM's are called Recursively Enumerable (RE) languages.
7. Instantaneous description of TM describes current configuration of a TM by finite - length string.
8. Storage in the finite control helps to design a TM for a particular language.
9. A TM can simulate the storage and control of a real computer by using one tape to store all the locations and their contents.

7.3 CONSTRUCTION OF TURING MACHINE (TM)

In this section, we shall see how TMs can be constructed.

Example 1 : Obtain a Turing machine to accept the language $L = \{ 0^n 1^n \mid n \geq 1 \}$.

Solution : Note that n number of 0's should be followed by n number of 1's. For this let us take an example of the string $w = 00001111$. The string w should be accepted as it has four zeroes followed by equal number of 1's.

General Procedure :

Let q_0 be the start state and let the read - write head points to the first symbol of the string to be scanned. The general procedure to design TM for this case is shown below :

1. Replace the left most 0 by X and change the state to q_1 and then move the read - write head towards right. This is because, after a zero is replaced, we have to replace the corresponding 1 so that number of zeroes matches with number of 1's.
2. Search for the leftmost 1 and replace it by the symbol Y and move towards left (so as to obtain the leftmost 0 again). Steps 1 and 2 can be repeated.

Consider the situation

$$\begin{array}{c} \text{XX0 0YY11} \\ \uparrow \\ q_0 \end{array}$$

where first two 0's are replaced by Xs and first two 1's are replaced by Ys. In this situation, the read - write head points to the left most zero and the machine is in state q_0. With this as the configuration, now let us design the TM.

Step 1 : In state q_0, replace 0 by X, change the state to q_1 and move the pointer towards right. The transition for this can be of the form

$$\delta(q_0,\ 0) = (q_1,\ X,\ R)$$

The resulting configuration is shown below.

$$\begin{array}{c} \text{XXX0YY11} \\ \uparrow \\ q_1 \end{array}$$

Step 2 : In state q_1, we have to obtain the left - most 1 and replace it by Y. For this, let us move the pointer to point to leftmost one. When the pointer is moved towards 1, the symbols encountered may be 0 and Y. Irrespective what symbol is encountered, replace 0 by 0, Y by Y, remain in state q_1 and move the pointer towards right. The transitions for this can be of the form

$$\delta(q_1,0)=(q_1,0,R)$$
$$\delta(q_1,Y)=(q_1,Y,R)$$

When these transitions are repeatedly applied, the following configuration is obtained.

$$\begin{array}{c} \text{XXX0YY11} \\ \uparrow \\ q_1 \end{array}$$

Step 3 : In state q_1, if the input symbol to be scanned is a 1, then replace 1 by Y, change the state to q_2 and move the pointer towards left. The transition for this can be of the form

$$\delta(q_1,1)=(q_2,Y,L)$$

and the following configuration is obtained.

$$\begin{array}{c}\text{XXX0YYY1}\\ \uparrow \\ q_2\end{array}$$

Note that the pointer is moved towards left. This is because, a zero is replaced by X and the corresponding 1 is replaced by Y. Now, we have to scan for the left most 0 again and so, the pointer was move towards left.

Step 4 : Note that to obtain leftmost zero, we need to obtain right most X first. So, we scan for the right most X. During this process we may encounter Y's and 0's . Replace Y by Y, 0 by 0, remain in state q_2 only and move the pointer towards left. The transitions for this can be of the form

$$\delta(q_2,Y)=(q_2,Y,L)$$
$$\delta(q_2,0)=(q_2,0,L)$$

The following configuration is obtained

$$\begin{array}{c}\text{XXX0YYY1}\\ \uparrow \\ q_2\end{array}$$

Step 5 : Now, we have obtained the right most X. To get leftmost 0, replace X by X, change the state to q_0 and move the pointer towards right. The transition for this can be of the form

$$\delta(q_2,X)=(q_0,X,R)$$

and the following configuration is obtained

$$\begin{array}{c}\text{XXX0YYY1}\\ \uparrow \\ q_0\end{array}$$

Now, repeating the steps 1 through 5, we get the configuration shown below :

$$\begin{array}{c}\text{XXXXYYYY}\\ \uparrow \\ q_0\end{array}$$

Step 6 : In state q_0, if the scanned symbol is Y, it means that there are no more 0's. If there are no zeroes we should see that there are no 1's. For this we change the state to q_3, replace Y by Y and move the pointer towards right. The transition for this can be of the form

$$\delta(q_0,Y)=(q_3,Y,R)$$

and the following configuration is obtained

$$\begin{array}{c}XXXXYYYY\\ \uparrow\\ q_3\end{array}$$

In state q_3, we should see that there are only Ys and no more 1's. So, as we can replace Y by Y and remain in q_3 only. The transition for this can be of the form

$$\delta(q_3,Y)=(q_3,Y,R)$$

Repeatedly applying this transition, the following configuration is obtained.

$$\begin{array}{c}XXXXYYYYB\\ \uparrow\\ q_3\end{array}$$

Note that the string ends with infinite number of blanks and so, in state q_3 if we encounter the symbol B, means that end of string is encountered and there exists n number of 0's ending with n number of 1's. So, in state q_3, on input symbol B, change the state to q_4, replace B by B and move the pointer towards right and the string is accepted. The transition for this can be of the form

$$\delta(q_3,B)=(q_4,B,R)$$

The following configuration is obtained

$$\begin{array}{c}XXXXYYYYBB\\ \uparrow\\ q_4\end{array}$$

So, the Turing machine to accept the language $L=\{a^n\ b^n \mid n\geq 1\}$
is given by $M=(Q,\Sigma,\Gamma,\delta,q_0,B,F)$
where
$Q=\{q_0,q_1,q_2,q_3\}$; $\Sigma=\{0,1\}$; $\Gamma=\{0,1,X,Y,B\}$
$q_0\in Q$ is the start state of machine; $B\in\Gamma$ is the blank symbol.
$F=\{q_4\}$ is the final state.
δ is shown below.

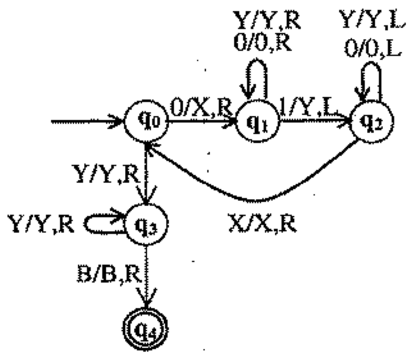
$$\delta(q_0,0) = (q_1,X,R)$$
$$\delta(q_1,0)=(q_1,0,R)$$

$$\delta(q_1,Y)=(q_1,Y,R)$$
$$\delta(q_1,1)=(q_2,Y,L)$$
$$\delta(q_2,Y)=(q_2,Y,L)$$
$$\delta(q_2,0)=(q_2,0,L)$$
$$\delta(q_2,X)=(q_0,X,R)$$
$$\delta(q_0,Y)=(q_3,Y,R)$$
$$\delta(q_3,Y)=(q_3,Y,R)$$
$$\delta(q_3,B)=(q_4,B,R)$$

The transitions can also be represented using tabular form as shown below.

