Type 1 Formalism : Context Sensitive Grammar

- Type 1 languages are described by Type 1 grammars. LHS of Type 1 grammar is not longer than RHS.

- Type 1 Grammars (Context Sensitive Grammars) are represented by the productions of the form

  $\alpha \rightarrow \beta$ where $|\alpha| \leq |\beta|$. $\alpha, \beta \in (V+T)$

- Type 1 Languages are recognised using Linear Bounded Automata (LBA)

eg: of Context Sensitive Grammar is

$S \rightarrow SBC$

$S \rightarrow aC$

$B \rightarrow a$

$CB \rightarrow BC$

$Ba \rightarrow aa$

$C \rightarrow b$.

no ε in CSL

LBA can accept ε but CSG do not contain ε is a contradiction

LBA is equivalent to CSG as long as languages are ε free

A language is Context Sensitive if and only if it can be generated by a grammar in which every production has the form:

$$\alpha A \beta \rightarrow \alpha X \beta$$

where α, β and X are strings of terminals & non-terminals, with $X \neq \varepsilon$ and A is a non-terminal.

1) Check whether the string abaabb is accepted by the following Context Sensitive Grammar.

$S \rightarrow SBC$
$S \rightarrow aC$
$B \rightarrow a$
$CB \rightarrow BC$
$Ba \rightarrow aa$
$C \rightarrow b$

Ans) $S \Rightarrow SBC$ (using $S \rightarrow SBC$)
$\Rightarrow SBCBC$ (using $S \rightarrow SBC$)
$\Rightarrow aCBCBC$ (using $S \rightarrow aC$)
$\Rightarrow abBCBC$ (using $C \rightarrow b$)
$\Rightarrow abaCBC$ (using $B \rightarrow a$)
$\Rightarrow abaBCC$ (using $CB \rightarrow BC$)
$\Rightarrow abaaCC$ (using $B \rightarrow a$)
$\Rightarrow abaabC$ (using $C \rightarrow b$)
$\Rightarrow abaabb$ (using $C \rightarrow b$)

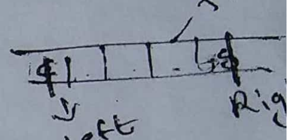
Thus the string abaabb is accepted by the given grammar.

Linear Bounded Automata

A Linear Bounded Automata (LBA) is a restricted form of non-deterministic turing machine which has a single tape and whose length is not infinite but bounded by a linear function of the length of the input string.

It is denoted as : (9-tuples)

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



Q - is the finite set of states.

Σ - is the finite set of input symbols.

Γ - finite non-empty set of tape symbols.

$\Gamma \subseteq (\Sigma \cup \{B\})$

B - $B \in \Gamma$ is the blank.

$q_0 \in Q$ is the initial state

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

header has to move in b/w the symbols ¢ & $

¢, $ are symbols in Σ.

¢ - left end marker, prevents read/write head from getting off the left end of tape.

$ - right end marker, prevents read/write head from getting off the right end of tape.

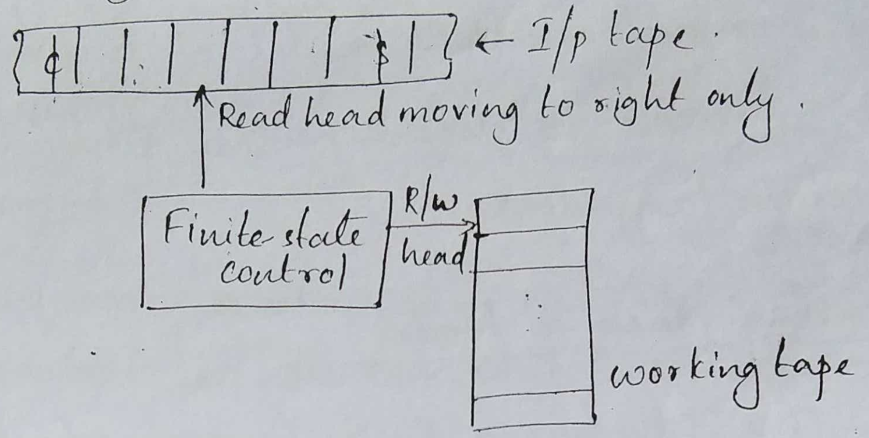
$\delta - Q \times \Gamma \longrightarrow 2^{Q \times \Gamma \times \{L, R\}}$

ie δ is the transition function mapping (q, x) onto (q', y, D) where D denotes the direction of movement of R/W head ; D = L or R

(left) (right)

Model of LBA

There are two tapes : one is called the i/p tape & the other is working tape. On the i/p tape, the head never prints and never moves to the left. On the working tape, the head can modify the contents, without any restriction in any way.



Whenever we process any string in LBA, we assume that the i/p string is enclosed within the endmarkers ¢ & $.

Q: Consider an LBA defined as :

$Q = \{q0, q1, q2, q3, q4\}$

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

$\Gamma = \{a, b, c, \#\}$

$\delta(q1, a) = (q2, \#, R)$ $\delta(q3, c) = (q4, \#, L)$ . . .

$\delta(q2, a) = (q2, a, R)$

$\delta(q2, \#) = (q2, \#, R)$

In the above, the transition $\delta(q1, a) = (q2, \#, R)$ means, LBA on state q1, head points to symbol a, remains in state q2, replaces a with # and head turns towards right by one cell.

