UNIT: 5

Alen Turing proposed a model of an abstract machine called the Turing Machine which could perform any computatio process carried out by the computer.

Turing machine is the machine format of unrestricted 'ng ie. all types of lang are accepted by the TM.

Basics of Turing Machine

The TM is defined by 7 attributes.

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

where

Q : finite set of states

Σ : finite set of i/p alphabets

Γ : finite set of allowable tape symbol

δ : Transitional function

q_0 : Initial state

B : A symbol of Γ called blank

F : final state

and δ is a Mapping from $Q * \Gamma \rightarrow (Q \times \Gamma \times \{L, R, H\})$

This denotes that one state by getting one i/p from the i/p tape, the machine moves to a state. It writes symbol on the tape & moves to left, right or halts.

Transitional Representation of TM

The mathematical notation for a TM is $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$. In a TM the δ consists of the form

$$Q \times \Gamma \rightarrow (Q \times \Gamma \times \{L, R, H\}),$$

where the first is a present state and the second is the present i/p (single character $\in \Gamma$).

In a graphical notation of the TM, there are states. A circle with an arrow indicates a beginning state and a state with double circle indicates a final state, where the machine halts finally.

The label of the state transitions consists of the i/p symbol, the symbol written on the tape after traversal and the direction of movement of the read-write head (left, right or halt)

Example:-

Design a TM by the transitional notation for the lang $L = a^n b^n$ where $n > 0$

Sol :-

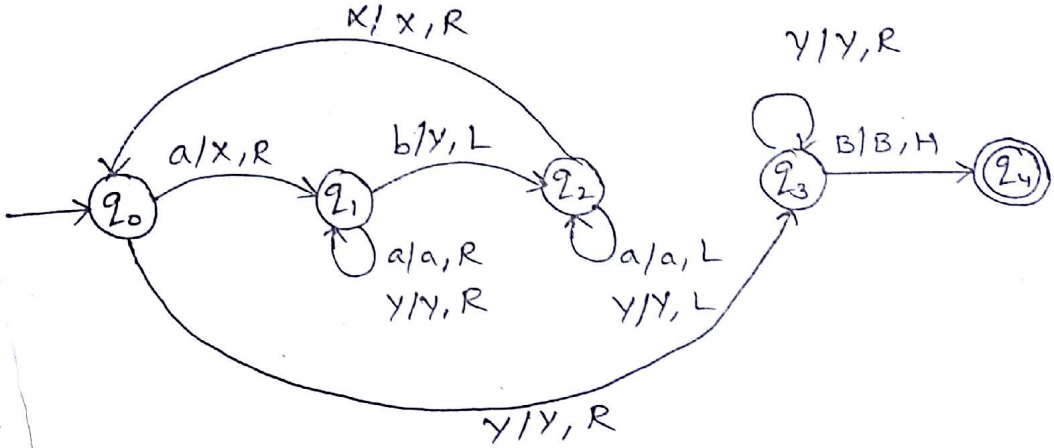


fig:- Transitional representation of Turing Machine.

Instantenous Description (ID) in respect of TM :-

The ID of the TM remembers the following at a given instance of time.

→ The contents of all the cells of the tape, starting from the rightmost cell up to atleast the last cell, containing a non block symbol and containing all cells up to the cell being

→ The cell currently being scanned by the read-write head and.

→ The state of the machine.

Example:-

Show the movement of the read-write head & the content of the i/p tape for the string 'abba' traversed by the following TM.

$\delta(q_0, a) \rightarrow (q_0, X, R)$

$\delta(q_0, b) \rightarrow (q_0, b, R)$

$\delta(q_0, B) \rightarrow (q_1, B, L)$

$\delta(q_1, b) \rightarrow (q_1, Y, L)$

$\delta(q_1, X) \rightarrow (q_1, X, L)$

$\delta(q_1, B) \rightarrow (q_1, B, H)$.

Solution.

left-side diagram shows before traversing		Right side diagram show After traversing
B a b b a B ↑	$\delta(q_0, a) \rightarrow (q_0, X, R)$	B X b b a B ↑
B X b b a B ↑	$\delta(q_0, b) \rightarrow (q_0, b, R)$	B X b b a B ↑
B X b b a B ↑	$\delta(q_0, b) \rightarrow (q_0, b, R)$	B X b b a B ↑
B X b b a B ↑	$\delta(q_0, a) \rightarrow (q_0, X, R)$	B X b b X B ↑

B x b b x B (↑ under B)	$\delta(q_0, B) \rightarrow (q_1, B, L)$	B x b b x B (↑ under x)
B x b b x B (↑ under x)	$\delta(q_1, x) \rightarrow (q_1, x, L)$	B x b b x B (↑ under b)
B x b b x B (↑ under b)	$\delta(q_1, b) \rightarrow (q_1, Y, L)$	B x b Y x B (↑ under b)
B x b Y x B (↑ under b)	$\delta(q_1, b) \rightarrow (q_1, Y, L)$	B x Y Y x B (↑ under x)
B x Y Y x B (↑ under x)	$\delta(q_1, x) \rightarrow (q_1, x, L)$	B x Y Y x B (↑ under B)
B x Y Y x B (↑ under B)	$\delta(q_1, B) \rightarrow (q_1, B, H)$	B x Y Y x B

2) Design a TM to accept the lang $L = \{a^n b^n, n \geq 1\}$.

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

$\delta(q_1, a) \rightarrow (q_1, a, R)$.

