



Q. 8 Consider the following productions :

$$S \rightarrow aB \mid bA$$
$$A \rightarrow as \mid bAAa$$
$$B \rightarrow bs \mid aBB \mid b$$

For the string aaababbba find

1. left most derivation
2. right most derivation
3. parse tree

Q. 9 Show that grammar is ambiguous

$$G : S \rightarrow alabSblaAb$$
$$A \rightarrow bSlaAAb$$

Q. 10 Show that grammar is ambiguous

$$G : S \rightarrow aBlab$$
$$A \rightarrow aAB \mid a$$
$$B \rightarrow ABblb$$

CHAPTER

5

Turing Machines

University Prescribed Syllabus

Turing Machines : Turing Machine Definition, Representations, Acceptability by Turing Machines, Designing and Description of Turing Machines, Turing Machine Construction, Variants of Turing Machine

Linear Bound Automata : The Linear Bound Automata Model, Linear Bound Automata and Languages

Pushdown Automata : Definitions, Acceptance by PDA, PDA and CFG

5.1 Introduction to Turing Machine

- In 1936, Mr. Alan Turing introduced advanced mathematical model for modern digital computer which was known as Turing machine. Turing machine is advanced machine of FA and PDA.
- This simple mathematical model has no difference between input and output set. This mathematical model can be constructed to accept a given language or to carry out some algorithm.
- This model sometimes uses it's own output as input for further computation.
- This machine or model can select current location and also decides location of memory by moving left or right. This mathematical model known as Turing machine has become an effective procedure.

In short Turing machine is equal to,

$$\star \quad \text{TM} = \text{FA} + \text{Tape}$$

$$\text{FA} = \text{Finite Automata}$$

Tape

- ✓ 1. Infinite memory unit
- ✓ 2. Infinite cell (Each cell contains only one alphabet at one time)
- ✓ 3. Head of records move left or right.

Turing machine's mathematical model consists of **Head** (moves right or left or stays in position), **infinite tape** and **finite set of states**.

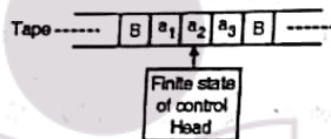


Fig. 5.1.1

Working of Turing machine can be explained as follows,

where, one move of TM shows

- ✓ 1. Changing state location
- ✓ 2. Replacing old symbol by new
- ✓ 3. Left or right move of head

A simple diagrammatic representation for Turing machine construction. As follows

Let's print the symbol 011 on initially blank tape.

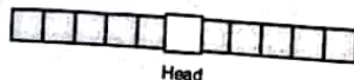


Fig. 5.1.2

First we write 0 in Head

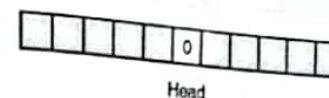


Fig. 5.1.3

Second we move the tape left by Head

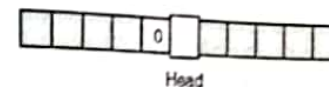


Fig. 5.1.4

Third we write 1 for Head

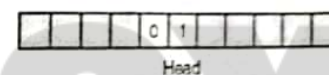


Fig. 5.1.5

Fourth we move left and insert 1

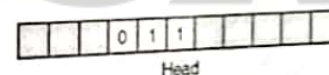


Fig. 5.1.6

- Let's see another one program for bit inversion in above tape (convert 0's and 1's vice-versa)

Simple instructions for program.

Read Symbol	Instructions to write	Instructions to move
Blank	None	None
1	Write 0	R
0	Write 1	R

The machine checks head and reads the symbol under the read / write new symbol and move left or right. Let's see it step by step.



Fig. 5.1.7

Change the head symbol by 0

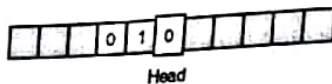


Fig. 5.1.8

Move right and check the head its again 1

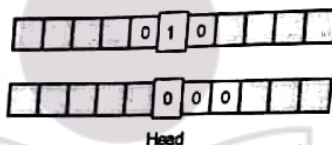


Fig. 5.1.9

Symbol read by head and repeat the same instruction

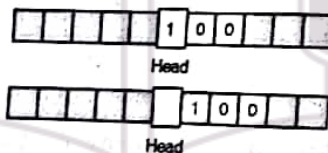


Fig. 5.1.10

hen the Blank symbol reads machine will be in halt state.

Syllabus Topic : Turing Machine Definition

Definition of Turing Machine

Define Turing Machine.

M define TM with seven Tuple,

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

Where,

Q = Finite set of states

Γ = Finite set of tape symbols

$B \in \Gamma$ = Blank space symbol

Σ = Set of input symbols where $\Sigma \subseteq \Gamma - \{B\}$
(excluding blank symbols)

δ = Function for next move $[Q \times \Gamma \rightarrow Q \times \Gamma \times \{S, R\}]$

q_0 = Start state

$F \subseteq Q$ = Final state set

Syllabus Topic : Representations

5.1.2 Representations of Turing Machine

How Turing machine can be represented ?

Turing machine can be represented by ID equations table or diagram.

- (1) Instantaneous Description (ID)
- (2) Transition Table
- (3) Transition Graph

1. Instantaneous Description (ID)

Let $\delta(q_0, a)$ be transition function then ID will be given as,

$$\delta(q_0, a) \rightarrow (q_1, x, R)$$

q_0 is initial state and read 'a' alphabet, transition function δ gives q_1 next state, a is replaced by x and move to the (R) right.

We denote ID of TM by $\alpha_1 q_0 \alpha_2$. q_0 is current state of M in Q. α_1 and α_2 are the strings of non blank symbols upto right most or left most head.



- Consider TM shown in the Fig. 5.1.11 obtain its ID

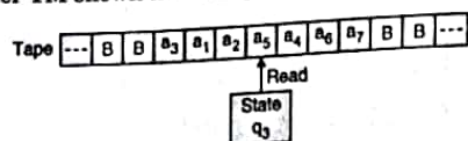


Fig. 5.1.11

- The symbol under head is a_5 and current state is q_3 , so a_5 is written to the right of q_3 . The non blank symbols to the left of a_5 is a string $a_3 a_1 a_2$ which is written to the left of q_3 .
- The sequence of non blank symbols to the right of a_5 is $a_4 a_6 a_7$, Thus ID is $a_3 a_1 a_2 q_3 a_5 a_4 a_6 a_7$ then, $\alpha_1 = a_3 a_1 a_2$ and $\alpha_2 = a_5 a_4 a_6 a_7$, $\alpha_1 q_3 \alpha_2$
- The head can be moved left or right, which can be explained as follows :

(a) TM moves for left

The $\delta(q_0, a_i)$ induces a change in ID of the TM, calling a move.