05/02/22

Module 5

Turing Machine

Turing machine provides an ideal theoretical model of a computer. Turing m/c was developed by ~~alan~~ Alan Turing in 1936. Turing m/cs are useful in several ways.

* It accepts type-0 languages.
* It can also be used for computing functions
* Turing m/cs are used for determining the undecidability of certain languages.
* It is also used for measuring the space & time complexibility of pblms.

Turing machine Model

Turing m/c can be thought of as finite control connected to a R/W (Read/write) head. It has one tape which is divided into a no. of cells.

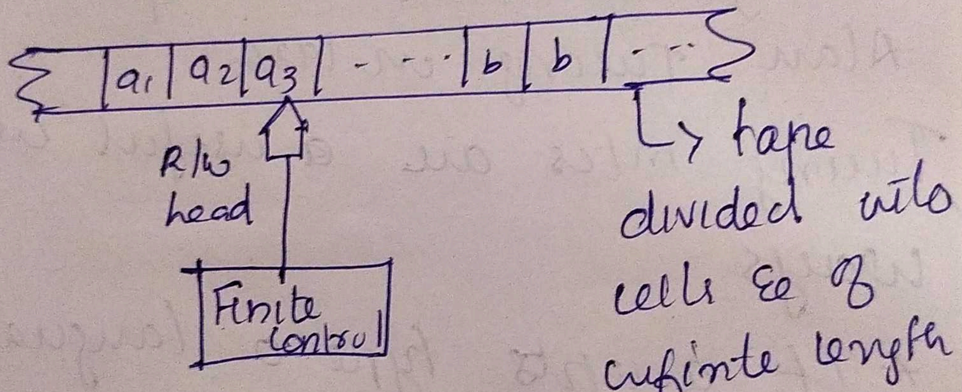


Fig: Turing m/c model

Each cell can store only one symbol. The I/p to & the O/p from the finite state automaton are effected by the R/W head which can examine one cell at a time. In one move, the m/c examines the present symbol under the R/W head on topa the tape ie the present state of an automaton to determine.

i) a new symbol to be written on the tape in the cell under the R/W head.

ii) a motion of the R/W head along the tape either the head moves one cell left (L), or one cell right (R).

iii) the next state of the automaton and

iv) whether to ~~halt~~ halt or not.

Defenition of TM

A turing m/c M is a 7 tuple.

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

Q = is a finite nonempty set of states

Γ = is a finite non empty set of tape symbols.

$b \in \Gamma$ is the blank

Σ - is a nonempty set of i/p symbols & is a subset of Γ & $b \notin \Sigma$

δ - is the transition fn mapping (q, x) onto to (q', y, D) where D denotes the direction of R/w. head movement of $D = L$ or R according as the movt. is to the left or right.

$q_0 \in Q$ is the initial state &
$F_0 \in Q$ is the set of final states

Notes:-

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

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

Representation of Turing Machines

We can describe a TM employing.

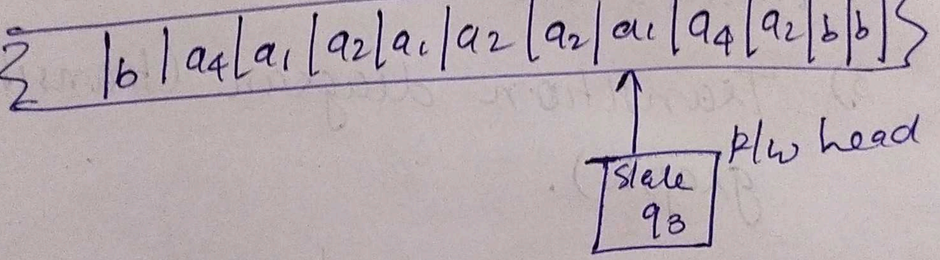
1) Instantaneous descriptions using move rlns
2) Transition table
3) Transition diagram (transition graph).

Representation by instantaneous descriptions.

An ID of a TM. M is a string $\alpha\beta\gamma$, where β is the present state of M

$\alpha\gamma$ the entire i/p string is split into $\alpha\gamma$, the first symbol of γ is the current symbol a under the R/W head & γ has all the subsequent symbols of the i/p string ie the string α is the substring of the i/p string formed by all the symbols to the left of a.

eg: A snapsot of TM is shown in the figure.



The present symbol under the R/w head is a_1. The present state is q_3. So, a_1 is written to the right of q_3.

The nonblank symbols to the left of a_1 from the string $a_4 a_1 a_2 a_1 a_2 a_2$ which is written to the left of q_3. The sequence of non blank symbols to the right of a_1 is a_4 ~~a_2~~-a_2.

Thus the ID is as given in fig.

$a_4 a_1 a_2 a_1 a_2 a_2 \quad q_3 \quad a_1 \quad a_4 a_2$

left sequence

present state

symbol under R/W head

Right sequence

Notes

1) For construction the ID, insert the current state in the I/P string to the left of the symbol under the R/W head.

2) The blank symbol may occur as part of left or right substring.

Moves in a TM

As in the case of pda, $\delta(q, x)$ induces a change in ID of the TM. We call this change in ID or move.