$\delta(q_1, b) \rightarrow (q_2, Y, L)$.

$\delta(q_2, a) \rightarrow (q_2, a, L)$

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

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

$\delta(q_2, Y) \rightarrow (q_2, Y, L)$.

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

$\delta(q_3, Y) \rightarrow (q_3, Y, R)$ $\quad \delta(q_3, B) \rightarrow (q_4, B, H)$.

Nondeterministic Turing Machine (NTM).

A non-deterministic TM is defined as $(Q, \Sigma, \Gamma, \delta, q_0, B, F)$ where δ is $Q \times \Gamma \to 2^{Q \times \Gamma \times (L,R)}$.

Let there exist a transitional function

$$\delta(q_0, 0) \to (q_0, 0, R), (q, 1, R)$$

For traversing the i/p symbol, we have to consider two transitions & construct the following accordingly.

→ NTM is more powerful than DTM.

→ An NTM T_1 can be simulated to an equivalent DTM T_2. One of the methods is to construct a computation tree of the NTM and perform a breadth first search of the computational tree from the root until the halt state is reached.

→ Any lang accepted by an NTM is also accepted by a DTM.

Ex:- Construct a TM over $\{a, b\}$ which contains a substring abb.

Solution:-

$$(a,b)^* abb\ (a,b)^*.$$

$\delta(q_1, a) \to (q_1, a, R), (q_2, a, R)$

$\delta(q_1, b) \to (q_1, b, R)$.

$\delta(q_2, b) \to (q_3, b, R)$

$\delta(q_3, b) \to (q_4, b, R)$

$\delta(q_4, a) \to (q_4, a, R)$

$\delta(q_4, b) \to (q_4, b, R)$

$\delta(q_4, B) \to (q_f, B, H)$.

Conversion of Regular Expression to Turing Machine.

Processing Steps:-

(i) Convert the regular expression to an equivalent automata without ε moves.

(ii) Change both the initial & final states of the automata to an intermediate state.

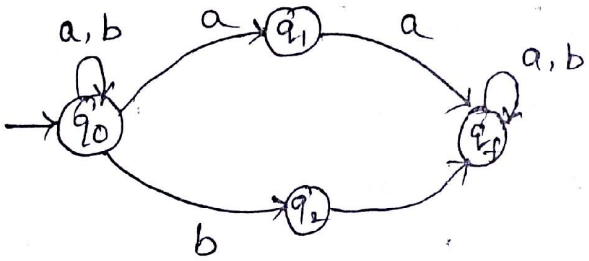
(iii) Insert a new initial state with a transition (B, B, R) to the automata's initial state.

(iv) Convert the ~~transitions~~ transitions with label 'ip' to (ip, ip, R).

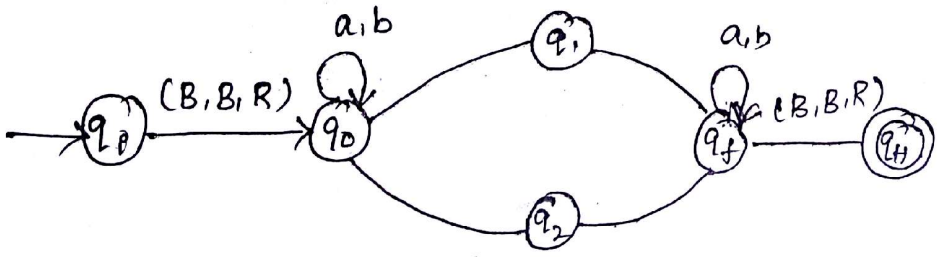
(v) Insert a new final state with a transition (B, B, R) from the automata's final state to the new final state.

Example:- Construct a TM for the regular expression $(a+b)^*(aa+bb)(a+b)^*$.

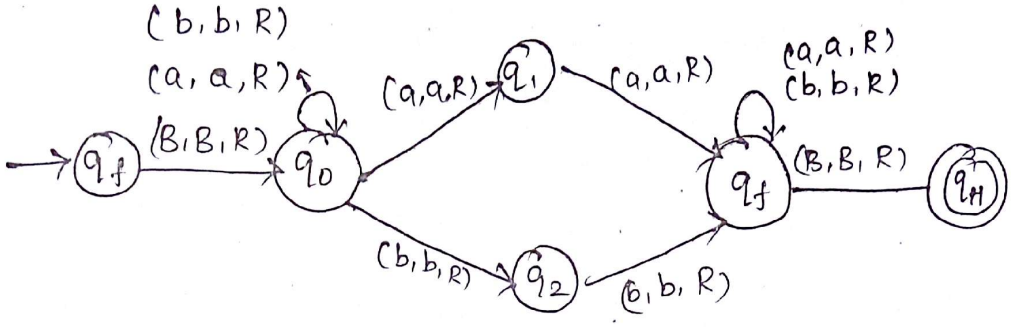
Solution:-



Inserting a new initial state q_i & final state q_H, the finite automata becomes.



converting the levels of the i/p, the TM becomes.



Two stack PDA and Turing Machine.

Minsky Theorem:

Any lang accepted by a two-stack PDA can also be accepted by some TM and vice versa.

A general minsky model is shown in the following figure.