Let $a_1 a_2 a_3 \dots a_n$ is an input string to be processed and present symbol under tube head is a_i .

Let $\delta(q_0, a_i) = (q_1, y, L)$

ID on a : Before processing will be, $a_1 a_2 a_3 \dots a_{i-1} q_0 a_i a_{i+1} \dots a_n$

ID on a : After processing will be, $a_1 a_2 a_3 \dots a_{i-2} q_1 a_{i-1} y a_{i+1} \dots a_n$

This change of ID is represented as,

$a_1 a_2 a_3 \dots a_{i-1} q_0 a_i \dots a_n \vdash a_1 \dots a_{i-2} q_1 a_{i-1} y a_{i+1} \dots a_n$

(b) TM moves on right

$\delta(q_0 a_i) = (q_1, y, R)$

then ID is represented by,

$a_1 a_2 \dots a_{i-1} q_0 a_i \dots a_n \vdash a_1 a_2 \dots a_{i-1} y q_1 a_{i+1} \dots a_n$

Note : \vdash^* symbol denotes the reflexive transitive closure of relation \vdash .



2. Representation by Transition Table

- We can define δ in the form of table is denoted as transition table.

	a
q_0	x, q_1 , R

- Transition function δ gives a alphabet from q_0 stated replaced by x, next state is q_1 and move to the right.

3. Representation of Transition Graph/Diagram

We can use transition diagram to represent Turing machine. Directed edges are used to represent transition of states.



Fig. 5.1.12

$L(M) = \{w \mid w \in \Sigma^* \text{ and } q_0 \vdash^* q_i \text{ for } q_i \in F, i \geq 1\}$

Where, q_0 and q_1 are transition states
 a is current alphabet in insert tape
 x is replacing alphabet
 R means move to the right

E.g. Construct TM for string having even number of 1's over $\Sigma = \{1\}$.



Fig. 5.1.13

B is blank symbol.

Syllabus Topic : Acceptability by Turing Machine

5.1.3 Language Accepted by Turing Machine

- The language accepted by TM is
 $L(M) = \{w \mid w \in \Sigma^* \text{ and } q_0 \vdash^* q_i \text{ for some } q_i \in F \text{ and } \alpha_1 \alpha_2 \in \Gamma^*\}$

- Accepting states of machine have no outgoing transition's.
- Let M be Turing machine represented as,

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$
- For this machine string w is set of input Σ^* of M is not accepted if M does not halt or halt in non accepting areas.
- The string w of set of input Σ^* is said to be acceptable by M only if

$$q_0 w \vdash^* \alpha_1 P \alpha_2 \text{ for } P \in F \text{ and } \alpha_1, \alpha_2 \in \Gamma^*$$

Acceptance of string by Turing machine

Accept IP string = If machine halts in an **accept state**

Reject IP string = If machine halts in a **non accept state** or
 If machine centers an **infinite loop**.

Syllabus Topic : Designing and Description of Turing Machines

5.2 Designing and Description of Turing Machine

5.2.1 Design of Turing Machine

Explain design of Turing Machine.

1. Despite their simplicity R/W Turing machines are very powerful computing devices.
2. Head of Turing machine scans the symbol and machine remembers it, for future.
3. Minimization of number of states is activity 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.
4. Turing machine works as mathematical model for solving complexity of language recognition problem.
5. TM is used for solving arithmetic operations, problems.
6. Diagram of language recognizing by Turing machine.

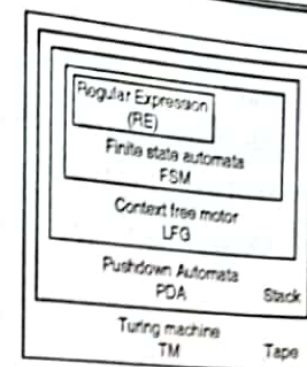


Fig. 5.2.1

All Regular Expressions and Context Free language are recognized by a particular type of machine, regular and not context free are also accepted by Turing machine.

Let's see the designing of Turing machine with examples.

Example 5.2.1 : Design Turing machine which can accept languages 0^*1^*

Solution :

1. $L = \{\epsilon, 0, 00, 000, \dots, 001, 0011, 00111, \dots, 0001, \dots\}$
2. Scan all zero's using state q_0 .
3. If string S ends accepts string, otherwise scan all 1's using state q_1 in this state, it will not accept any 0's

Turing machine is given below,

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

$\delta =$ Transition function

Transition Table

State	0	1	B
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	q_{acc}
q_1	-	$(q_1, 1, R)$	q_{acc}
q_{acc}			Final

4. Transition function

$$\delta(q_0, 0) = (q_0, 0, R) \quad \delta(q_0, 1) = (q_1, 1, R)$$

$$\delta(q_1, 1) = (q_1, 1, R) \quad \delta(q_0, B) = q_{acc} \quad \delta(q_1, B) = q_{acc}$$

q_0 = It is initial state, $q_0 \in Q$

B = Blank $\in \Gamma$ it is used to mention start of string as well as end of string

F = It is finite set of the final state

$$F \subseteq Q \setminus \{q_{acc}\}$$

5. ID for 000111

In tape, string is B000111BB

$\rightarrow B q_0 000111BB \rightarrow B0 q_0 00111BB \rightarrow B00 q_0 0111BB$

$\rightarrow B000 q_0 111BB \rightarrow B0001 q_1 11BB \rightarrow B00011 q_1 1BB$

$\rightarrow B000111 q_1 BB \rightarrow B000111 q_{acc} B \rightarrow B000111 q_{acc} \rightarrow \text{Accept}$

6. ID for 000110B

In tape string B000110BB

$\rightarrow B q_0 000110BB \rightarrow B0 q_0 00110BB \rightarrow B00 q_0 0110BB$

$\rightarrow B000 q_0 110BB \rightarrow B0001 q_1 10BB \rightarrow B00011 q_1 0BB \rightarrow \text{Error state}$

$\rightarrow \text{Not accept}$

Example 5.2.2 : Design Turing machine which can accept language $0(0+1)^*1$.