Suppose $\delta(q, x_i) = (p, y, L)$. The i/p string to be processed is $x_1 x_2 \ldots x_n$ ie the present symbol under the R/W head is x_i. So the ID before processing x_i is.

$$x_1 x_2 \ldots x_{i-1} q x_i \ldots x_n.$$

After processing x_i, the resulting ID is

$$x_1 x_2 \ldots x_{i-2}\, p\, x_{i-1}\, y\, x_{i+1} \ldots x_n$$

This change of ID is represented by

$$x_1 x_2 \ldots x_{i-1} q x_i \ldots x_n \vdash x_1 x_2 \ldots x_{i-2} p x_{i-1} y x_{i+1} \ldots x_n.$$

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

$$x_1 x_2 \ldots x_{i-1} q x_i \ldots x_n \vdash x_1 x_2 \ldots x_{i-1} y x_{i+1} \ldots x_n.$$

The symbol ~~$\vdash^*$~~ $\vdash^*$ denotes the reflexive- transitive closure of the r/n $\vdash$

If $I_1 \vdash^{*} I_n$, then we can split this as $I_1 \vdash I_2 \vdash I_3 \vdash \ldots \vdash I_n$ for some ID's $I_2 \ldots I_{n-1}$

Representation by Transition Table.

If $\delta(q, a) = (\gamma, \alpha, \beta)$, we write $\alpha \beta \gamma$ under the a column & in the q row, ~~it~~ it $(\alpha, \beta \gamma)$ means that α is written in the current cell, 'β' gives the movement of the head (L or R) & γ denotes the new states into which the TM enters.

eg:- Consider a TM, with 5 states $q_1 \ldots q_5$. The tape symbols are 0, 1 & b. The transition table is given as follows.

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

Representation by Transition Diagram

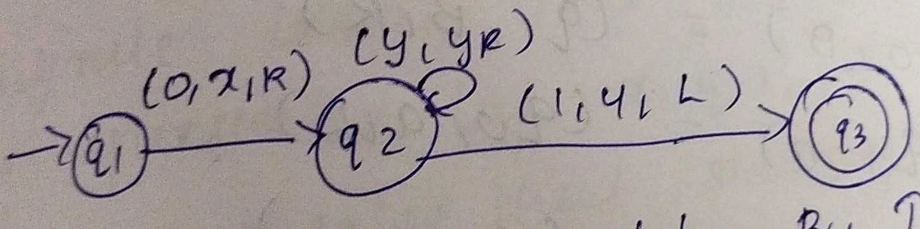
The states are represented by vertices. Directed edges are used to represent transition of states. The labels are triples of the form (α, β, γ), where $\alpha, \beta \in \Gamma$ ie $\gamma \in (L, R)$. When there is a directed edge from q_i to q_j with label (α, β, γ), it means that

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

every edge in the transition s/m can be represented by a 5-tuple $(q_i, \alpha, \beta, \gamma, q_j)$

eg:

Q/Γ	x	y	0	1
→q_1			xRq_2	
q_2		yRq_2		yLq_3
q_3				



Language acceptability By T.M

Let us consider the TM $M = (Q, \Sigma, \Gamma, \delta, q_0, b, F)$ A string w in Σ^* is said to be accepted by M if $q_0 w \vdash^* \alpha_1 p \alpha_2$ for some $p \in F$ & $\alpha_1, \alpha_2 \in \Gamma^*$ w if the M/c M does not accept

M either halts in a ~~on~~ non-accepting state or does not halt (M can enter in an infinite loop & never halts).

eg:- $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $F = \phi$, δ as follows.

$F = \{a, b, B\}$;

$\delta(q_0, a) = (q_1, a, R)$

$\delta(q_0, b) = (q_1, b, R)$

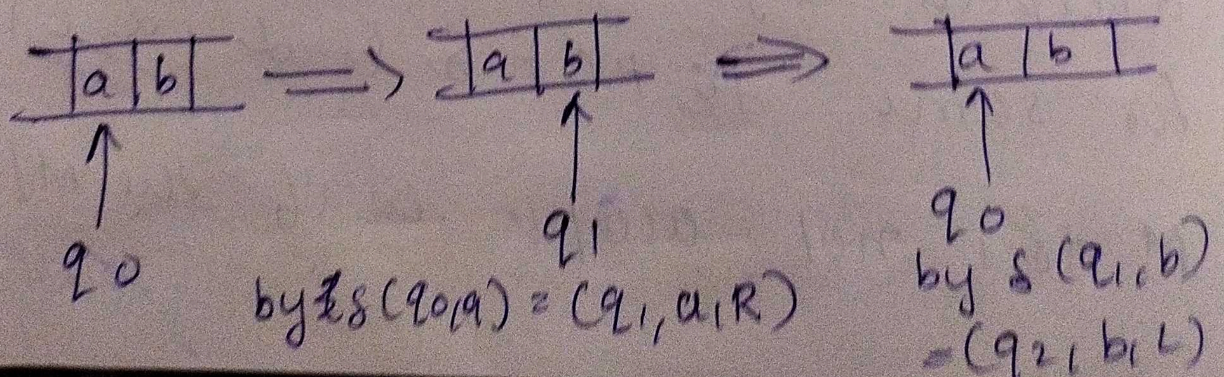
$\delta(q_0, B) = (q_1, B, R)$

$\delta(q_1, a) = (q_0, a, L)$

$\delta(q_1, b) = (q_0, b, L)$

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

Suppose that the tape initially contains 'ab' with the R/W head is on the symbol 'a'.