δ	Tape Symbols (Γ)				
States	0	1	X	Y	B
q_0	(q_1, X, R)	-	-	(q_3, Y, R)	-
q_1	$(q_1,0,R)$	(q_2, Y, L)	-	(q_1, Y, R)	-
q_2	$(q_2,0,L)$	-	(q_0, X, R)	(q_2, Y, L)	-
q_3	-	-	-	(q_3, Y, R)	(q_4, B, R)
q_4	-	-	-	-	-

The transition table shown above can be represented as transition diagram as shown below:



To accept the string :

The sequence of moves or computations (IDs) for the string 0011 made by the Turing machine are shown below:

Initial ID

$q_0 0011 \quad \vdash Xq_1011 \quad \vdash X0q_111$

$\vdash Xq_20Y1 \quad \vdash q_2X0Y1$

$\vdash Xq_00Y1 \quad \vdash XXq_1Y1$

$\vdash XXYq_11 \quad \vdash XXq_2YY$

$\vdash Xq_2XYY \quad \vdash XXq_0YY$

$\vdash XXYq_3Y \quad \vdash XXYYq_3$

$\vdash XXYYBq_4$

(Final ID)

Example 2 : Obtain a Turing machine to accept the language $L(M) = \{0^n 1^n 2^n \mid n \geq 1\}$

Solution : Note that n number of 0's are followed by n number of 1's which in turn are followed by n number of 2's. In simple terms, the solution to this problem can be stated as follows :

Replace first n number of 0's by X's, next n number of 1's by Y's and next n number of 2's by Z's. Consider the situation where in first two 0's are replaced by X's, next immediate two 1's are replaced by Y's and next two 2's are replaced by Z's as shown in figure 1(a).

XX00YY11ZZ22	XXX0YY11ZZ22	XXX0YY11ZZ22
↑	↑	↑
q_0	q_1	q_1
(a)	(b)	(c)

FIGURE 1 : Various Configurations

Now, with figure 1(a). a as the current configuration, let us design the Turing machine. In state q_0, if the next scanned symbol is 0 replace it by X, change the state to q_1 and move the pointer towards right and the situation shown in figure 1(b) is obtained. The transition for this can be of the form

$$\delta(q_0, 0) = (q_1, X, R)$$

In state q_1, we have to search for the leftmost 1. It is clear from figure 1(b) that, when we are searching for the symbol 1, we may encounter the symbols 0 or Y. So, replace 0 by 0, Y by Y and move the pointer towards right and remain in state q_1 only. The transitions for this can be of the form

$$\delta(q_1, 0) = (q_1, 0, R)$$

$$\delta(q_1, Y) = (q_1, Y, R)$$

The configuration shown in figure 1(c) is obtained. In state q_1, on encountering 1 change the state to q_2, replace 1 by Y and move the pointer towards right. The transition for this can be of the form

$$\delta(q_1,1)=(q_2,Y,R)$$

and the configuration shown in figure 2(a) is obtained

(a)	(b)	(c)
XXX0YYY1ZZ22	XXX0YYY1ZZ22	XXX0YYY1ZZZ2
↑	↑	↑
q_2	q_2	q_3

FIGURE 2 : Various Configurations

In state q_2, we have to search for the leftmost 2. It is clear from figure 2(a) that, when we are searching for the symbol 2, we may encounter the symbols 1 or Z. So, replace 1 by 1, Z by Z and move the pointer towards right and remain in state q_2 only and the configuration shown in figure 2(b) is obtained. The transitions for this can be of the form

$$\delta(q_2,1)=(q_2,1,R)$$
$$\delta(q_2,Z)=(q_2,Z,R)$$

In state q_2, on encountering 2, change the state to q_3, replace 2 by Z and move the pointer towards left. The transition for this can be of the form

$$\delta(q_2,2)=(q_3,Z,L)$$

and the configuration shown in figure 2(c) is obtained. Once the TM is in state q_3, it means that equal number of 0's, 1's and 2's are replaced by equal number of X's, Y's and Z's respectively. At this point, next we have to search for the rightmost X to get leftmost 0. During this process, it is clear from figure 2(c) that the symbols such as Z's, 1,s, Y's, 0's and X are scanned respectively one after the other. So, replace Z by Z, 1 by 1, Y by Y, 0 by 0, move the pointer towards left and stay in state q_3 only. The transitions for this can be of the form

$$\delta(q_3,Z)=(q_3,Z,L)$$
$$\delta(q_3,1)=(q_3,1,L)$$
$$\delta(q_3,Y)=(q_3,Y,L)$$
$$\delta(q_3,0)=(q_3,0,L)$$

Only on encountering X, replace X by X, change the state to q_0 and move the pointer towards right to get leftmost 0. The transition for this can be of the form

$$\delta(q_3,X)=(q_0,X,R)$$

All the steps shown above are repeated till the following configuration is obtained.

$$\underset{\substack{\uparrow\\q_0}}{\text{XXXXYYYYZZZZ}}$$

In state q_0, if the input symbol is Y, it means that there are no 0's. If there are no 0's we should see that there are no 1's also. For this to happen change the state to q_4, replace Y by Y and move the pointer towards right. The transition for this can be of the form

$$\delta(q_0, Y) = (q_4, Y, R)$$

In state q_4 search for only Y's, replace Y by Y, remain in state q_4 only and move the pointer towards right. The transition for this can be of the form

$$\delta(q_4, Y) = (q_4, Y, R)$$

In state q_4, if we encounter Z, it means that there are no 1's and so we should see that there are no 2's and only Z's should be present. So, on scanning the first Z, change the state to q_5, replace Z by Z and move the pointer towards right. The transition for this can be of the form

$$\delta(q_4, Z) = (q_5, Z, R)$$

But, in state q_5 only Z's should be there and no more 2's. So, as long as the scanned symbol is Z, remain in state q_5, replace Z by Z and move the pointer towards right. But, once blank symbol B is encountered change the state to q_6, replace B by B and move the pointer towards right and say that the input string is accepted by the machine. The transitions for this can be of the form

$$\delta(q_5, Z) = (q_5, Z, R)$$
$$\delta(q_5, B) = (q_6, B, R)$$

where q_6 is the final state.

So, the TM to recognize the language $L = \{ 0^n1^n2^n \mid n \geq 1 \}$ is given by

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

where

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$; $\Sigma = \{0, 1, 2\}$

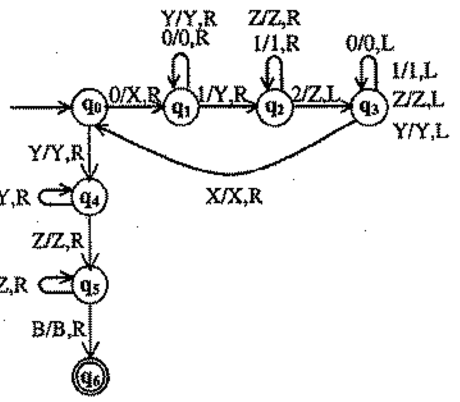
$\Gamma = \{0, 1, 2, X, Y, Z, B\}$; q_0 is the initial state

B is blank character; F = $\{ q_6 \}$ is the final state

δ is shown below using the transition table.