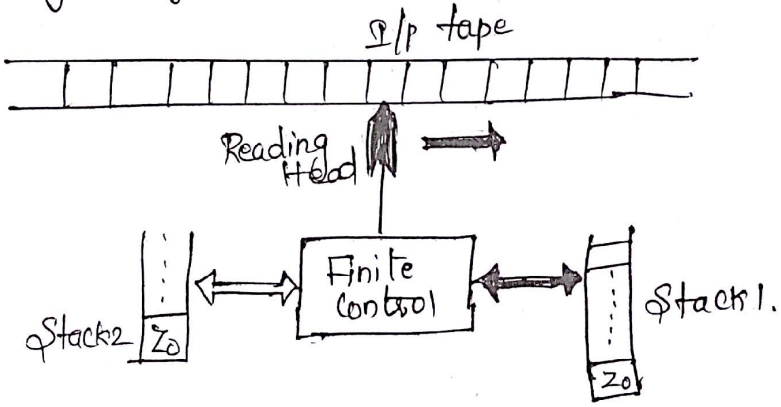


Fig:- Two stack PDA.

The model in figure is a two-stack PDA containing an i/p tape, a reading head & finite control as it was in PDA. The additionals are two stacks S1 and S_2.

Example:- Design a two-stack PDA for $\{a^n b^n c^n d^n\}$.

Solution:- Traversing a, X are put into stack S_1, 'b', to the stack top X in S_1, X is popped from S_1 and Y is pushed into S_2.

Traversing c with the stack top Y in S_2 Y is popped from S_2 and Z is pushed into S_1. Traversing 'd', X' is popped from S_1. The transitional functions are.

$\delta(q_0, \lambda, Z_1, Z_2) \rightarrow (q_1, Z_1, Z_2)$.

$\delta(q_1, a, Z_1, Z_2) \rightarrow (q_1, XZ_1, Z_2)$

$\delta(q_1, a, X, Z_2) \rightarrow (q_1, XX, Z_2)$

$\delta(q_1, b, X, Z_2) \rightarrow (q_2, \lambda, YZ_2)$

$\delta(q_2, b, X, Y) \rightarrow (q_2, \lambda, YY)$.

$\delta(q_2, c, Z_1, Y) \rightarrow (q_3, ZZ_1, \epsilon)$.

$\delta(q_3, c, Z, Y) \rightarrow (q_3, ZZ, \epsilon)$

$\delta(q_3, d, Z, Z_2) \rightarrow (q_4, \epsilon, Z_2)$

$\delta(q_4, d, Z, Z_2) \rightarrow (q_4, \epsilon, Z_2)$.

$\delta(q_4, \epsilon, Z_1, Z_2) \rightarrow (q_f, Z_1, Z_2)$ //accepted by final state.

Variations of the Turing Machine:

The different types of Turing Machines are

(1) Multi tape T.M

(2) Multi head TM

(3) Two-way infinite tape

(4) K-dimensional TM

(5) Non-deterministic TM

(6) Enumerator.

(i) Multi-tape Turing Machine.

In multiple tape each of the tapes is connected to the finite control by a read-write head. It means m tapes have m read-write heads. Depending on the Present States I/P symbols scanned by ~~the~~ each of the head, the TM can

→ Change its state
→ write a new symbol on the respective cell of the respective tape from where the i/ps were scanned.
→ move the head to one left or one right.

A multi-tape Turing Machine having K tapes ($K \geq 1$) is symbolically represented as

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

where Q: Finite set of states
Σ: finite set of I/P symbols
Γ: finite set of allowable tape symbols
q_0: Initial state
B: A symbol of Γ called blank.
F: final states
δ: Transitional function.

where δ is

$$Q \times (\Gamma, \Gamma_2 \dots \Gamma_K) \rightarrow (Q \times (\Gamma, \Gamma_2 \dots \Gamma_K) \times [(L, R, H), (L, R, H) \dots \text{K times}]$$

In general

$$Q \times \Gamma^K \rightarrow (Q \times \Gamma^K \times (L, R, H)^K).$$

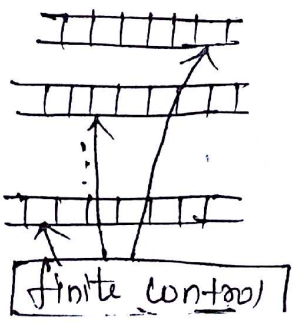


fig:- Multi tape TM.

Ex:- Design a multitape Turing Machine for checking a binary string is palindrome or not. Show an ID for 10. The transition diagram

Sol :-

Consider the TM have two tapes. The I/P is written on the 1st tape. The machine works by the following process.

1) Copy the i/p from the 1st tape to the second tape by traversing the 1st tape from L → R.

2) Traverse the 1st tape again from R → L & point the head to the 1st symbol of the I/P on tape 1.

3) Moves the two heads pointing on the two tapes in opposite directions checking that the two symbols are identical & erasing the copy in tape 2 at the same time

Here, the moves on two tapes are described in a group.

The string is written on tape 1. Tape 2 is blank. The string may start with '0' or '1'. The machine is in state q_0 initially.

Tape 1: $\delta(q_0, 0) \rightarrow (q_0, 0, R)$ Tape 2: $\delta(q_0, B) \rightarrow (q_0, 0, R)$

Tape 1: $\delta(q_0, 1) \rightarrow (q_0, 1, R)$ Tape 2: $\delta(q_0, B) \rightarrow (q_0, 1, R)$

When the head of the 1st tape finds 'B' at the right side, the string is totally traversed. The state changed to q_1 & the head of the first tape starts to traverse to the left.

tape 1: $\delta(q_0, B) \rightarrow (q_1, B, L)$

tape 2: $\delta(q_0, B) \rightarrow (q_1, B, -)$ (- means no traversal)

Traversing the first tape from R → L, the head has to traverse '0' and '1' & the head on the 2nd tape has no traversal.