Solution :

1. $L = \{01, 001, 011, 0101, 0011 \dots\}$
2. State q_0 is used to check whether string started with 0 or not? (1 is not the 1st symbol)
3. State q_1 and q_2 are used to assure incoming symbol and check last one symbol is 1 or not? (0 is not the last symbol)
4. State q_{acc} is final state.

Turing machine is given below :

$$Q = \{q_0, q_1, q_2, q_{acc}\}$$

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

$$\Gamma = \{0, 1, B\}$$

δ = Transition function

Transition function table

State	0	1	B
q_0	$(q_1, 0, R)$	-	-
q_1	$(q_1, 0, R)$	$(q_2, 1, R)$	-
q_2	$(q_1, 0, R)$	$(q_2, 1, R)$	q_{acc}
q_{acc}			Final

Transition function

$$(q_0, 0) = (q_1, 0, R), (q_1, 0) = (q_1, 0, R), (q_1, 1) = (q_2, 1, R)$$

$$(q_2, 0) = (q_1, 0, R), (q_2, 1) = (q_2, 1, R), (q_2, B) = q_{acc}$$

ID for 0101 string \rightarrow In tape, string is B0101BB

$\rightarrow B q_0 0101BB \rightarrow B0 q_1 101BB \rightarrow B01 q_2 01BB$

$\rightarrow B010 q_1 1BB \rightarrow B0101 q_2 BB \rightarrow B0101 q_{acc} \rightarrow \text{Final state accept}$

Now we see ID for 0100

\rightarrow In tape string is B0100BB

$\rightarrow B q_0 0100BB \rightarrow B0 q_1 1 00BB \rightarrow B01 q_2 00BB$

$\rightarrow B q_0 010 q_1 0BB \rightarrow B0100 q_1 BB \rightarrow \text{Reset}$

Example 5.2.3 : Design Turing machine that accepts $\{a^n b^n \mid n \geq 1\}$

Solution :

- $L = \{ab, aabb, aaabbb, \dots\}$
- 1st scan leftmost a and make it x using state q_0 .



- State q_1 is used for scan b, change b to y and move backward.
- Find rightmost X and repeat above steps in all characters change.
- Check whether there are extra characters or not.

Turing machine is given below :

$$Q = \{q_0, q_1, q_2, q_{chk}, q_{acc}\}$$

$$\Sigma = \{a, b\}$$

$$\Gamma = \{a, b, x, y, B\}$$

δ = Transition function

Transition table

State	a	b	x	y	B
q_0	(q_1, x, R)	-	-	(q_{chk}, 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_{chk}	-	-	-	(q_{chk}, y, R)	q_{acc}
q_{acc}	-	-	-	-	Final

Make it clear by transition diagram

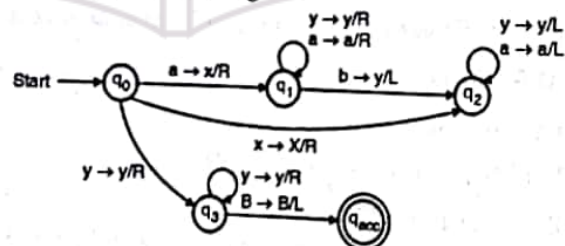


Fig. P. 5.23

Diagram explains that :

1. State q_0 takes initiative to replace leftmost a by x and change the state to q_1 .



2. Function of q_1 is to search rightmost a's and y's until it finds leftmost b. If machine finds b, it changes it to y and enters state q_2 .
3. State q_2 proceeds searching left for x and enters state q_0 upon finding it, move right to change a and the state.
4. All a's and b's are changes to x and y then q_{chk} condition is satisfied. If B or x is encountered before b, then input is rejected either there is existence of a is more than b or input a' b'.
5. From q_0 scanning y state q_3 is entered to scan over y's and check for number of b's remaining. If the y's are followed by a B, state q_{acc} is accepted otherwise rejected.

ID for aa bb \rightarrow BaabbBB

$\rightarrow Bq_0$ aa bb BB

$\rightarrow Bx q_1$ abb BB $\rightarrow xa q_1$, bb BB $\rightarrow x q_2$ ay b BB

$\rightarrow B q_2$ x a yb $\rightarrow Bx q_0$ a y b BB $\rightarrow Bxx q_1$ yb BB

$\rightarrow Bxx y q_1$ b BB $\rightarrow Bxx q_2$ yy BB $\rightarrow Bx q_2$ x yy BB

$\rightarrow Bxx q_0$ yyBB $\rightarrow Bxx y q_3$ yBB $\rightarrow Bxx yy q_3$ BB

$\rightarrow Bxx yy B q_{acc} \rightarrow q_{accept} \rightarrow$ Final state

\rightarrow aabb is accepted.

Example 5.2.4 : Design Turing machine for language, $L = \{a^m b^n \mid m > n > 0\}$

Solution :

$$L = \{aab, aaab, aaaab, \dots, aaabb, aaaabb\}$$

Turing machine,

$$Q = \{q_0, q_1, q_2, q_{chk}, q_{acc}\}$$

$$\Sigma = \{a, (b)\}$$

$$\Gamma = \{a, b, x, y, B\}$$

δ = Transition function

Steps in this construction can be written as,

1. M starts in state q_0



2. If symbol a is read, then state q_0 changes to q_1 and replace symbol by x and move towards right then check for 1st occurrence of b , skip a and y .
3. b is replaced by y and enter the state q_2 and move towards left to search for x then move to right.
4. If symbol is b , then repeat above step, if symbol x is found out then move towards left until current symbol is not equal to a .
5. If current symbol is a that means there is at least one a is extra, then check blank on left side of the string.