Here in 3rd step, m/c goes back to state q_0. Again it is the orginal state i.e the sequence of moves starts again. Here TM will run forever with the R/W head moving alternately right then left, but making no modifications to the tape. This is an instance of a TM that does not halt, i.e TM is in infinite loop. This situation can be represented by $x_1 q x_2 \vdash^{*} \infty$. This indicates that starting from initial ~~con~~, $x_1 q x_2$, the m/c never halts.

Q: Consider the TM 'M' described by the transition table given in the following table. Describe the processing of a) 011 b) 0011 c) 001 using IDs. Which of the following strings are accepted by M?

Q/Γ	0	1	x	y	B
→ q_1	xRq_2				BRq_5
q_2	$0Rq_2$	yLq_3		yRq_2	
q_3	$0Lq_4$		xRq_5	yLq_3	
q_4	$0Lq_2$		xRq_1		
q_5				yRq_5	BRq_6.
q_6					

a) 011

$q_1, 011 \vdash x q_2 11 \vdash q_3 x y 1 \vdash$

$x q_5 y 1 \vdash x y q_5 1$

$\delta(q_5, 1)$ is not defined so the i/p string 011 is not accepted.

b) 0011

$q_1, 0011 \vdash x q_2 011 \vdash x 0 q_2 11$

$\vdash x q_3 0 Y 1 \vdash q_4 x 0 Y 1$

$\vdash x q_1 0 Y 1 \vdash x x q_2 Y 1 \vdash$

$x x y q_2 1 \vdash x x q_3 y y \vdash$

$x q_3 x y y \vdash x x q_5 y y \vdash$

$x x y q_5 y \vdash x x y y q_5 B \vdash$

$x x y y B \underline{q_6}$

0011 is accepted.

0001

$q_1 001 \vdash x q_2 01 \vdash x 0 q_2 1 \rightarrow$

$x q_3 0 y \vdash q_4 x 0 y \vdash x q_1 0 y$

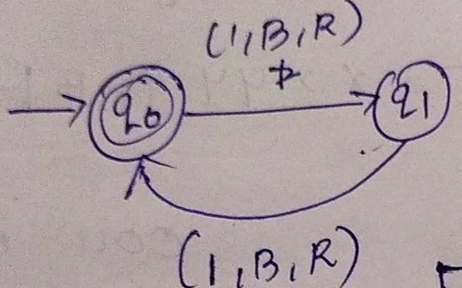
$\vdash x x q_2 y 1 \vdash x x y q_2 B$

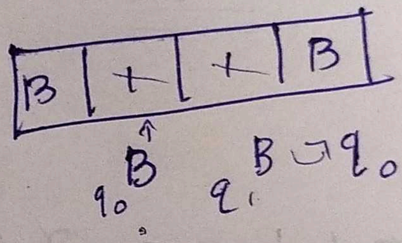
M halts as q_2 is not an accepting i.e 001 is not accepted by M

Designing a Turing M/C

Q Design a turing machine to recognize all strings consisting of even no. of 1's.

→





$M = (\{q_0, q_1\}, \{1\}, \{1, B\}, \delta, q_0, B, \overset{q_0}{\cancel{F}})$

$\delta(q_0, 1) = (q_1, B, R)$

$\delta(q_1, 1) = (q_0, B, R)$

Consider the string $w = \cancel{(11)}\ 11$

~~$q_0, 11 \vdash Bq_1 BR$~~

$Bq_0, 11 \vdash Bq_1 1 \vdash BBq_0 \in F$

As q_0 is the final stage le.
The 11 is accepted by M.

$w = 111$

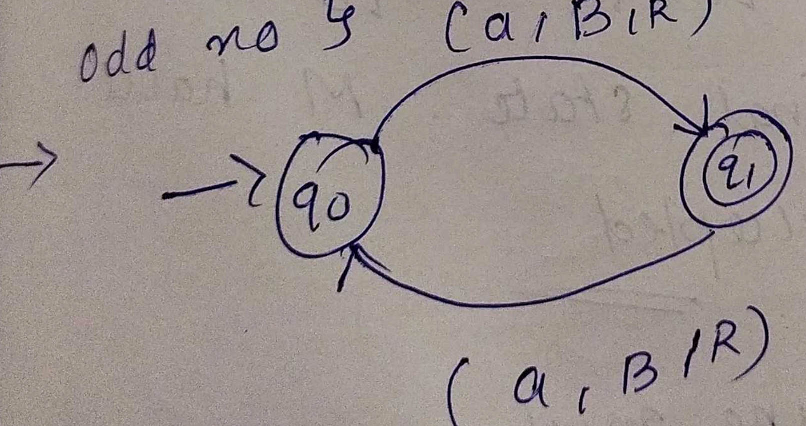
$q_0, 111 \vdash Bq_1 11 \vdash BBq_0 1$

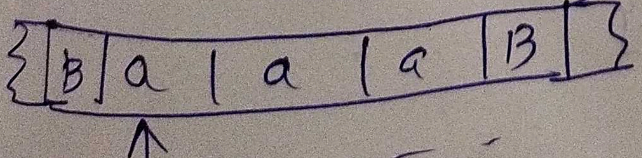
$\vdash BBBq_1$

~~The machine halts~~

As q_1 is not accepting state,
111 is not accepted by M.