The transition functions are

Tape 1: $\delta(q_1, 0) \rightarrow (q_1, 0, L)$ // for '0'.

Tape 2: $\delta(q_1, B) \rightarrow (q_1, B, -)$

Tape 1: $\delta(q_1, 1) \rightarrow (q_1, 1, L)$ // for '1'.

$\delta(q_1, B) \rightarrow (q_1, B, -)$

When the head of the first tape finds 'B' at the left side, the string is totally traversed from R → L. The state is changed to q_2 & the head of tape tape starts to traverse to right

Tape 1: $\delta(q_1, B) \rightarrow (q_2, B, R)$

Tape 2: $\delta(q_1, B) \rightarrow (q_2, B, L)$

If both the heads traverse '0' or both the head traverse '1' then it is allowed for them to traverse R → L, respectively.

If one head traverses '0' & other traverses '1' or vice versa, it stops on a state which signifies that the string is not a palindrome.

Tape 1: $\delta(q_2, 0) \rightarrow (q_2, 0, R)$ // for '0'

Tape 2: $\delta(q_2, 0) \rightarrow (q_2, B, L)$

Tape 1: $\delta(q_2, 1) \rightarrow (q_2, 1, R)$ // for '1'

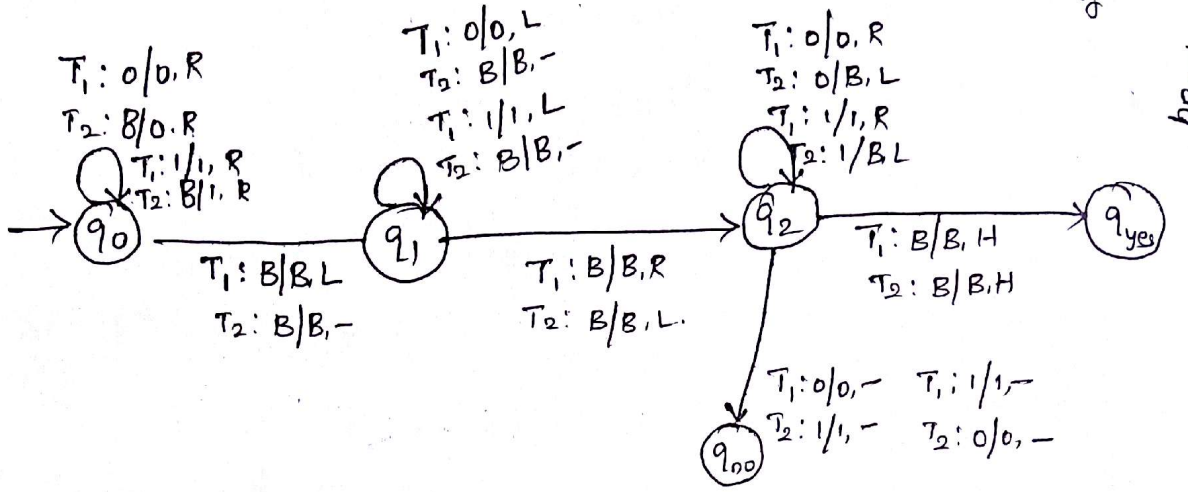
Tape 2: $\delta(q_2, 1) \rightarrow (q_2, B, L)$

Tape 1: $\delta(q_2, 0) \rightarrow (q_{no}, 0, -)$
Tape 2: $\delta(q_2, 1) \rightarrow (q_{no}, 1, -)$
Tape 1: $\delta(q_2, 1) \rightarrow (q_{no}, 1, -)$
Tape 2 $\delta(q_2, 0) \rightarrow (q_{no}, 0, -)$

} // If H is not palindrome

Tape 1: $\delta(q_2, B) \rightarrow (q_{yes}, B, H)$
Tape 2: $\delta(q_2, B) \rightarrow (q_{yes}, B, R)$

} q_{yes} is the state which signifies H



ID for 1001

$$q_0\begin{bmatrix} B\underline{1}001B \\ B\underline{B}BBBB \end{bmatrix} \to q_0\begin{bmatrix} B1\underline{0}01B \\ B1\underline{B}01B \end{bmatrix} \to q_0\begin{bmatrix} B10\underline{0}1B \\ B10\underline{B}BB \end{bmatrix} \to q_0\begin{bmatrix} B100\underline{1}B \\ B100\underline{B}B \end{bmatrix}$$

$$q_0\begin{bmatrix} B1001\underline{B} \\ B1001\underline{B} \end{bmatrix}$$

$$\to q_1\begin{bmatrix} B100\underline{1}B \\ B1001B \end{bmatrix} \to q_1\begin{bmatrix} B10\underline{0}1B \\ B1001\underline{B} \end{bmatrix} \to q_1\begin{bmatrix} B1\underline{0}01B \\ B1001\underline{B} \end{bmatrix} \to q_1\begin{bmatrix} B\underline{1}001B \\ B1001\underline{B} \end{bmatrix}$$

$$\to q_1\begin{bmatrix} \underline{B}1001B \\ B1001\underline{B} \end{bmatrix} \to q_2\begin{bmatrix} B\underline{1}001B \\ B100\underline{1}B \end{bmatrix} \to q_2\begin{bmatrix} B1\underline{0}01B \\ B100\underline{B}B \end{bmatrix}$$

$$\to q_2\begin{bmatrix} B100\underline{1}B \\ B10\underline{0}BB \end{bmatrix} \to q_2\begin{bmatrix} B1\underline{0}01B \\ B\underline{1}BBBB \end{bmatrix} \to q_2\begin{bmatrix} B1001\underline{B} \\ \underline{B}BBBBB \end{bmatrix}$$

$$\to q_{yes}\begin{bmatrix} B1001B \\ B\underline{B}BBBB \end{bmatrix}$$ Halt & accepts.