Transition table

	a	b	x	y	B
q_0	(q_1, x, R)	-	-	-	-
q_1	(q_1, a, R)	(q_2, y, L)	-	(q_1, y, R)	(q_{chk}, y, L)
q_2	(q_2, a, L)	-	(q_0, x, R)	(q_2, y, L)	-
q_{chk}	(q_{chk}, a, L)	-	(q_{chk}, x, L)	(q_{chk}, y, L)	q_{acc}
q_{acc}	-	-	-	-	Final state

ID for $aaab \rightarrow BaaabBB$

$\rightarrow Bq_0 aaa b BB \rightarrow Bx q_1 aa bB \rightarrow Bx a q_1 a b B$
 $\rightarrow Bx aa q_1 b B \rightarrow Bx aa q_2 y B \rightarrow x a q_2 a y B$
 $\rightarrow Bx q_2 a a y B \rightarrow Bx q_0 aa y B \rightarrow Bxx q_1 a y B$
 $\rightarrow Bxx a q_1 y B \rightarrow Bxx a y q_1 B \rightarrow Bxx a y q_{chk} B$
 $\rightarrow Bxx a q_{chk} k B$
 $\rightarrow Bxx q_{chk} ay \rightarrow Bx q_{chk} x a y B$
 $\rightarrow B q_{chk} xx ay B \rightarrow q_{accept}$



Example 5.2.5 : Design Turing machine to recognize language $L = \{a^{2^n} \mid n \geq 0\}$.

Solution :

$$L = \{aa, aaaa, aaaaaa\}$$

Turing machine for language,

$$Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$$

$$\Sigma = \{a\}$$

$$\Gamma = \{a, x, B\}$$

Steps in this construction can be written as,

1. For constructing Turing machine for this language we have to check step 1 contained single a , then accept it.
2. If step 1 contained more than a single a and the number of a 's was odd then reject it.
3. Return the head to the left hand of the tape.
4. Visit the step 1 again. At each repetition of step 1 number of a 's will be half.
5. If the number of a 's is odd and greater than one, the original number could not have been power of 2 then machine rejects it.
6. If the number of a is one than the original number of ' a ' must have been power of 2 so it will be acceptable by machine.

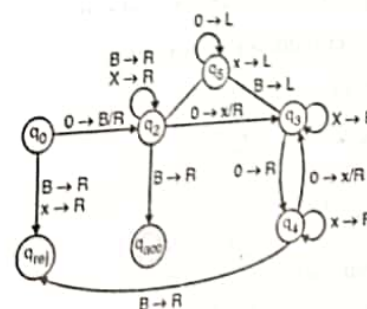


Fig. P. 5.2.5

Find out ID for aaaa

$\rightarrow q_0 \text{ aaaa} \rightarrow Bq_1 \text{ aaa} \rightarrow Bxq_2 \text{ aa} \rightarrow Bxaq_3 \text{ a}$
 $\rightarrow Bxaqx_2B \rightarrow Bxaxq_4x \rightarrow Bq_4xaxB$
 $\rightarrow q_4BxaxB \rightarrow Bq_1xaxB \rightarrow Bxxq_1axB$
 $\rightarrow Bxxq_2xB \rightarrow Bxxxq_2B \rightarrow Bxxq_4xB$
 $\rightarrow Bxq_4xxB \rightarrow Bq_4xxxB \rightarrow q_5BxxxB$
 $\rightarrow Bq_1xxxB \rightarrow Bxq_1xxB \rightarrow Bxxq_1xB$
 $\rightarrow Bxxxq_1B \rightarrow Bxxxq_{acc}$

This Turing machine begins for using a blank over the leftmost. This allows to find the left end of the tape. It also allows machine to identify the case when tape contains one a only.

Example 5.2.6 : Construct Turing machine accepting following language,

$$L = \{a^n b^m c^{m+n} \mid m, n \geq 0\}.$$

Solution :

Language $L = \{\epsilon, ac, bc, aacc, bbcc, abcc\}$

1. First scan leftmost a or b and make it α using state q_0 .
2. Scan rightmost c and make it y.
3. Find out rightmost x
4. Repeat step 1 to 3 till all a's, b's and c's are not over.
5. Check whether there is extra a or b or c.

Turing machine is given below

$$Q = \{q_0, q_1, q_2, q_3, q_{acc}\}$$

$$\Sigma = \{a, b, c\}$$

$$\Gamma = \{a, b, c, x, y, B\}$$

δ = Transition function.

Transition table

State	a	b	c	x	y	B
q_0	(q_1, x, R)	(q_1, y, R)	-	-	(q_4, y, R)	B
q_1	(q_1, a, R)	(q_1, b, R)	(q_1, c, R)	-	(q_2, y, L)	(q_2, B, L)
q_2	-	-	(q_3, y, L)	-	-	-
q_3	(q_3, a, L)	(q_1, b, L)	(q_1, c, L)	(q_0, x, R)	-	q_{acc}
q_4	-	-	-	-	(q_4, y, R)	q_{acc}
q_{acc}	-	-	-	-	-	Final

Instantaneous description for $abcc \rightarrow BabccBB$

$\rightarrow Bq_0abccB \rightarrow Bxq_1bccB \rightarrow Bxbq_1ccB$
 $\rightarrow Bxbccq_1B \rightarrow Bxbccq_2B$
 $\rightarrow Bxbccq_3yB \rightarrow Bxbq_3cyB$
 $\rightarrow Bxxq_3bcyB \rightarrow Bxq_0bcyB$
 $\rightarrow Bxxq_1cyB$
 $\rightarrow Bxxc q_1yB$
 $\rightarrow Bxxc q_2yB$
 $\rightarrow Bxxq_3yyB$
 $\rightarrow Bxxyq_3yB$
 $\rightarrow Bxxyq_3B \rightarrow q_{accept}$

Example 5.2.7 : Design a Turing machine for language,

$$L = \{ww^R \mid w \in (0+1)^R\} \quad L = \{ww^R \mid w \in (0+1)^R \text{ where } w^R \text{ is reverse of } w\}.$$

Solution :

Language for above Turing machine is

$$L = \{00, 11, 0110, 1001, 001100, 100001, \dots\}$$

We can apply simple logic on this language.

1. If initial state q_0 read 0, then last state also read 0.
2. If initial state q_0 read 1, then last state also read 1.