Q Design a turing machine to accept the language $L = \{a^n \mid n$ is odd no of





$M = \{ \{q_0, q_1\}, \{a\}, \{a, B\}, \delta, q_0, B, \{q_1\} \}$

$\delta(q_0, a) = (q_1, B, R)$

$\delta(q_1, a) = (q_0, B, R)$

$w = aaa$

$q_0 aaa \vdash B q_1 aa \vdash BB q_0 a$

$\vdash BBB q_1 \in F$

String accepted.

$w = aa$

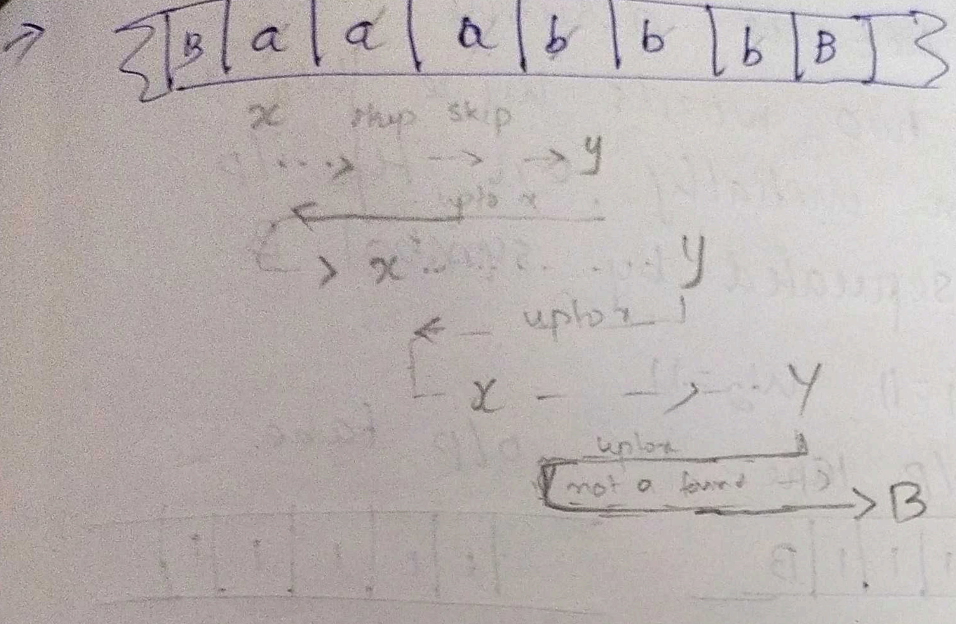
$q_0 aa \vdash B q_1 a \vdash BB q_0 \notin F$

q_0 is not final state. M halts

& String not accepted

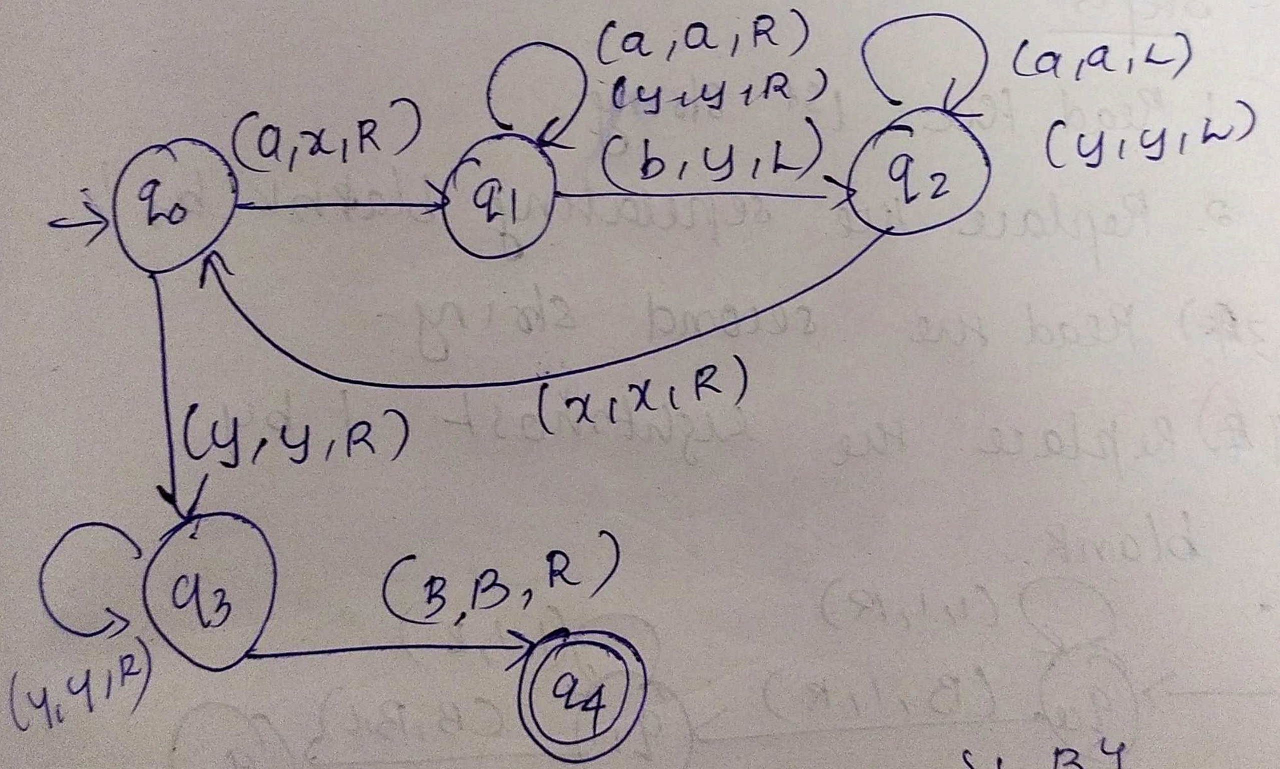
Q Design a turing machine

$L = \{a^n b^n / n \geq 1\}$



If one 'a' is there then find out whether a matching 'b' can be found

(If y comes after x, then move to B, skip & move to final)