a) Multihead Turing Machine:-

In this type of turing machine has multiple heads. All the heads are connected to the finite control. The transitional function for the multi head Turing machine having n heads can be described as

$$\delta(Q, \Sigma_1, \Sigma_2, \ldots \Sigma_n) \rightarrow Q \times (\Gamma_1, L, R, N, H) \times (\Gamma_2, L, R, N, H) \times \ldots (\Gamma_n, L, R, N, H),$$

where Σ_i → i/p Symbol under head i

Γ_i → Symbol written by replacing Σ_i.

N → no movement of the head.

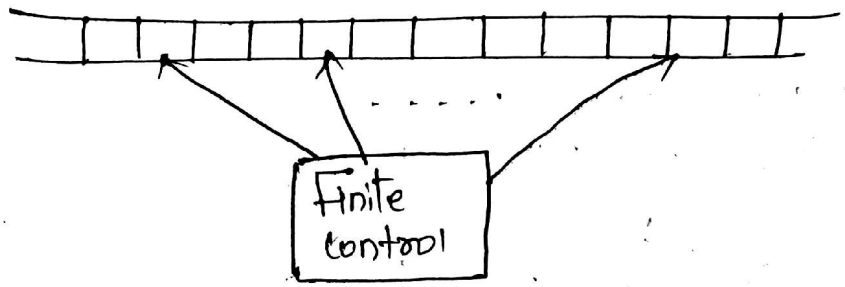


Fig:- Multi head TM.

Two Special Cases in designing a multi head TM.

→ A Situation may arise when more than one heads are Scanning a particular cell at the same time. But the Symbol written by one head is diff from the Symbol written by other head

→ Another Situation may arise when a cell under a head is the leftmost cell & the transitional function instructs the head to go left. This condition is called hanging.

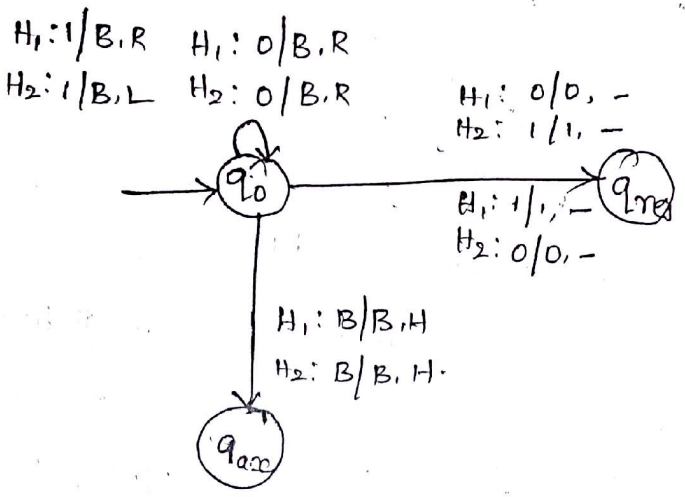
"Every multihead Turing Machine has an equivalent Single-head Turing Machine".

Example:- Design a multihead TM for checking whether a binary string is a palindrome or not. Show the ID for 1001 & 101.

Sol Consider a TM with two heads. The heads are Pointing to the two ends of the strings on the tape. Both head traverse the string in the opposite direction.

→ If both the heads get the same symbols, then it traverses the next s/p right or left by replacing the present symbol by B.

→ If both the heads get B, then halt & declare the string as a palindrome.



ID for 1001

$(q_0, B1001B) \rightarrow (q_0, BB00BB) \rightarrow (q_0, BBBBBB)$

$\downarrow$

$(q_{acc}, BBBBBB)$

Halt.

ID for 101:

$(q_0, B101B) \rightarrow (q_0, BB0BB) \rightarrow (q_0, BBBBB)$

$\downarrow$

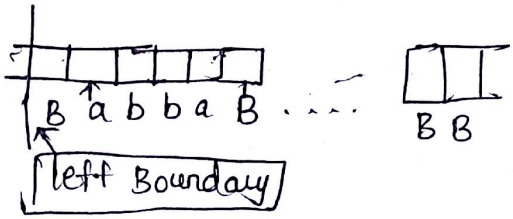
$(q_{acc}, BBBBB)$

Halt.

3) Two-way Infinite Tape:-

In a Turing machine there is a left boundary. If the head crosses the boundary and wants to go left, then the situation is a called crash condition. But the head may traverse the right side up to infinity.

The i/p tape of the general TM can be treated as a one-way infinite tape.



A Turing machine where there are infinite number of sequence of blank on both sides of the tape is called a two-way infinite tape TM.

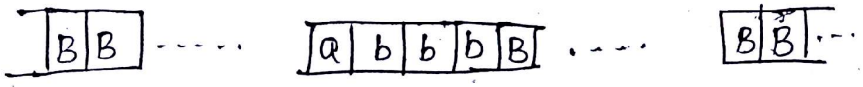


Fig:- Typical diagram of a Two-way Infinite tape.