Turing machine is :

$$Q = \{ q_0, q_1, q_2, q_3, q_4, q_5, q_{acc} \}$$

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

$$\Gamma = \{ 0, 1, x, B \}$$

Where δ is

Step	0	1	x	B
q_0	(q_1, x, R)	(q_4, x, R)	(q_{acc}, x, R)	-
q_1	$(q_1, 0, R)$	$(q_1, 1, R)$	(q_2, x, L)	(q_2, B, L)
q_2	(q_3, x, L)	-	-	-
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_0, x, R)	-
q_4	$(q_4, 0, R)$	$(q_4, 1, R)$	(q_5, x, L)	(q_5, B, L)
q_5	-	(q_3, x, L)	-	-
q_{acc}	Final state	-	(q_{acc}, x, R)	(q_{acc}, B, R)

Consider string 0110

ID for B 0 11 0 BB

- $\rightarrow B q_0 0 110 B \rightarrow B x q_1 110 B B$
 $\rightarrow B x 1 q_1 10 B \rightarrow B x 110 q_1 BB$
 $\rightarrow B x 11 q_2 0 BB \rightarrow B x 1 q_3 1x B B$
 $\rightarrow B x q_3 11 x B B \rightarrow B q_3 x 11 x B B$
 $\rightarrow B q_0 11 x BB \rightarrow B x x q_4 1x B B$
 $\rightarrow B x x 1 q_4 x B \rightarrow B x x q_5 1x B$

$$\rightarrow B x q_3 xxx B$$

$$\rightarrow B xx q_0 xx B \rightarrow B xxx q_{acc} x B$$

$$\rightarrow B xx xx q_{acc} B \rightarrow B xxxx q_{acc}$$

$$\rightarrow q_{acc} = \text{accept the string}$$

5.2.2 Description of Turing Machine

- The formal description of Turing machine includes current status of head input symbol, resulting state, the tape symbol replacing the input symbol and movement of Read/Write (R/W) head to the left or right.
- The transition function δ is a partial function and not defined for all (q, x) where partial function δ is defined as,

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times (L, R)$$
- The higher level of description and transition function for a Turing machine is called as a implementation description.
- Where movement of head, symbols stored, states etc can be described in English.
- E.g. move to the left till the end of input string. This instruction requires multiple move.

$$\text{One instruction} = n \times \text{instruction}$$

(Implementation description) (standard Turing machine)

$$n \rightarrow n \text{ can be any number more than one.}$$

- High level description involves instruction in English language and specification of state or transition function is not required,

Syllabus Topic : Turing Machine Construction

5.3 Turing Machine Construction Techniques

- Construction of Turing machine needs complete set of states used for storing information.



- In this section the basic ideas will be presented through various example.

(a) Use of storage in the finite control

- Finite control of Turing machine means its tape, head which is used for reading and writing information on the tape.
- Finite control can be used to hold finite amount of information.
- To do so, the state is written on a pair of elements are exercise of control and others storing the symbol and the new state becomes $Q \times \Gamma$.
- Turing machine is supplied input from left to the end of the tape squares immediately to the right of the blank symbol.
- We define $\delta(q_0, a) \rightarrow (q_1 X, R/L)$. This means that q_1 is changing state of q_0 , a is replaced by X and move the state right or left and continues it in remains cells.

(b) Use of multiple tracks

- We can imagine single tape of Turing machine is divided into k tracks where, $K \geq 1$.
- In multiple track, Turing machine consists K tuples, therefore tape symbols are elements of Γ^K . The moves are in similar manner.

For $K = 3$

B	1	0	1	1	B
B	B	B	1	1	B
B	1	0	1	0	B

(c) Use of checking symbols

Checking of symbols is a useful track for visualizing how a Turing machine recognizes languages defined by repeated strings such as $(ww \mid w \in \Sigma^*)$

$(wwR \mid w \in \Sigma^* \text{ and } w^R \text{ is reverse of } w)$



(d) Use of shifting over

A Turing machine can make space on its tape by shifting all non blank symbols a finite number of cells to the right.

(e) Use of subroutines

- In computer languages, programs are designated by modulating or top-down design. Some task elementary process or has to be done repeatedly, this is called sub-routines.
- A Turing machine is similar are type of sub-routing found in programming languages including recursive procedure and any of the known parameter parsing mechanism.

Syllabus Topic : Variants of Turing Machine

5.4 Variants of Turing Machine

Explain variants of Turing Machine.

- We have seen basic structure for Turing machine with infinite tape in both direction and deterministic standard.
- Now we are going to learn enhancement in Turing machine with its modified structure.
- Acceptance power of variations of Turing machine are same as basic structure of Turing machine. Different types of Turing machine as per following :

1. Multitrack Turing machine
2. Multitape Turing machine
3. Non-deterministic Turing machine
4. Multistack Turing machine
5. Counter machine.

1. Multitrack Turing Machine

- Multitrack Turing machine is a one tape, containing multiple track. In multitrack Turing machine, a single finite control head reads n symbols from n tracks at one step. It can be understood by Fig. 5.4.1.

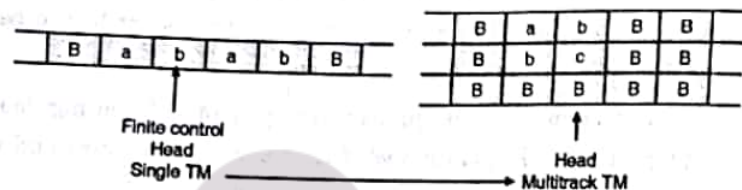


Fig. 5.4.1

- Transition function for multitrack Turing machine :
 $\delta(q_0, b, c, B) = (q_{next}, x, y, z, R \text{ or } L)$
- q_0 is initial state, b, c, B are symbols which are held by head. Head will be changing the states to q_{next} .
- Transition function for multitrack Turing machine is
 $\delta : Q \times \Gamma^n \Rightarrow Q \times \Gamma^n \times X(L, R)^n$.
- Symbol, b is changed to the x , c is changed to y and B is changed to z , move the head left or right. Language acceptance power of single track Turing machine is equivalent to multitrack Turing machine.

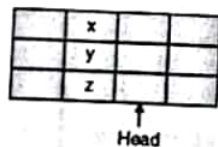


Fig. 5.4.2

2. Multitape Turing Machine

- Multitape Turing machine have n tapes for reading and writing information's with different finite control head.

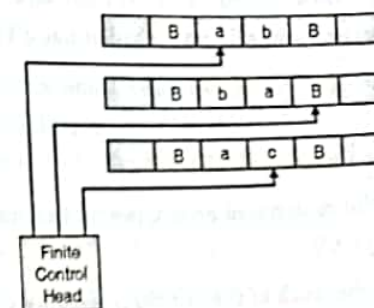


Fig. 5.4.3

- Fig. 5.4.3 depicts that different heads for multitape Turing machine.
- We can see transition function for multitape Turing machine,
 $\delta(q_0, a, B, C) = (q_{next}, x, y, z, L, R, L)$
- q_0 is initial state contains a, B, C symbols, individuals head of each tape change the symbol to x, y, z in q_{next} state as per the movement left or right.