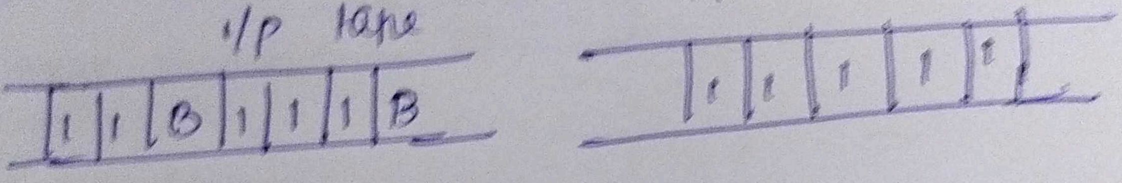
Q. Design a TM over {1, B} which can compute concatenation fn over $\Sigma = \{1\}$.

soln: Let us assume that the two words w_1 & w_2 are written initially on the i/p tape separated by symbol B.

eg: $w_1 = 11$ $w_2 = 11$

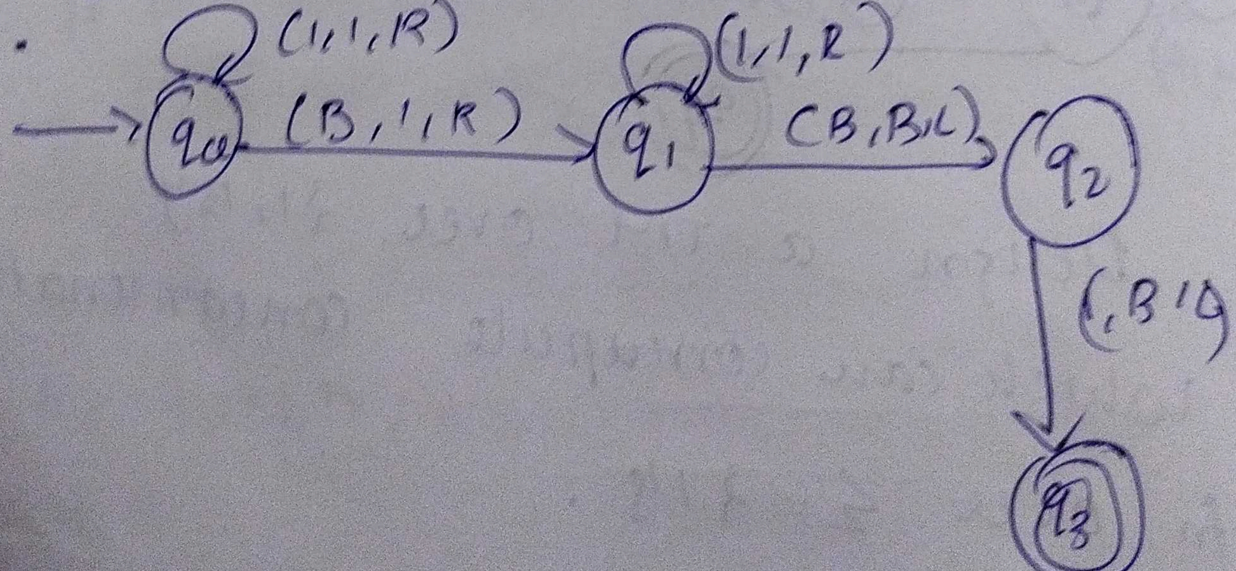
o/p tape

i/p tape



Steps

1. Read the 1st string
2. Replace the separating blank by 1
3) Read the second string.
4) Replace the rightmost 1 by blank.



$M = (\{q_0, q_1, q_2, q_3\}, \{1\}, \{1, B\},$
$\delta, q_0, B, \{q_3\})$

δ is given in the following table

Q/F	1	B
$\rightarrow q_0$	$1Rq_0$	$1Rq_1$
q_1	$1Rq_1$	BLq_2
q_2	BLq_3	
q_3		

$w = 1B1$, $w_1 = 1$ $w_2 = 1$

$q_0 1B1 \vdash 1q_0B1 \vdash 11q_1 B 1 \vdash$

$11q_2 1B \vdash 1q_3 1BB$

$111q_1 B \vdash$

$w_1 = 11$ $w_2 = 111$

$1q_0 1B111 \vdash 11q_0 B111 \vdash$

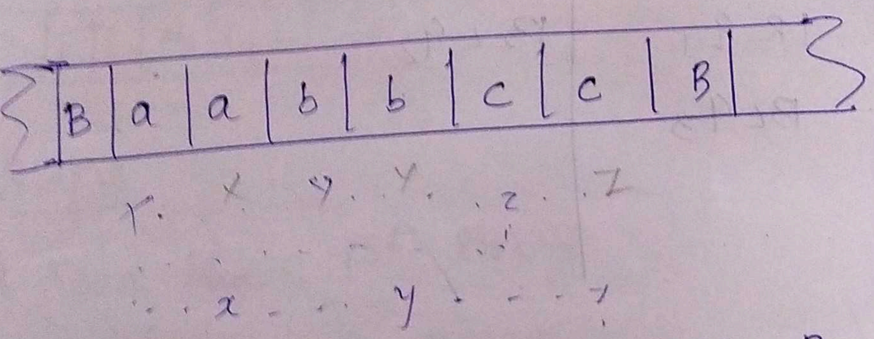
$q_0 11B111 \vdash 1q_0 1B111 \vdash 1111q_1 11 \vdash$