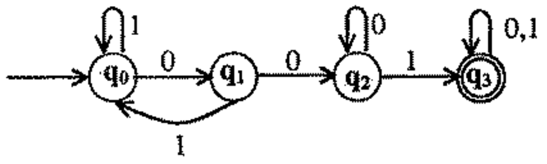
	Γ						
States	0	1	2	Z	Y	X	B
q_0	q_1, X, R				q_4, Y, R		
q_1	q_1, 0, R	q_2, Y,R			q_1, Y,R		
q_2		q_2, 1,R	q_3, Z,L	q_2, Z,R			
q_3	q_3, 0,L	q_3, 1,L		q_3, Z,L	q_3, Y,L	q_0, X,R	
q_4				q_5, Z,R	q_4, Y,R		
q_5				q_5, Z,R			(q_6, B, R)
q_6							

The transition diagram for this can be of the form



Example 3 : Obtain a TM to accept the language $L = \{w \mid w \in (0+1)^*\}$ containing the substring 001.

Solution : The DFA which accepts the language consisting of strings of 0's and 1's having a sub string 001 is shown below :



The transition table for the DFA is shown below :

	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_3
q_3	q_3	q_3

We have seen that any language which is accepted by a DFA is regular. As the DFA processes the input string from left to right in only one direction, TM also processes the input string in only one direction (unlike the previous examples, where the read - write header was moving in both the directions). For each scanned input symbol (either 0 or 1), in whichever state the DFA was in, TM also enters into the same states on same input symbols, replacing 0 by 0 and 1 by 1 and the read - write head moves towards right. So, the transition table for DFA and TM remains same (the format may be different. It is evident in both the transition tables). So, the transition table for TM to recognize the language consisting of 0's and 1's with a substring 001 is shown below :

	0	1	B
q_0	q_1 ,0, R	q_0 , 1, R	-
q_1	q_2 , 0, R	q_0 , 1, R	-
q_2	q_2 , 0, R	q_3 , 1, R	-
q_3	q_3 , 0, R	q_3 , 1, R	q_4 , B, R
q_4			

The TM is given by

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

where

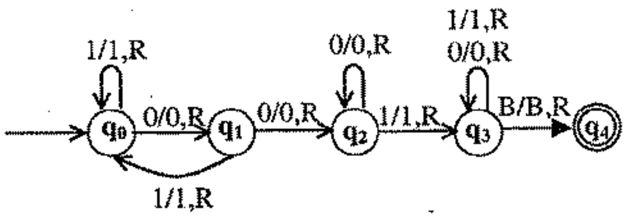
$Q = \{ q_0, q_1, q_2, q_3, q_4 \}$; $\Sigma = \{0, 1\}$

$\Gamma = \{0, 1\}$; δ – is defined already

q_0 is the initial state ; B blank character

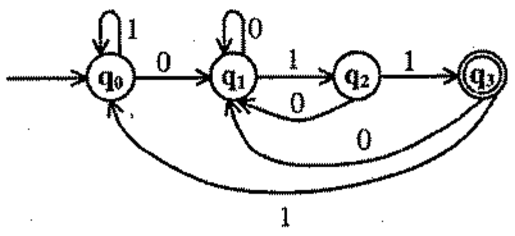
F = { q_4 } is the final state

The transition diagram for this is shown below.



Example 4 : Obtain a Turing machine to accept the language containing strings of 0's and 1's ending with 011.

Solution : The DFA which accepts the language consisting of strings of 0's and 1's ending with the string 001 is shown below :



The transition table for the DFA is shown below :

δ	0	1
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_1	q_0

We have seen that any language which is accepted by a DFA is regular. As the DFA processes the input string from left to right in only one direction, TM also processes the input string in only one direction. For each scanned input symbol (either 0 or 1), in whichever state the DFA was in, TM also enters into the same states on same input symbols, replacing 0 by 0 and 1 by 1 and the read - write head moves towards right. So, the transition table for DFA and TM remains same (the format may be different. It is evident in both the transition tables). So, the transition table for TM to recognize the language consisting of 0's and 1's ending with a substring 001 is shown below :

δ	0	1	B
q_0	q_1,0, R	q_0, 1, R	-
q_1	q_1, 0, R	q_2, 1, R	-
q_2	q_1, 0, R	q_3, 1, R	-
q_3	q_1, 0, R	q_0, 1, R	q_4, B, R
q_4	-	-	-

The TM is given by $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$
where

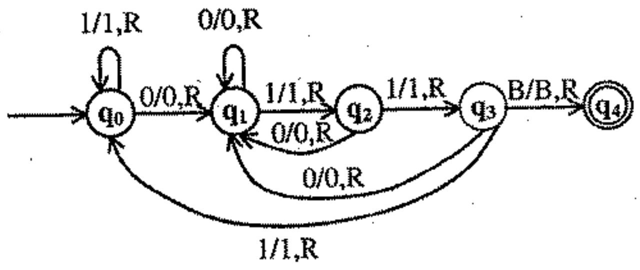
$Q = \{ q_0, q_1, q_2, q_3 \}$; $\Sigma = \{0, 1\}$; $\Gamma = \{0, 1\}$

δ – is defined already

q_0 is the initial state ; B does not appear

F = { q_4 } is the final state

The transition diagram for this is shown below:

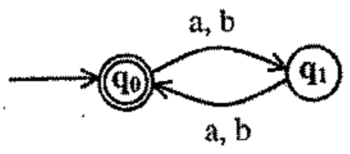


Example 5 : Obtain a Turing machine to accept the language

$L = \{ w | w \text{ is even and } \Sigma = \{ a, b \} \}$

Solution :

The DFA to accept the language consisting of even number of characters is shown below.



The transition table for the DFA is shown below :

	a	b
q_0	q_1	q_1
q_1	q_0	q_0

We have seen that any language which is accepted by a DFA is regular. As the DFA processes the input string from left to right in only one direction, TM also processes the input string in only one direction. For each scanned input symbol (either a or b), in whichever state the DFA was in, TM also enters into the same states on same input symbols, replacing a by a and b by b and the read - write head moves towards right. So, the transition table for DFA and TM remains same (the format may be different). So, the transition table for TM to recognize the language consisting of a's and b's having even number of symbols is shown below :

δ	a	b	B
q_0	q_1, a, R	q_1, b, R	q_2, B, R
q_1	q_0, a, R	q_0, b, R	-
q_2	-	-	-

The TM is given by

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

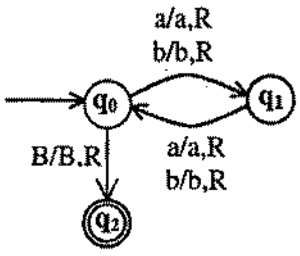
where

Q = { q_0, q_1 }; $\Sigma = \{a, b\}$; $\Gamma = \{a, b\}$

δ – is defined already ; q_0 is the initial state

B does not appear ; F = { q_2 } is the final state

The transition diagram of TM is given by



Example 6 : Obtain a Turing machine to accept a palindrome consisting of a's and b's of any length.

Solution : Let us assume that the first symbol on the tape is blank character B and is followed by the string which in turn ends with blank character B. Now, we have to design a Turing machine which accepts the string, provided the string is a palindrome. For the string to be a palindrome, the first and the last character should be same. The second character and last but one character in the string should be same and so on. The procedure to accept only string of palindromes is shown below. Let q0 be the start state of Turing machine.