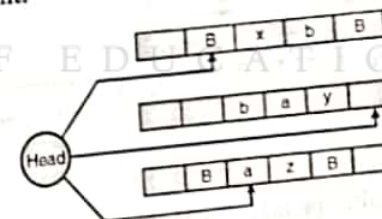


Fig. 5.4.4

- Transition function for Multitape Turing machine,
 $\delta : Q \times \Gamma^n \rightarrow Q \times \Gamma^n \times (L, R, S)^n$
- Where S is no shift or the tape head should stay there. It is transition function based on all n current symbols, write all symbols to each of the n tapes and move each of the n tapes either left or right.

- Tree structure for ID is,

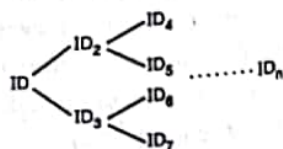


Fig. 5.4.7

- NDTM accepts string w if there exists a sequence of moves starting from the initial ID to an accepting condition.
- Example Fig. 5.4.8 explains the states of NDTM.

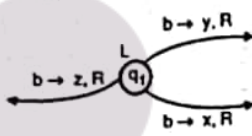


Fig. 5.4.8

- We can construct tree structure for Fig. 5.4.8. Like.

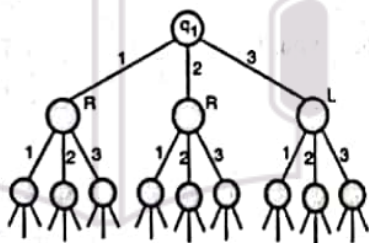


Fig. 5.4.9

- In Fig. 5.4.9 q_1 has 3 choices, in some computation there will be fewer than 3 choices in the computation steps.
- Some points will have zero choice these states or branches of the non determinism halt or reject.
- Despite of NDTM has several action performing over step.
- Acceptance power of NDTM and DTM are same.

- We will show this by simulating the NDTM at each step in the computation with deterministic one.

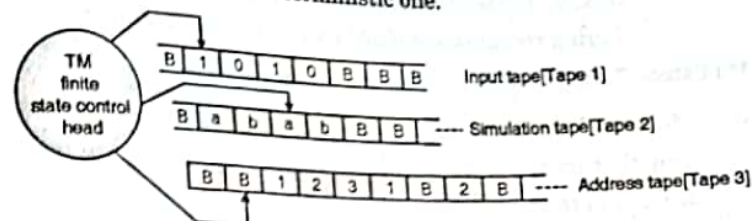


Fig. 5.4.10

1. Input tape store initial input, never modified.
 2. Simulation tape used like the tape of a deterministic Turing machine to perform the simulation.
 3. Address tape keep track of states like breadth first search, it tells which choices to make during simulation.
- A set of transitions exist at each step in the computation of NDTM. We will number these 1 to n where n is the largest number of options any state has.
 1. Initially tape 1 contains the input and tapes 2 and 3 are empty.
 2. Copy tape 1 to tape 2.
 3. Run the simulation using tape 2. When choices occur (i.e. when non deterministic branch points are encountered) consult tape 3. Tape 3 contains the path, the next symbol on each number tells which choice to make. For example, 132 means that we took the first option initially, then the third option and last will be the second option. Run the simulation all the way as far as the address or path goes, if the choice given is invalid, abort this state or branch, and go to the step 4. If get abort for reject state then go to step 4, otherwise get accept.



4. Try the next state or branch and increment the address on tape 3, repeat the step 2. So we can say that, a language is Turing recognizable if and only if NDTM recognizes it.

4. Multistack Turing Machine

- Multistack Turing machine is like semi-infinite tape i.e. read only tape that means you can only read but not write, we can move left or right in tape.
- In PDA, we cannot modify the input and scan the symbol only left to right in one stack.
- But PDA with more than one stack can accept any language which a Turing machine can accept.

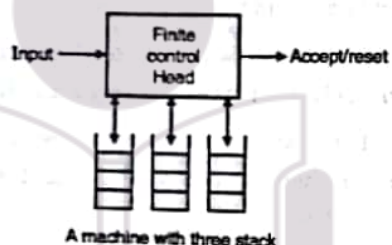


Fig. 5.4.11

In Multistack Turing machine,

1. Finite control head obtain its input from an input source and not from the tape.
2. It has finite stack symbols, which it uses for all its stack.
3. Finite control head reads input symbol which is chosen from the finite input symbols.
4. The top stack symbol on each of its stacks.
5. Finite control head moves to the next state, replace the top symbol with blank or another stack symbols. Usually a different replacement of string for each stack.



5. Counter Machines

1. Counter machine is like multistack machine but in place of each stack is a counter.
2. Counter register holds non-negative integers.
3. Move of the counter machine depends on its state, input symbol and counters which are zero.
 - (a) Change the new state.
 - (b) Increment or decrement by one from any of its counters.

A counter has always positive value, so counter value cannot be less than 0.

Syllabus Topic : The Linear Bound Automata Model

5.5 Linear Bound Automaton

Write a short note on Linear Bound Automaton and Linear Bound Automaton Model.

- LBA is a restricted form of Turing machine. This machine accepts context sensitive languages.

A Turing machine that uses only the tape space occupied by the input is called a Linear Bound Automaton (LBA).

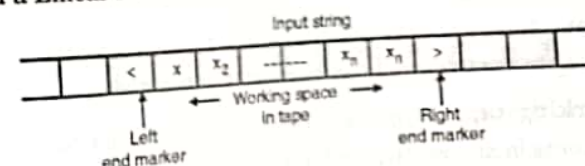


Fig. 5.5.1

- A Linear Bounded Automata is a Non-deterministic Turing machine which has a restricted length of tape but bounded by a linear function of the length of input string.

$$M = \{Q, \Sigma, \gamma, \delta, q_0, B, <, >, R\}$$



Such that all symbols have the same meaning as in the basic model, $<$ and $>$ are special symbols, $<, > \in \Sigma$.

Both symbols are end markers $<$ is left end marker and $>$ is right end marker. The Turing machine cannot replace $<$ or $>$ with anything else, nor move the tape head left of $<$ or right of $>$.