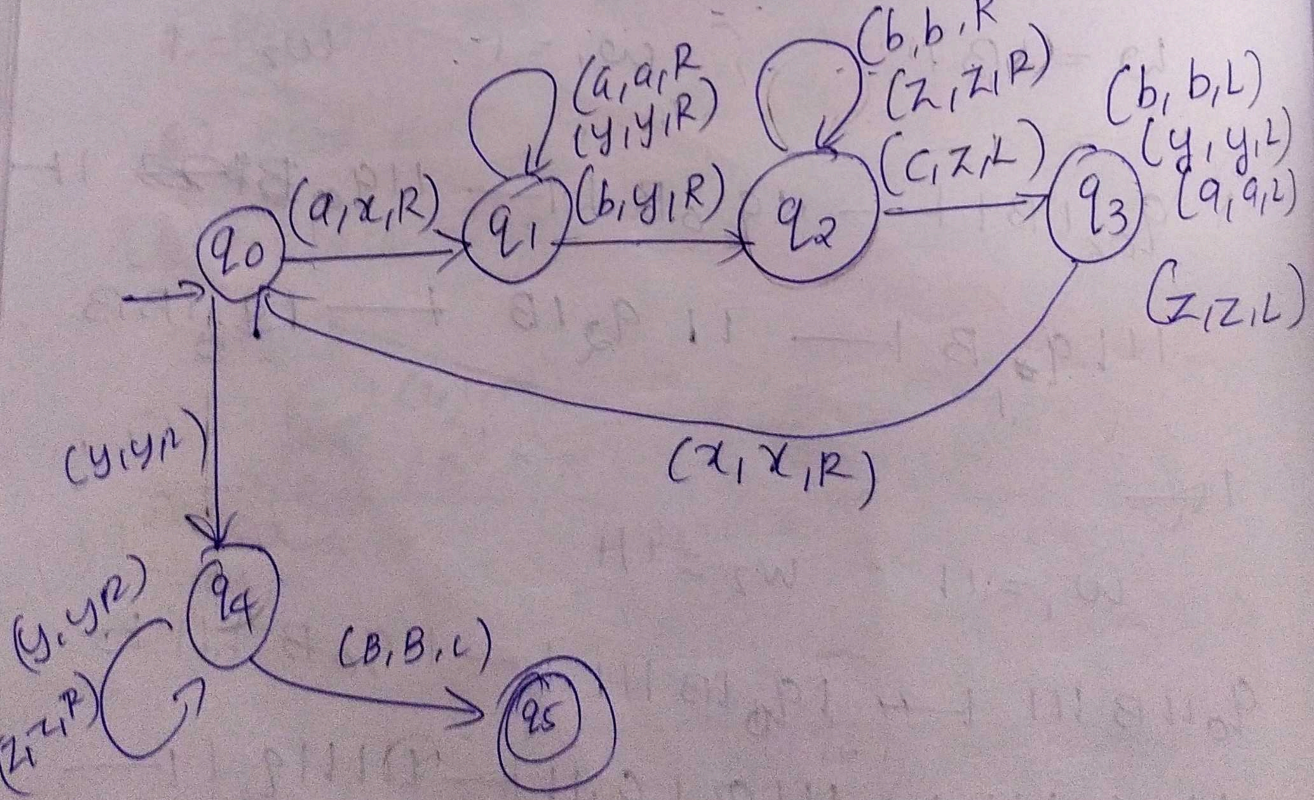
$111q_1 111 \vdash 111q_1 1q_1 11 \vdash$