4) K-dimensional Turing Machine:

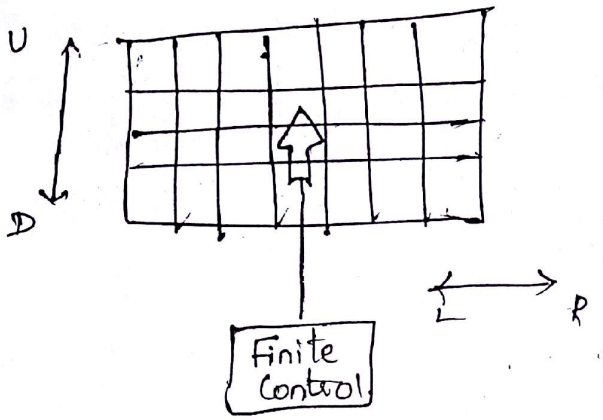
The i/p tape of two-dimensional Turing Machine is extended to infinity in both sides, but in one direction.

If the i/p tape can extend infinitely in more than one dimension, then the Turing Machine is called a multi dimensional Turing Machine.

The transitional function for a K-dimensional Turing Machine is

$$Q \times \Sigma \rightarrow Q \times \Gamma \times (L, R, U, D, H).$$

where L= left , R= Right U= up D= Down



5) Non-deterministic Turing Machine :-

The transitional function of the non-deterministic finite automata have a single state & a single i/p and there may be more than one move.

$$Q \times \Sigma \rightarrow \text{Power Set of } Q \times \Gamma \times (L, R, H).$$

Ex:- Design a Turing Machine for $0^n 1^m$ where $n, m > 0$ and n may not be equal to m.

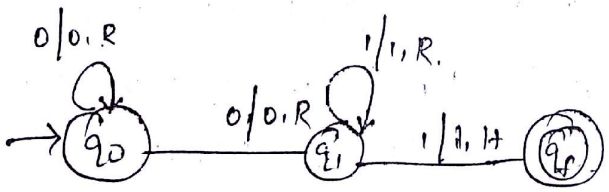
Solution

$$\delta(q_0, 0) \rightarrow (q_0, 0, R)$$

$$\delta(q_0, 0) \rightarrow (q_1, 0, R)$$

$$\delta(q_1, 1) \rightarrow (q_1, 1, R)$$

$$\delta(q_1, 1) \rightarrow (q_f, 1, H).$$



—x—

Every non-deterministic Turing machine has an equivalent deterministic Turing machine.

Enumerator

Enumerator is a type of Turing Machine which is attached to a Printer.

It has a work tape & an o/p tape. The work tape is a write only once tape. At each step, the machine chooses a symbol to write from the o/p alphabet on the o/p tape.

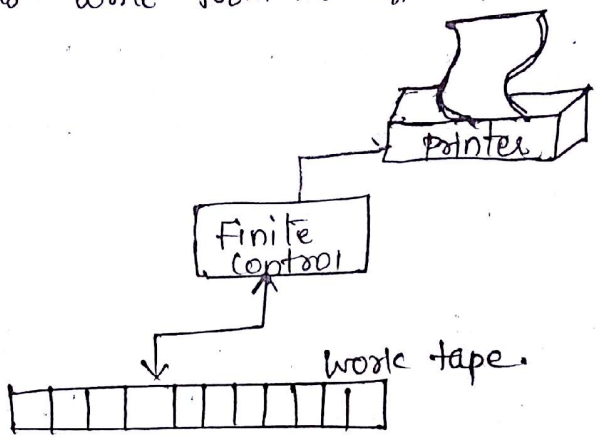


fig:- Enumerator.

An enumerator is defined by 7 tuples.

$(Q, \Sigma, \Gamma, \delta, q_0, B, q_p)$.

q_p - Final state (Print state).

Turing Machine as an Integer function.

An integer function is a function which takes its arguments as integers & returns the result as integer.

If we consider integers addition, subtraction, Multiplication etc., all these functions take the arguments in integer & produce the result in integer.

Let an integer function be

$$f(i_1, i_2, \ldots i_k) = M.$$

We can solve this function using TM.

→ First of all, the i/p tape must represent the i/p arguments.

→ Let the number of '0' be used to represent the arguments. But there must be a seperator to differentiate the number of '0' for each arguments.

→ In the TM, generally, a blank 'B' is used as the seperator. After performing the integer function, the result will be M number of 0, which will remain on the tape.

Ex:- Addition of two integer.

Consider the function $f(x, y) = x + y$.

Solution:-

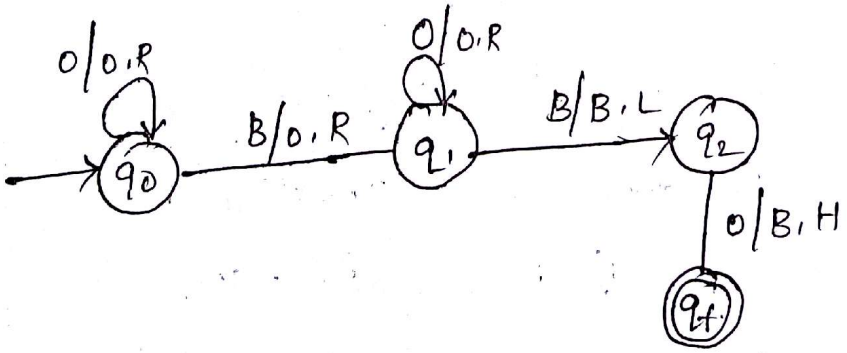
The i/p tape is $0^x B 0^y$. The calculation process is as follows.