LBA model

- We represent Fig. 5.5.2 in block diagram as a LBA model.

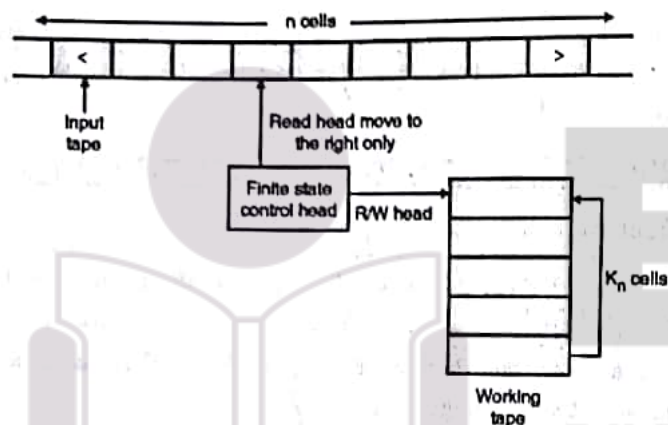


Fig. 5.5.2 : LBA model

- There are two tapes
 1. **Input tape** : On the input tape, the head never prints and never moves to the left.
 2. **Working tape** : On the working tape the head can modify the contents in any way without any restriction.

The mapping of transition function δ is represented by (q, w, k)

where $q \in Q$, $w \in \Gamma$, k is some integer between 1 and n .

The transition of ID's is similar except,

- ✓ 1. If R/W head move to left, k changes to $k - 1$.
- ✓ 2. If R/W head move to right, k changes to $k + 1$.



The language accepted by LBA is defined as the set,
 $\{w \in (\Sigma - \{<, >\})^* \mid (q_0, <w>, 1) \vdash^* (q, \alpha, i)\}$

for some $q \in F$ and for some integer i between one to n .

- * Null string may accepted by LBA.

Syllabus Topic : Linear Bounded Automata and Languages

5.5.1 Linear Bounded Automata and Languages

What are the languages accepted by Linear Bound Automaton ?

- The set of strings accepted by non deterministic LBA is the set of strings generated by the context sensitive grammar (CSG), excluding the null strings.
- If L is context sensitive language, then L is accepted by a linear bounded automata the converse is also true. We can construct type 0 grammar accepted by Turing machine for language which will take input string M , tries to apply productions backwards to get finally start state q_0 .
- The productions are constructed in two steps.
 1. Transform the string $(q_0 <w>)$ into the string $(q_1 b)$ where q_0 is initial state q_1 is an accepting state, $<$ and $>$ are left and right end marker respectively. This is called the **transformational grammar**.
 2. Inverse of production rules by reversing the production of transformational grammar. The construction is in such a that w is accepted by M iff $w \in L(G)$.
- A LBA machine accepts a string w after starting at the state with R/W head reading $<$ (left end mark), and halts over $>$ (right end mark) in a final state, otherwise w is rejected.
- The production rules for the generative grammar are constructed as in the case of Turing machine.



- The following additional productions are needed in case of LBA

$$a_i q_f \Rightarrow q_f \text{ for all } a_i \in \Gamma$$

$$\langle q_f \rangle \Rightarrow \langle q_f, \langle q_f \rangle \rightarrow q_f$$

Examples of languages accepted by LBA

1. $L = \{a^n b^n c^n \mid n \geq 1\}$
2. $L = \{a^n \mid n \geq 0\}$
3. $L = \{ww \mid w \in (a, b)^*\}$
4. $L = \{w^n \mid w \in (a, b)^*, n \geq 1\}$
5. $L = \{www^R \mid w \in (a, b)^*\}$

Syllabus Topic : Pushdown Automata

5.6 Pushdown Automata

Write components of Pushdown Automata.

- A pushdown automata is a way to implement a context free grammar for a regular grammar.
- It can remember on infinite information.
- It is combination of finite state machine and a stack.
- A stack has two operations namely **PUSH operation** and **POP operation**.

Push operation : It is used to insert an element into stack.

Pop operation : It is used to delete an element from stack.

→ **Top pointer** : Which point to the current input of stack.

PDA has 3 components

1. An input tape
- A control unit
- A stack with infinite size

Appⁿ - can be used as language acceptor

→ compiler design (parser design for syntactic analysis)

→ Online Transaction processing system

→ Tower of Hanoi (Recursive soln)

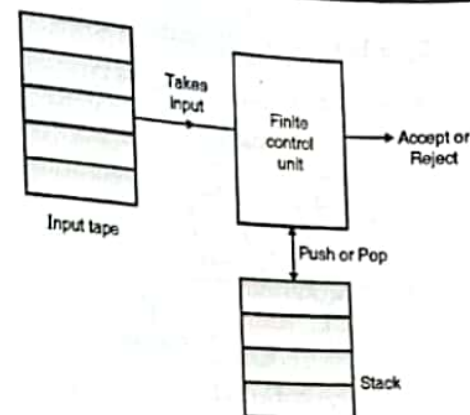


Fig. 5.6.1

Operation

User can give the input to the input tape and it will take to finite control unit, stack symbol only push or pop the input values and can be control led by finite control unit, then it can be accept or reject.

5.6.1 Definition of Push Down Automata(PDA)

Define a Push Down Automata.

PDA has 7 tuples

$$(Q, \Sigma, \Gamma, \delta, q_0, Z_0, f)$$

τ symbol is pronouns "Tau"

Where,

Q = Finite number of states

Σ = Input alphabet

Γ = Stack symbol

δ = Transition function

$$Q \times (\Sigma \cup \{ \epsilon \}) \times (\Gamma) \rightarrow Q \times \Gamma^*$$

q_0 = Initial state ($q_0 \in Q$)



Z_0 = Initial stack top symbol ($\Gamma \in S$)

F = Set of accepting states ($F \subseteq Q$)

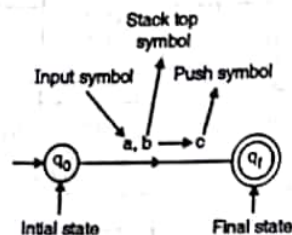


Fig. 5.6.2

For example,

Construct a PDA that accepts $L = \{0^n, 1^n \mid n \geq 0\}$.