Step 1 : Move the read - write head to point to the first character of the string. The transition for this can be of the form $\delta(q_0, B) = (q_1, B, R)$

Step 2 : In state q_1, if the first character is the symbol a, replace it by B and change the state to q_2 and move the pointer towards right. The transition for this can be of the form

$$\delta(q_1, a) = (q_2, B, R)$$

Now, we move the read - write head to point to the last symbol of the string and the last symbol should be a. The symbols scanned during this process are a's, b's and B. Replace a by a, b by b and move the pointer towards right. The transitions defined for this can be of the form

$$\delta(q_2, a) = (q_2, a, R)$$

$$\delta(q_2, b) = (q_2, b, R)$$

But, once the symbol B is encountered, change the state to q_3, replace B by B and move the pointer towards left. The transition defined for this can be of the form

$$\delta(q_2, B) = (q_3, B, L)$$

In state q_3, the read - write head points to the last character of the string. If the last character is a, then change the state to q_4, replace a by B and move the pointer towards left. The transitions defined for this can be of the form

$$\delta(q_3, a) = (q_4, B, L)$$

At this point, we know that the first character is a and last character is also a. Now, reset the read - write head to point to the first non blank character as shown in step5.

In state q_3, if the last character is B (blank character), it means that the given string is an odd palindrome. So, replace B by B change the state to q_7 and move the pointer towards right. The transition for this can be of the form

$$\delta(q_3, B) = (q_7, B, R)$$

Step 3 : If the first character is the symbol b, replace it by B and change the state from q_1 to q_5 and move the pointer towards right. The transition for this can be of the form

$$\delta(q_1, b) = (q_5, B, R)$$

Now, we move the read - write head to point to the last symbol of the string and the last symbol should be b. The symbols scanned during this process are a's, b's and B. Replace a by a, b by b and move the pointer towards right. The transitions defined for this can of the form

$$\delta(q_5, a) = (q_5, a, R)$$
$$\delta(q_5, b) = (q_5, b, R)$$

But, once the symbol B is encountered, change the state to q_6, replace B by B and move the pointer towards left. The transition defined for this can be of the form

$$\delta(q_5, B) = (q_6, B, L)$$

In state q_6, the read - write head points to the last character of the string. If the last character is b, then change the state to q_6, replace b by B and move the pointer towards left. The transitions defined for this can be of the form

$$\delta(q_6, b) = (q_4, B, L)$$

At this point, we know that the first character is b and last character is also b. Now, reset the read - write head to point to the first non blank character as shown in step 5.

In state q_6, If the last character is B (blank character), it means that the given string is an odd palindrome. So, replace B by B, change the state to q_7 and move the pointer towards right. The transition for this can be of the form

$$\delta(q_6, B) = (q_7, B, R)$$

Step 4 : In state q_1, if the first symbol is blank character (B), the given string is even palindrome and so change the state to q_7, replace B by B and move the read - write head towards right. The transition for this can be of the form

$$\delta(q_1, B) = (q_7, B, R)$$

Step 5 : Reset the read - write head to point to the first non blank character. This can be done as shown below.

If the first symbol of the string is a, step 2 is performed and if the first symbol of the string is b, step 3 is performed. After completion of step 2 or step 3, it is clear that the first symbol and the last symbol match and the machine is currently in state q_4. Now, we have to reset the read - write head to point to the first nonblank character in the string by repeatedly moving the head towards left and remain in state q_4. During this process, the symbols encountered may be a or b or B (blank character). Replace a by a, b by b and move the pointer towards left. The transitions defined for this can be of the form

$$\delta(q_4, a) = (q_4, a, L)$$
$$\delta(q_4, b) = (q_4, b, L)$$

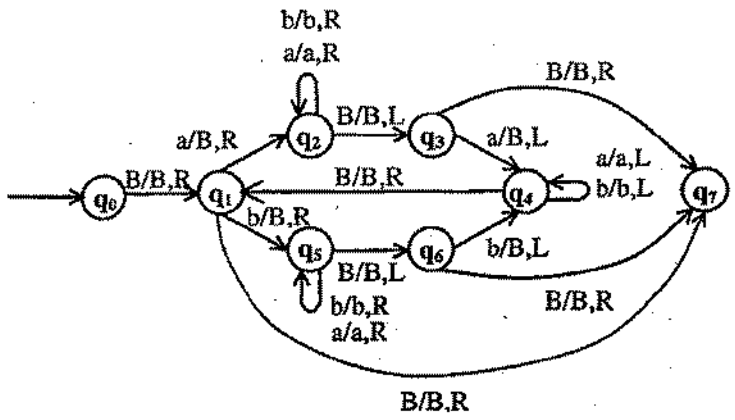
But, if the symbol B is encountered, change the state to q_1, replace B by B and move the pointer towards right. the transition defined for this can be of the form

$$\delta(q_4, B) = (q_1, B, R)$$

After resetting the read - write head to the first non - blank character, repeat through step 1.
So, the TM to accept strings of palindromes over { a, b } is given by $M = (Q, \Sigma, \delta, q_0, B, F)$
where $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$; $\Sigma = \{a, b\}$; $\Gamma = \{a, b, B\}$; q_0 is the initial state
B is the blank character ; F = { q_7 } ; δ is shown below using the transition table

	Γ		
δ	a	b	B
q_0	-	-	q_1, B, R
q_1	q_2, B, R	q_5, B, R	q_7, B, R
q_2	q_2, a, R	q_2, b, R	q_3, B, L
q_3	q_4, B, L	-	q_7, B, R
q_4	q_4, a, L	q_4, b, L	q_1, B, R
q_5	q_5, a, R	q_5, b, R	q_6, B, L
q_6	-	q_4, B, L	q_7, B, R
q_7	-	-	-

The transition diagram to accept palindromes over { a, b } is given by

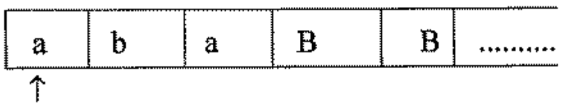


The reader can trace the moves made by the machine for the strings abba, aba and aaba and is left as an exercise.

Example 7 : Construct a Turing machine which accepts the language of aba over $\Sigma = \{a, b\}$.

Solution : This TM is only for L = { aba }

We will assume that on the input tape the string 'aba' is placed like this

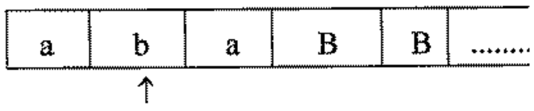


The tape head will read out the sequence upto the B character if 'aba' is readout the TM will halt after reading B.

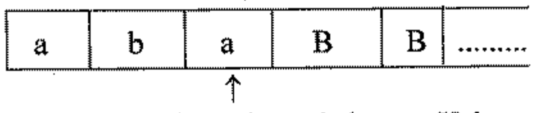
$$\begin{tikzcd}
\arrow[r] & \text{Start} \arrow[r, "{(a,a,R)}"] & q_1 \arrow[r, "{(b,b,R)}"] & q_2 \arrow[r, "{(a,a,R)}"] & q_3 \arrow[d, "{(B,B,S)}"] \\
 & & & & \text{HALT}
\end{tikzcd}$$

The triplet along the edge written is (input read, output to be printed, direction)

Let us take the transition between start state and q_1 is (a, a, R) that is the current symbol read from the tape is a then as a output a only has to be printed on the tape and then move the tape head to the right. The tape will look like this



Again the transition between q_1 and q_2 is (b, b, R). That means read b, print b and move right. Note that as tape head is moving ahead the states are getting changed.