(1) Traverse x number of 0 up to 'B'.

(2) On getting B change the state & replace 'B' by 0

(3) Traverse the 2nd string 'B' means it is the end of the second string. change the state & tranverse left.

(4) Replace the last '0' of the string by B & halt.



Ex:- Compute the function $f(x) = x+2$.

Sol

The i/p tape is 0^7.

Calculation Process

→ Traverse x number 0 upto B.

→ change B by '0' & change the state.

→ Replace the Second B by '0' & halt.

Universal Turing Machine:-

A Turing Machine can be said to be a Univers Turing Machine if it can accept

a) the i/p data and.

b) the algorithm for performing the different tasks.

All the Turing Machines designed for each seperate task by

→ increasing the no of read-write heads

→ Increasing the no of i/p tapes

→ Increasing the dimension of moving the read-write head.

→ Adding special purpose memory like stack

Linear Bounded Automata (LBA).

An LBA is a special type of TM with restricted tape space.

For TM, the i/p tape is virtually infinite in both directions whereas the allowable p/p tape space of LBA is linear function of the length of p/p string.

The limitation of tape space makes LBA a real model of computer that exists than TM is some respect.

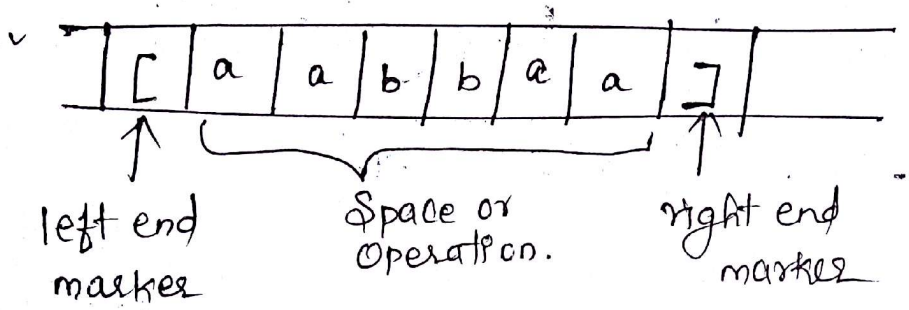


fig:- Linear Bounded Automata.

An LBA is a 7 tuple non-deterministic Turing Mac

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

where Q finite set of states.

Σ: finite set of p/p alphabets with two end markers

Γ: finite set of allowable tape symbols except two end markers '[' and ']'.

q_0: S. state

B: Blank

F: F. state

δ: Transitional function.

where δ is a mapping from

$Q \times \Gamma \rightarrow Q \times 2^Q \times \Gamma \times \{L, R\}$ with two more transitional function

$$\delta(q_p, [) \rightarrow (q_1, [, R).$$

$$\delta(q_1,]) \rightarrow (q_1,], L).$$

An LBA accepts context-sensitive lang. It is powerful than the PDA but less powerful than TM.

Example:- Construct a Linear bounded automata for the following context sensitive language.

$$L = \{a^n b^n c^n \mid n \geq 0\}$$

Solution:- The design concept is

(1) on each pass, the machines matches the leftmost 'a'. 'b' & 'c' & replaces them with X.

(2) If no a, b & c are left & it gets '[' then it halts.

The transition (δ) function are.

$\delta(q_0, [) \rightarrow (q_1, [, R)$

$\delta(q_1, a) \rightarrow (q_2, X, R)$

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

$\delta(q_2, b) \rightarrow (q_3, X, R)$

$\delta(q_3, b) \rightarrow (q_3, b, R)$

$\delta(q_3, c) \rightarrow (q_4, X, L)$

$\delta(q_4, X) \rightarrow (q_4, X, L)$

$\delta(q_4, b) \rightarrow (q_4, b, L)$

$\delta(q_4, a) \rightarrow (q_4, a, L)$

$\delta(q_4, [) \rightarrow (q_1, [, R)$.

$\delta(q_1, X) \rightarrow (q_1, X, R)$

$\delta(q_2, X) \rightarrow (q_2, X, R)$

$\delta(q_3, X) \rightarrow (q_3, X, R)$

$\delta(q_1,]) \rightarrow (q_1,], H)$

Unrestricted Grammar:-

A Grammar (V_N, Σ, P, S) is called an unrestricted grammar if all its productions are in the form $LS \to RS$, where $LS \in (V_N \cup \Sigma)^+$ and $RS \in (V_N \cup \Sigma)^*$.

In the chomsky hierarchy, type 0 or unrestricted grammar is the superset of all grammars & thus it accepts more languages than type 1.

Unrestricted grammar is accepted by the TM & the lang accepted by a TM is called recursively enumerable lang.

Theorem: Any language set generated by an unrestricted grammar is recursively enumerable.

Proof:-

Let $G = (V_N, \Sigma, P, S)$ be an unrestricted grammar. A grammar defines a set of ~~Productio~~ procedures for enumerating all strings $\in L(G)$.

→ List all $w \in L(G)$ such that w is derived from the Production rules P in one step. As S is the start symbol, we can write that w is the set of strings derived from S in one step. It can be written as

$$S \to w.$$

→ List all $w \in L(G)$ such that w is derived from the Production rules P in two steps. It is written as

$$S \to x \to w.$$

3) The intersection of two recursive languages is recursive.