Step I:

Problem Description

- If you encounter input '0' and top is null, push '0' into stack. This may iterate.
- If you encounter input '1' and top is '0', pop '0' from stack. This may iterate.
- The process is repeated until blank space and Z_0 came.
- This indicates that string is valid, otherwise string is invalid.

Step II:

Construct transition diagram for PDA

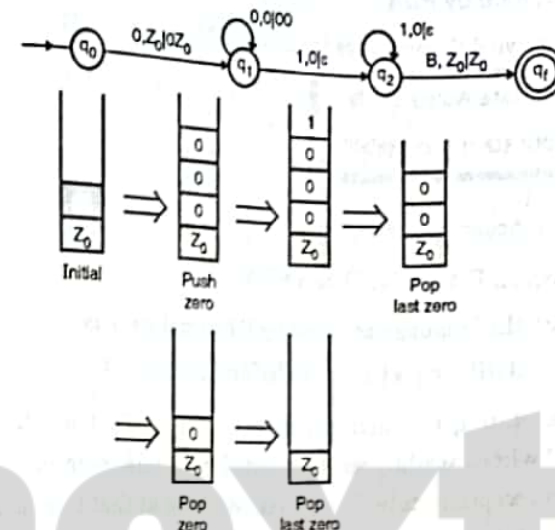


Fig. 5.6.3

Step III:

Transition table for PDA

Transition	Operation Performed
$q_0 : \delta(q_0, 0, Z_0) \rightarrow (q_1, 0Z_0)$	push 0
$q_1 : \delta(q_1, 0, 0) \rightarrow (q_1, 00)$	push 0
$\delta(q_1, 0, 1) \rightarrow (q_2, \epsilon)$	pop 0
$q_2 : \delta(q_2, 1, 0) \rightarrow (q_2, \epsilon)$	pop 0
$\delta(q_2, B, Z_0) \rightarrow (q_f, Z_0)$	Accept
$q_f : \text{Accept}$	

**5.6.2 Acceptance by PDA**

There are two different ways to define PDA acceptability.

1. Final state Acceptability
2. Empty state Acceptability

1. Final state Acceptability

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA

then $L(P)$, the language accepted by P by final state is

$$L(P) = \{w \mid (q_0, w, Z_0) \vdash^* (q, \epsilon, \alpha)\}$$

For some state $q \in F$ and any stack string α . That is, starting in the initial ID with w waiting on the input, P consumes w from the input and enters an accepting state. The content of stack at that time is irrelevant.

2. Empty stack Acceptability

Let $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. Then $N(P)$; the language accepted by P by empty stack is,

$$N(P) = \{w \mid (q_0, w, Z_0) \vdash^* (q, \epsilon, \epsilon)\}$$

for any state q . That is $N(P)$ is the set of input w , that P can consume and at the same time empty in stack. ($N(P)$ stands for null stack or empty stack).

Syllabus Topic : PDA and CFG**5.6.3 Push Down Automata (PDA) and Context Free Grammar (CFG)**

Push Down Automata (PDA) to Context Free Grammar (CFG)

Theorem : If L is a CFL, then we can construct a PDA accepting language L by empty store i.e. $L = N(A)$

Method

Let $L = L(G)$, where $G = (V_N, \Sigma, P, s)$ in a context free grammar we can construct a PDA as,

$$A = (\{q\}, \Sigma, V_N \cup \Sigma, \delta, q, S, \Phi)$$



Where δ is defined by following rules.

$$R_1 = \delta(q, \epsilon, A) = \{(q, \alpha) \mid A \rightarrow \alpha \text{ is in } P\}$$

$$R_2 = \delta(q, a, a) = \{(q, \epsilon) \mid \text{for every } a \text{ in } \Sigma\}$$

Rule 1

[Pushdown symbols in A are variable or terminals. If PDA reads a variable A on the top of PDS (Push Down Stack), it makes a ϵ move by placing RHS of any A production]

Rule 2

[If PDA reads a terminal a on PDS (Push Down Stack) and if it matches with current input symbol then PDA erase a .]

Example 5.6.1 : Construct a PDAA which is equivalent to the following given CFG.

$$S \rightarrow 0CC, C \rightarrow 0S, C \rightarrow 1S, C \rightarrow 0$$

Test whether 010^4 is accepted by $N(A)$

Solution :**Step 1**

The target PDA A is as following :

$$A = (\{q\}, \{0, 1\}, \{S, C, 0, 1\}, \delta, q, S, \Phi)$$

δ is defined by following rules :

$$R_1 : \delta(q, \epsilon, S) = \{(q, 0CC)\}$$

$$\delta(q, \epsilon, C) = \{(q, 0S), (q, 1S), (q, 0)\}$$

$$R_2 : \delta(q, 0, 0) = \{(q, \epsilon)\}$$

$$\delta(q, 1, 1) = \{(q, \epsilon)\}$$

Step 2

Check if 010^4 can get from this production,

$$S \Rightarrow 0CC$$

$$\Rightarrow 01SC$$

$$\Rightarrow 010CCC$$

$$\Rightarrow 010^4$$

Step 3

Construct PDA


 $(q, 010^4, S) \vdash (q, 010^4, 0CC) \vdash (q, 10^4, CC)$ $\vdash (q, 010^4, 1SC) \vdash (q, 0^4, SC)$ $\vdash (q, 0^4, 0CCC) \vdash (q, 0^3, CCC)$ $\vdash (q, 0^3, 0CC) \vdash (q, 0^2, CC)$ $\vdash (q, 0^2, 0C) \vdash (q, 0, C)$ $\vdash (q, 0, 0) \vdash (q, \epsilon, \epsilon)$ Thus $010^4 \in N(A)$

□□□

CHAPTER**6****Undecidability****University Prescribed Syllabus**

The Church-Turing thesis, Universal Turing Machine, Halting Problem, Introduction to Unsolvability Problems.

Syllabus Topic : The Church-Turing Thesis**6.1 Church Turing Thesis**

 Write Short note on Church Turing Thesis.

- Many mathematicians were finding if there is a procedure, algorithm for solving a problem that can be implemented on Turing machine. Alonzo Church also worked on same problem and he solved some problems. Chomsky unrestricted grammar recursive function theory by λ calculus.
- If a computational problem P is solvable by human being then it is solved by computer and if it is solvable by computer then it is solved by Turing machine.
- Mr. Church formalised simple hypothesis over it. If a computational problem P is solvable by human being then it is directly solved by Turing machine.