The TM will accept the language when it reaches to halt state. Halt state is always a accept state for any TM. Hence the transition between q_3 and halt is (B, B, S). This means read B, print B and stay there or there is no move left or right. Eventhough we write (B, B, L) or (B, B, R) it is equally correct. Because after all the complete input is already recognized and now we simply want to enter into a accept state or final state. Note that for invalid inputs such as abb or ab or bab ..... there is either no path reaching to final state and for such inputs the TM gets stucked in between. This indicates that these all invalid inputs can not be recognized by our TM.

The same TM can be represented by another method of transition table

	a	b	B
Start	(q_1, a, R)	-	-
q_1	-	(q_2, b, R)	-
q_2	(q_3, a, R)	-	-
q_3	-	-	(HALT, B, S)
HALT	-	-	-

In the given transition table, we write the triplet in each row as :

(Next state, output to be printed, direction)

Thus TM can be represented by any of these methods.

Example 8 : Design a TM that recognizes the set $L = \{0^{2n} 1^n \mid n \geq 0\}$.

Solution : Here the TM checks for each one whether two 0's are present in the left side. If it match then only it halts and accept the string.

The transition graph of the TM is ,

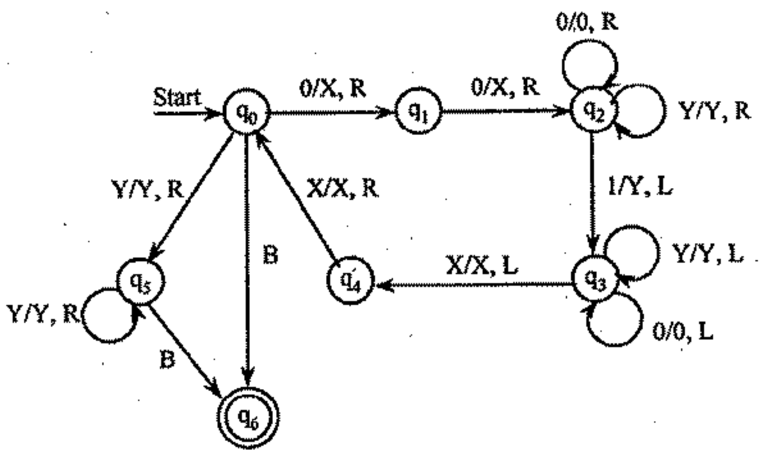


FIGURE : Turing Machine for the given language $L = \{0^{2n} 1^n \mid n \geq 0\}$

Example 9 : Design Turing machine to recognize the palindromes of digits { 0, 1 }. Give its state transition diagram also.

Solution : The construction is made by defining moves in the following manner.

i. The machine scans the first input symbol (either 0 or 1), erases (but remembers) it, writes a blank symbol in place and changes state $(q_1 \text{ or } q_2)$.

ii. It scans the remaining part without changing the tape symbol until it encounters b. It then moves the read / write head a step left. If the rightmost symbol tallies with the leftmost symbol, the rightmost symbol is erased. Otherwise T. M. halts. The read/write head moves to the left until b is encountered.

iii. The above steps are repeated after changing the states suitably.

The transition table is shown below.

Present State	**Tape Symbols**		
	0	**1**	**b**
$\rightarrow q_0$	bRq_1	bRq_2	bRq_7
q_1	$0Rq_1$	$1Rq_1$	bLq_3
q_2	$0Rq_2$	$1Rq_2$	bLq_4
q_3	bLq_5	-	bRq_6
q_4	-	bLq_5	bRq_6
q_5	$0Lq_5$	$1Lq_5$	bRq_6
$\boxed{q_6}$	-	-	-

The transition diagram is shown in below figure.

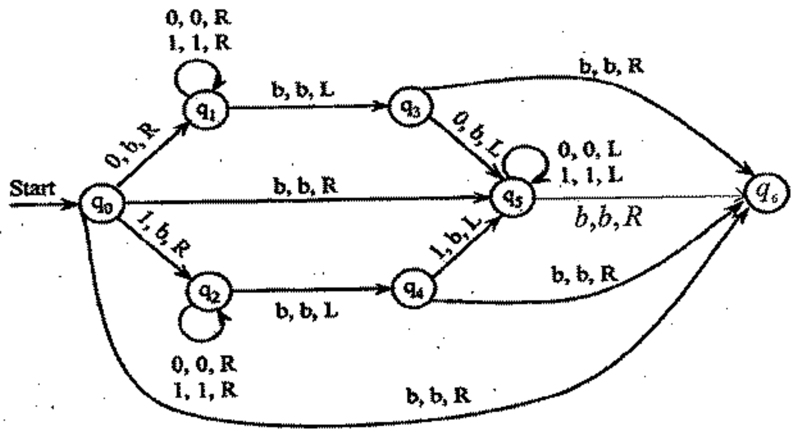


FIGURE : Transition State Diagram for the Palindromes

Example 10 : Design a Turing machine that accepts $L = \{a^n b^n | n \geq 0\}$.

Solution : The logic that we use for the Turing machine to be constructed is,

The Turing machine will remember leftmost a, by replacing it with B, then it moves the tape head right keeping the symbols it scans as it is, until it gets rightmost b, it remembers rightmost b, by replacing it with B, and moves the tape head left keeping the symbols it scans as it is till it reaches the B, on getting B, it moves the tape head one position right and repeats the above cycle if it gets a. If it gets B instead of a, then it is an indication of the fact the string is of the form $a^n b^n$, hence the Turing machine enters into the final state. Therefore, the moves of the Turing machine are given in below table.

	a	b	B
q_0	(q_1, B, R)		(q_4, B, R)
q_1	(q_1, a, R)	(q_1, b, R)	(q_2, B, L)
q_2		(q_3, B, L)	
q_3	(q_3, a, L)	(q_3, b, L)	(q_0, B, R)
q_4			

TABLE : Moves of the Turing Machine for the given language

Therefore, the Turing machine $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{a, b\}, \{a, b, B\}, \delta, q_0, B, \{q_4\})$, where is given above.
The transition diagram corresponding to the above Table is shown in below figure.

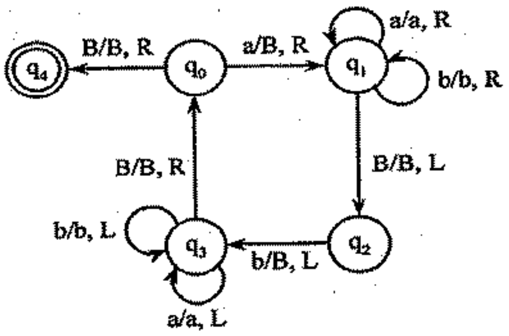


FIGURE : Transition Diagram for the above Table

Example 11 : What does the Turing Machine described by the 5 - tuples, $(q_0,0,q_0,1,R),(q_0,1,q_1,0,r),(q_0,B,q_2,B,R)$, $(q_1,0,q_1,0,R)$, $(q_1,1,q_0,1,R)$ *and* (q_1,B,q_2,B,R). Do when given a bit string as input ?