Proof:- Let L_1 and L_2 be two recursive lang., so there exis two TM say T_1 and T_2 accepting L_1 and L_2 respectively.

A string w is accepted by M if it is accepte by both T_1 and T_2 and rejected if any of them reject.

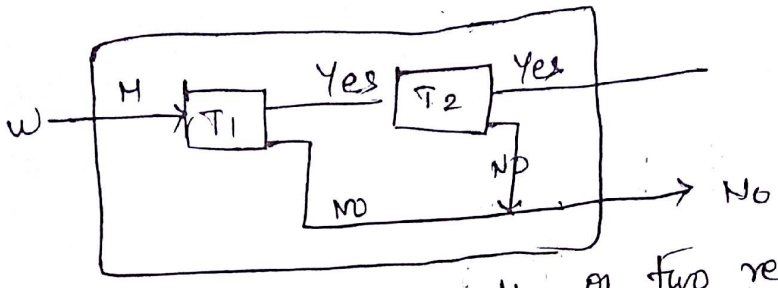


fig:- intersection of two recursive language.

Algorithm

On i/p w

Run T_1 on w

if T_1 accepts w then

Run T_2 on w

if accepts w then

accept

else reject

else reject w.

4) The intersection of two recursively enumerable language is recursively enumerable.

It is similar to the above property, excluding the rejection part as given in figure.

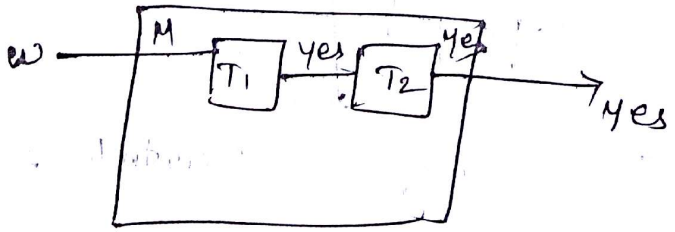


fig:- intersection of two recursively enumerable lang.

Properties of Recursive & Recursively Enumerable Language

1) The Union of two recursively languages is recursive.

Proof:- Let L_1 and L_2 be two Recursive lang, so there exist two TMs say T_1 and T_2 accepting L_1 and L_2 respectively. Let us Construct a new TM M by combining T_1 and T_2 as given in figure.

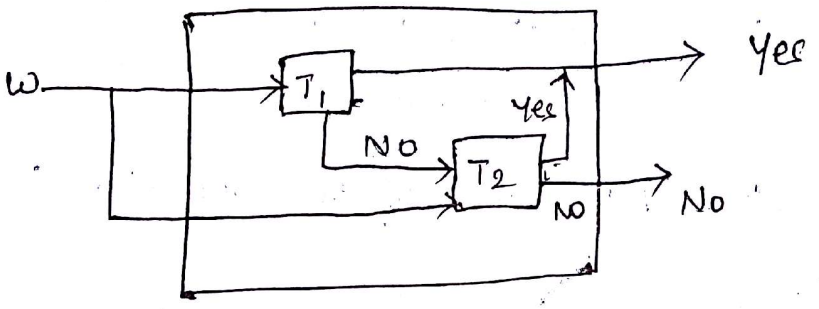


fig:- Union of Two Recursive languages.

As there exist a TM for $L_1 \cup L_2$ which guarantees t halt, it is thereby proved that $L_1 \cup L_2$ is recursive.

2) The Union of two recursively enumerable language is recursively enumerable.

Proof:- Let L_1 and L_2 be two recursively enumerable lang. so there exist two TMs, say T_1 and T_2 accepting L_1 and L_2 respectivel Let us construct a new TM M by combining T_1 and T_2 as denoted in figure.

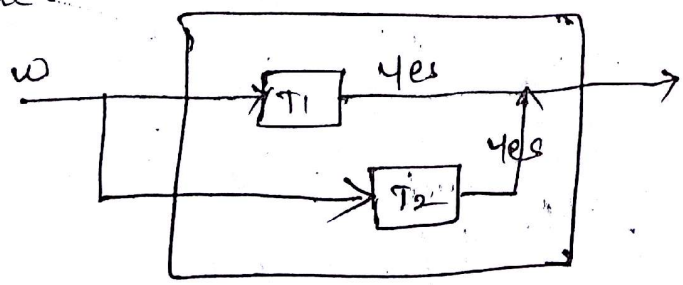


fig:- Union of two recursively enumerable langua

Language

Turing Machine to unrestricted Grammar.

A TM accepts type 0 lang which is generated by type 0 grammar. This grammar is constructed directly from the TM's transitional functions.

Let the String to be traversed by a TM M be S which is enclosed between two end markers $ S $. The IDs are enclosed within brackets. $[q_0 \$ S \$]$ to $[q_f B]$

Here $q_0 \rightarrow$ Initial state $q_f \rightarrow$ final or halt state.

The length of the ID may change if the read-write head of the TM reaches the left hand side or right hand side bracket

The production of the grammar equivalent to the transition of ID are divided into two steps

(i) No change in length &

(ii) change in length.

Input / State	a_1 ..... a_j
q_1	
:	
q_p	

fig:- Transition Table.

No change in length:-

Right move: If there is a δ

$\delta(q_i, a_j) \rightarrow (q_k, a_l, R)$ of the TM, then the equivalent production rule of the grammar is

$$q_i a_j \rightarrow a_l q_k.$$