Solution : The transition diagram of the TM is ,

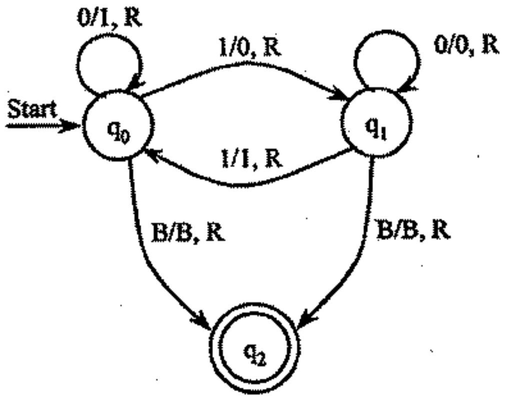


FIGURE : Transition Diagram for the given TM

The TM here reads an input and starts inverting 0's to 1's and 1's to 0's till the first 1. After it has inverted the first 1, it read the input symbol and keeps it as it is till the next 1. After encountering the 1 it starts repeating the cycle by inverting the symbol till next 1. It halts when it encounters a blank symbol.

7.4 COMPUTABLE FUNCTIONS

A Turing machine is a language acceptor which checks whether a string x is accepted by a language L. In addition to that it may be viewed as computer which performs computations of functions from integers to integers. In traditional approach an integer is represented in unary, an integer $i \geq 0$ is represented by the string 0^i.

Example 1 : 2 is represented as 0^2. If a function has k arguments, $i_1, i_2, ...i_k$, then these integers are initially placed on the tape separated by 1's, as $0^{i}10^{i_2}1 .... 10^{i_k}$.

If the TM halts (whether in or not in an accepting state) with a tape consisting of 0's for some m, then we say that $f(i_1, i_2,......i_k) = m$, where f is the function of k arguments computed by this Turing machine.

Example 2 :

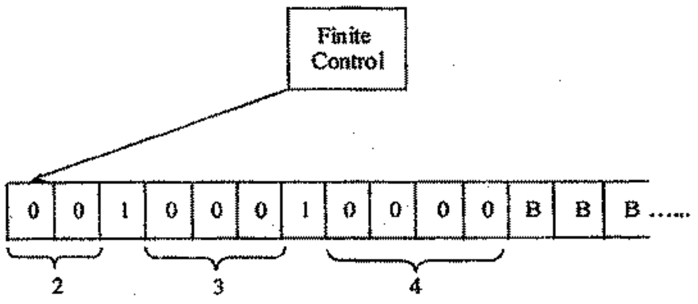
Consider a function in C.

```
int sum ( int x, int y, int z )
{       int s ;
        s = x + y + z ;
        return s;
}
```

Suppose this function is invoked using statement,

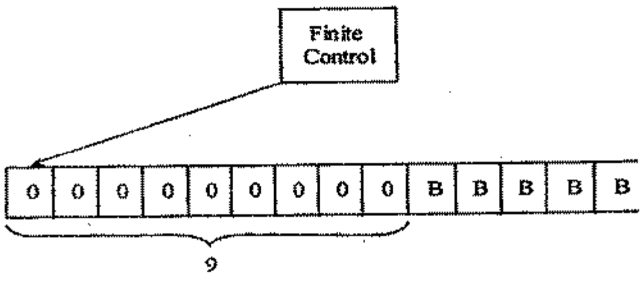
```
        c = sum ( 2, 3, 4 ) ;
```

After invoking sum ( ) , c will have the value 9. The same computation can be performed by Turing machine also. Initially, the Turing machine will have the arguments of sum( ) i.e., 2, 3, 4 on its tape as shown in figure (a).



(a) Before Computation

This Turing machine performs the sum of these arguments. After some moves it halts with the tape containing value 9, as shown in figure (b).



(b) After Computation

FIGURE : Elements on Tape to Compute Sum

Note that a Turing machine may compute a function of one argument, a function of two arguments and so on. The Turing machine given in figure can perform sum of two arguments or three arguments or in general sum of any finite number of arguments.

If TM M computes function f of k arguments i then f need not have a value for all different k - tuples of integers $i_1, i_2, ... i_k$ .if $f(i_1, i_2, ... i_k)$ is defined for all, $i_1 ... i_k$, then we say f is a total recursive function, otherwise we say f is partial recursive function. Total recursive functions are analogues to recursive language because they are computed by TM that always halts. Partial recursive function are analogues to recursively enumerable languages. Because they are computed by TM that may or may not halt. Examples of total recursive functions, all common arithmetic functions on integers, such as multiplication etc, are total recursive functions.

Example 3 : Construct Turing machine to find proper subtraction m - n is defined to be m - n for $m \geq n$ and zero for m < n.

Solution : The TM $M = (\{q_0, q_1, ...q_6\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \phi)$ defined below, started with $0^m / 0^n$ on its tape, halts with 0^{m-n} on its tape. M repeatedly replaces its leading 0 by blank, then searches right for a 1 followed by a 0 and changes the 0 to 1. Next, M moves left until it encounters a blank and then repeats the cycle. The repetition ends if

i. Searching right for a 0, M encounters a blank. Then, the n 0's in $0^m \; 10^n$ have all changed to 1's and n + 1 of the m 0's have been changed to B. M replaces the n + 1 1's by a 0 and n B's leaving m - n 0's on its tape.

ii. Beginning the cycle, M cannot find a 0 to change to a blank, because the first m 0 is already have been changed. Then $n \geq m$. So m - n = 0. M replaces all remaining 1's and 0's by B.

The function δ is described below.

1. $\delta(q_0, 0) = (q, B, R)$

   Begin the cycle, Replace the leading 0 by B.

2. $\delta(q_1, 0) = (q_1, 0, R)$

   $\delta(q_1, 1) = (q_2, 1, R)$

   Search right, looking for the first 1.

3. $\delta(q_2, 1) = (q_2, 1, R)$

   $\delta(q_2, 0) = (q_3, 1, L)$

   Search right past 1's until encountering a 0, change that to 1.

4. $\delta(q_3, 0) = (q_3, 0, L)$

   $\delta(q_3, 1) = (q_3, 1, L)$

   $\delta(q_3, B) = (q_0, B, R)$

   Move left to a blank. Enter state q_0 to repeat the cycle.

5. $\delta(q_2, B) = (q_4, B, L)$

$$\delta(q_4,1) = (q_4, B, L)$$
$$\delta(q_4,0) = (q_4, 0, L)$$
$$\delta(q_4,0) = (q_6, 0, R)$$

If in state q_2 a B is encountered before a 0, we have situation (i) described above. Enter state q_4 and move left, changing all 1's to B 's until encountering a 'B'. This B is changed back to a 0, state q_6 is entered, and M halts.

6. $\delta(q_0,1) = (q_5, B, R)$
$$\delta(q_5,0) = (q_5, B, R)$$
$$\delta(q_5,1) = (q_5, B, R)$$
$$\delta(q_5,B) = (q_6, B, R)$$

If in state q_0 a 1 is encountered instead of a 0, the first block of 0's has been exhausted, as in situation (ii) above. M enters state q_5 to erase the rest of the tape, then enters q_6 and halts.

Example 4 : Design a TM which computes the addition of two positive integers.

Solution : Let TM $M = (Q, \{0, 1, \#\}, \delta, s)$ computes the addition of two positive integers m and n. It means, the computed function f(m, n) defined as follows :

$$f(m,n) = \begin{cases} m + n (If\ m, n \geq 1) \\ 0 \quad (m = n = 0) \end{cases}$$

1 on the tape separates both the numbers m and n. Following values are possible for m and n.

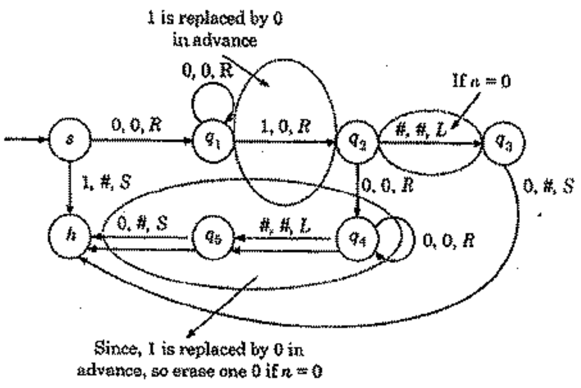
1. $m = n = 0$	(# 1 # ...... is the input),
2. m = 0 and $n \neq 0$	($\#10^n\#$ .... is the input),
3. $m \neq 0$ and n = 0	($\#0^m1\#$ ... is the input), and
4. $m \neq 0$ and $n \neq 0$	($\#0^m10^n\#$ ..... is the input)

Several techniques are possible for designing of M, some are as follows :

(a) M appends (writes) m after n and erases the m from the left end.

(b) M writes 0 in place of 1 and erases one zero from the right or left end . This is possible in case of $n \neq 0$ or $m \neq 0$ only. If m =0 or n = 0 then 1 is replaced by #.

We use techniques (b) given above. M is shown in below figure.



1 is replaced by 0 in advance
0, 0, R
0, 0, R
1, 0, R
If n = 0
#, #, L
1, #, S
0, 0, R
0, #, S
0, #, S
#, #, L
0, 0, R
Since, 1 is replaced by 0 in advance, so erase one 0 if n = 0

FIGURE : TM for addition of two positive integers

7.5 RECURSIVELY ENUMERABLE LANGUAGES

A language L over the alphabet Σ is called recursively enumerable if there is a TM M that accept every word in L and either rejects (crashes) or loops for every word in language L' the complement of L.

Accept (M) = L

Reject (M) + Loop (M) = L'

When TM M is still running on some input (of recursively enumerable languages) we can never tell whether M will eventually accept if we let it run for long time or M will run forever (in loop).

Example : Consider a language (a + b) * bb (a + b) *.

TM for this language is,

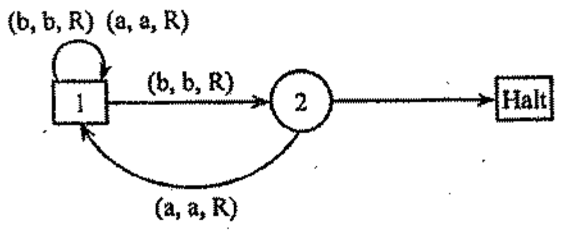


FIGURE : Turing Machine for (a + b) * bb (a + b) *

Here the inputs are of three types.

1. All words with bb = accepts (M) as soon as TM sees two consecutive b's it halts.
2. All strings without bb but ending in b = rejects (M). When TM sees a single b, it enters state2. If the string is ending with b, TM will halt at state 2 which is not accepting state. Hence it is rejected.
3. All strings without bb ending in 'a' or blank 'B' = loop (M) here when the TM sees last a it enters state 1. In this state on blank symbol it loops forever.

Recursive Language

A language L over the alphabet Σ is called recursive if there is a TM M that accepts every word in L and rejects every word in L' i. e.,

accept (M) = L
reject (M) = L'
loop (M) = ϕ.

Example : Consider a language b (a + b) * . It is represented by TM as :

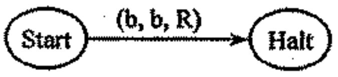


FIGURE : Turing Machine for b (a + b) *

This TM accepts all words beginning with 'b' because it enters halt state and it rejects all words beginning with a because it remains in start state which is not accepting state.

A language accepted by a TM is said to be recursively enumerable languages. The subclass of recursively enumberable sets (r. e) are those languages of this class are said to be recursive sets or recursive language.

7.6 CHURCH'S HYPOTHESIS

According to church's hypothesis, all the functions which can be defined by human beings can be computed by Turing machine. The Turing machine is believed to be ultimate computing machine.

The church's original statement was slightly different because he gave his thesis before machines were actually developed. He said that any machine that can do certain list of operations will be able to perform all algorithms. TM can perform what church asked, so they are possibly the machines which church described.

Church tied both recursive functions and computable functions together. Every partial recursive function is computable on TM. Computer models such as RAM also give rise to partial recursive functions. So they can be simulated on TM which confirms the validity of churches hypothesis.

Important of church's hypothesis is as follows .

1. First we will prove certain problems which cannot be solved using TM.

2. If churches thesis is true this implies that problems cannot be solved by any computer or any programming languages we might every develop.

3. Thus in studying the capabilities and limitations of Turing machines we are indeed studying the fundamental capabilities and limitations of any computational device we might even construct.

It provides a general principle for algorithmic computation and, while not provable, gives strong evidence that no more powerful models can be found.

7.7 COUNTER MACHINE

Counter machine has the same structure as the multistack machine, but in place of each stack is a counter. Counters hold any non negative integer, but we can only distinguish between zero and non zero counters.

Counter machines are off - line Turing machines whose storage tapes are semi - infinite, and whose tape alphabets contain only two symbols, Z and B (blank). Furthermore the symbol Z, which serves as a bottom of stack marker, appears initially on the cell scanned by the tape head and may never appear on any other cell. An integer i can be stored by moving the tape head i cells to the right of Z. A stored number can be incremented or decremented by moving the tape head right or left. We can test whether a number is zero by checking whether Z is scanned by the head, but we cannot directly test whether two numbers are equal.

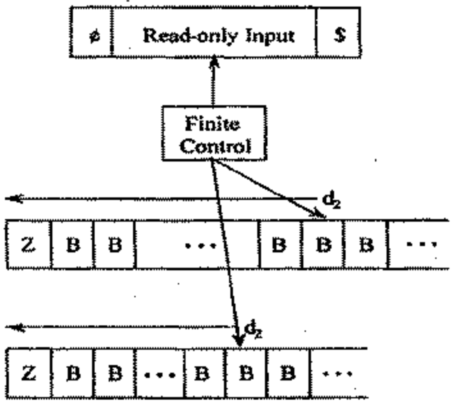


FIGURE : Counter Machine

¢ and $ are customarily used for end markers on the input. Here Z is the non blank symbol on each tape. An instantaneous description of a counter machine can be described by the state, the input tape contents, the position of the input head, and the distance of the storage heads from the symbol Z (shown here as d_1 and d_2). We call these distances the counts on the tapes. The counter machine can only store a count an each tape and tell if that count is zero.

Power of Counter Machines

- Every language accepted by a counter Machine is recursively enumerable.
- Every language accepted by a one - counter machine is a CFL so a one - counter machine is a special case of one - stack machine i. e., a PDA

7.8 TYPES OF TURING MACHINES

Various types of Turing Machines are :

i. With multiple tapes.
ii. With one tape but multiple heads.
iii. With two dimensional tapes.
iv. Non deterministic Turing machines.

It is observed that computationally all these Turing Machines are equally powerful. That means one type can compute the same that other can. However, the efficiency of computation may vary.

1. Turing machine with Two - Way Infinite Tape :

This is a TM that have one finite control and one tape which extends infinitely in both directions.

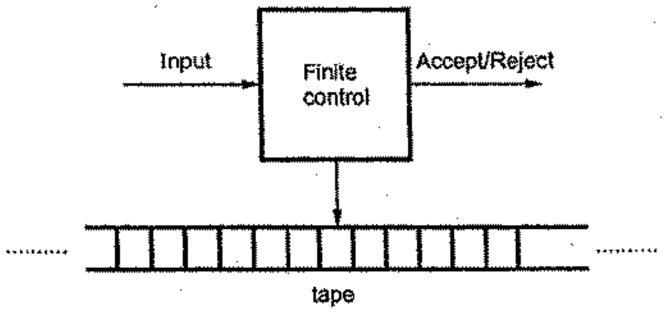


FIGURE : TM with infinite Tape

It turns out that this type of Turing machines are as powerful as one tape Turing machines whose tape has a left end.

2. Multiple Turing Machines :

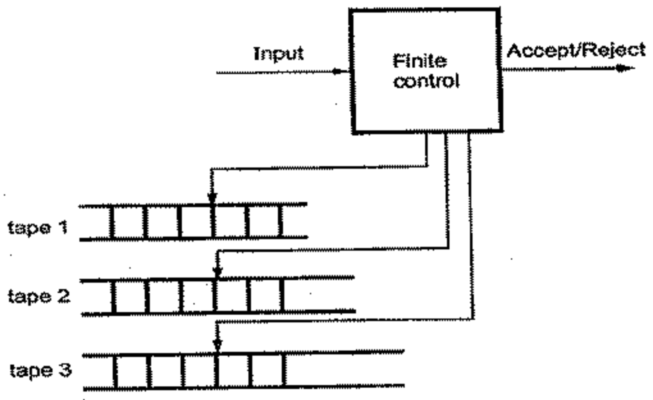


FIGURE : Multiple Turing Machines

A multiple Turing machine consists of a finite control with k tape heads and k tapes, each tape is infinite in both directions. On a single move depending on the state of the finite control and the symbol scanned by each of the tape heads, the machine can

1. Change state.
2. Print a new symbol on each of the cells scanned by its tape heads.
3. Move each of its tape heads, independently, one cell to the left or right or keep it stationary.

Initially, the input appears on the first tape and the other tapes are blank.

3. Nondeterministic Turing Machines :

A nondeterministic Turing machine is a device with a finite control and a single, one way infinite tape. For a given state and tape symbol scanned by the tape head, the machine has a finite number of choices for the next move. Each choice consists of a new state, a tape symbol to print, and a direction of head motion. Note that the non deterministic TM is not permitted to make a move in which the next state is selected from one choice, and the symbol printed and / or direction of head motion are selected from other choices. The non deterministic TM accepts its input if any sequence of choices of moves leads to an accepting state.

As with the finite automaton, the addition of nondeterminism to the Turing machine does not allow the device to accept new languages.

4. Multidimensional Turing Machines :

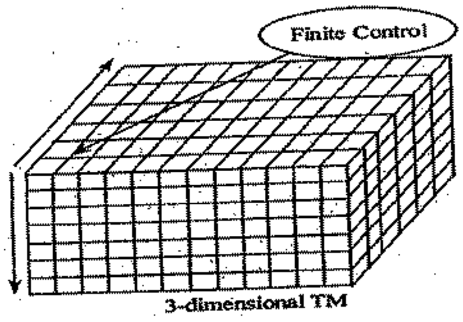


FIGURE : Multidimensional Turing Machine

The multidimensional Turing machine has the usual finite control, but the tape consists of a k - dimensional array of cells infinite in all 2k directions, for some fixed k. Depending on the state and symbol scanned, the device changes state, prints a new symbol, and moves its tape head in one of 2 k directions, either positively or negatively, along one of the k axes. Initially, the input is along one axis, and the head is at the left end of the input.At any time, only a finite number of rows in any dimension contains nonblank symbols, and these rows each have only a finite number of nonblank symbols

5. Multihead Turing Machines :

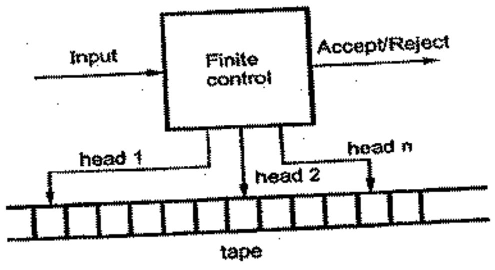


FIGURE : Multihead Turing Machine

A k - head Turing machine has some fixed number, k, of heads. The heads are numbered 1 through k, and a move of the TM depends on the state and on the symbol scanned by each head. In one move, the heads may each move independently left, right or remain stationary.

6. Off - Line Turing Machines :

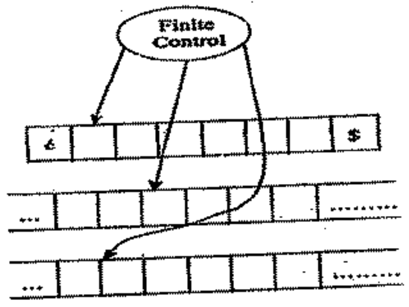


FIGURE : Off - line Turing Machine

An off - line Turing machine is a multitape TM whose input tape is read - only. Usually we surround the input by end markers, ¢ on the left and $ on the right. The Turing machine is not allowed to move the input tape head off the region between ¢ and $.

Off - line TM is just a special case of the multitape TM, and is no more powerful than any of the models we have considered. Conversely, an off - line TM can simulate any TM M by using one more tape than M. The first thing the off - line TM does is copy its own input onto the extra tape, and it then simulates M as if the extra tape were M's input.

7. Multistack Machines :

A deterministic two - stack machine is a deterministic Turing machine with a read only input and two storage tapes. If a head moves left on either tape, a blank is printed on that tape.

Multistack machine and counter machines are restricted Turing machines equivalent to the basic model.

7.9 COMPARISON OF FM, PDA AND TM

Basically have discussed three models viz. finite automata or finite machines (FM), Pushdown automata (PDA) and Turing machine (TM). We will now discuss the comparison between these models,

1. The finite machine is of two types - deterministic finite state machine and non deterministic finite state machine. Both of these DFA and NFA accept regular language only. Hence both the machines have equal power i. e. DFA = NFA.
2. We have then learn push down automata again, pushdown automata consists of two types of models deterministic PDA and Non deterministic PDA. The advantage of PDA over FA is that PDA has a memory and hence PDA accepts large class of languages than FA. Hence PDA has more power than FA. The non deterministic PDA accepts the language of context free grammar power of DPDA is less than NPDA as NPDA accepts a larger class of CFL.
3. The class of two stack or n - stack PDA has more power than one stack DPDA or NPDA. Hence two - stack / n - stack PDAS are more powerful.
4. Turing machines can be programmed. Hence TM accepts very very large class of languages. TM, therefore is the most powerful computational model.

TM > PDA > FM

TM accepts regular and non - regular languages ; context free and context sensitive languages as well.