$111\$11\$1q_1B \vdash 1111q_2{}'B \vdash$

$11111q_3{}'B$

Q. Design a turing machine for the language $L = \{a^n b^n c^n\}$





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

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

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

$\delta(q_1, b) = (q_2, y, R)$

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

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

$\delta(q_2, c) = (q_3, z, L)$

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

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

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

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

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

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

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

$\delta(q_4, B) = (q_5, B, L)$

$q_0\, 000111222 \vdash$

$x\, q_1\, 00111222 \vdash x0q_1\, 0111222$

$\vdash x00q_1\, 111222 \vdash$

$q_0\, aaabbccc \vdash x\, q_1\, aabbbccc$

$\vdash xa\, q_1\, abbbccc \vdash xaa\, q_1\, bbbccc$

$\vdash xaaay\, q_2\, bbcccc \vdash xaaybb\, q_2\, bccc$

$\vdash xaaybb\, q_2\, ccc \vdash$ ~~xaayy~~

$xaayb\, q_3\, b\, z\, cc$

$q_0\, aabbcc \vdash x\, q_1\, abbcc \vdash$

$xa\, q_1\, bbcc \vdash xay\, q_2\, bcc \vdash$

$xayb\, q_2\, cc \vdash xay\, q_3\, bzc$

$\vdash xa\, q_3\, ybzc \vdash x\, q_3\, aybzc$

$\vdash q_3\, xaybzc \vdash x\, q_0\, aybzc \vdash$

$\vdash xx\, q_1\, ybzc \vdash xxy\, q_1\, bzc \vdash$

$xxyyq_2zc \vdash xxyq_3yz$

$xxyyzq_3c \vdash xxyyq_3zz \vdash$

$xxyyq_2zc \vdash xxyyzq_2c$

$\vdash xxyyq_3zz \vdash xxyq_3yzz$

$\vdash xxq_3yyzz \vdash xq_3xyyzz$

$\vdash xxq_0yyzz \vdash xxyq_4yzz$

$\vdash xxyyq_4zz \vdash xxyyzq_4z$

$\vdash xxyyzzq_4B \vdash xxyyzzBq_5$

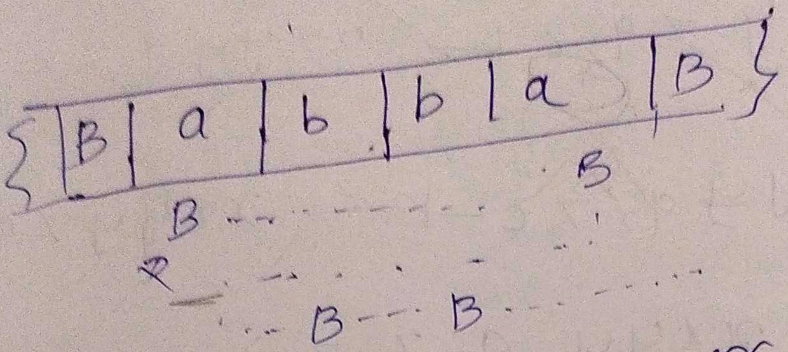
$q_5 \in F$

String accepted

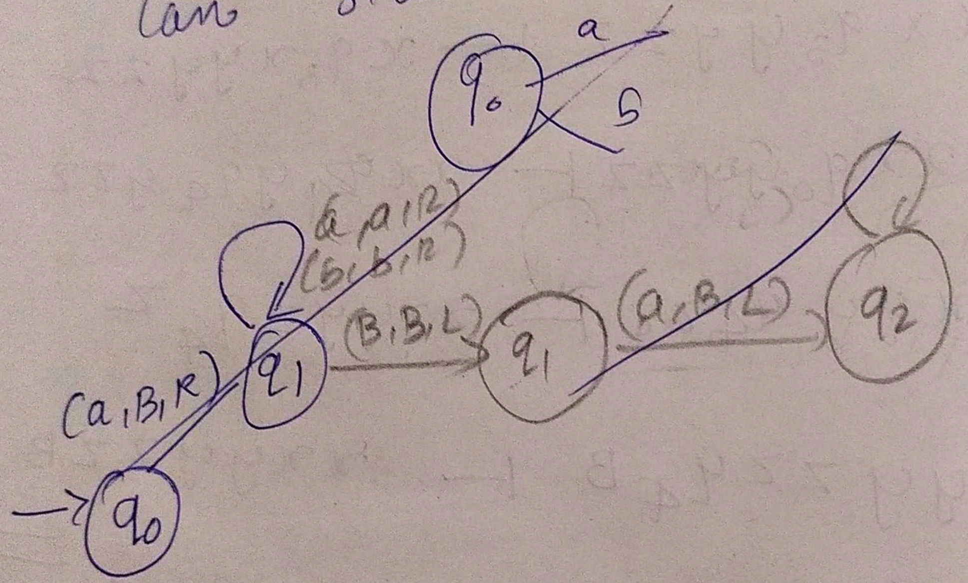
Q. 1) Construct a turing machine that $L = \{ww^R / w \in (a, b)^*\}$

Q 2) Construct a TM that displays the Language $L = \{w \in (0,1)^* / w$ has equal no. of 0's & 1's. $\}$

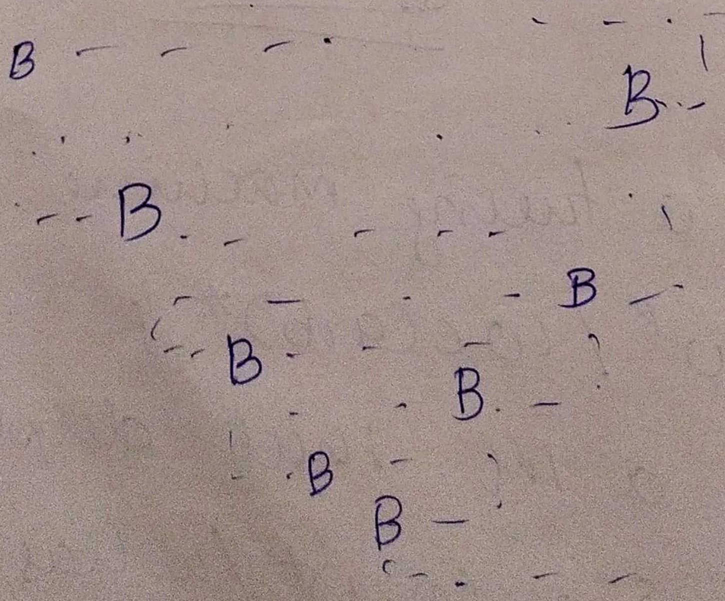
1) $L = \{ww^R / w \in (a,b)^*\}$

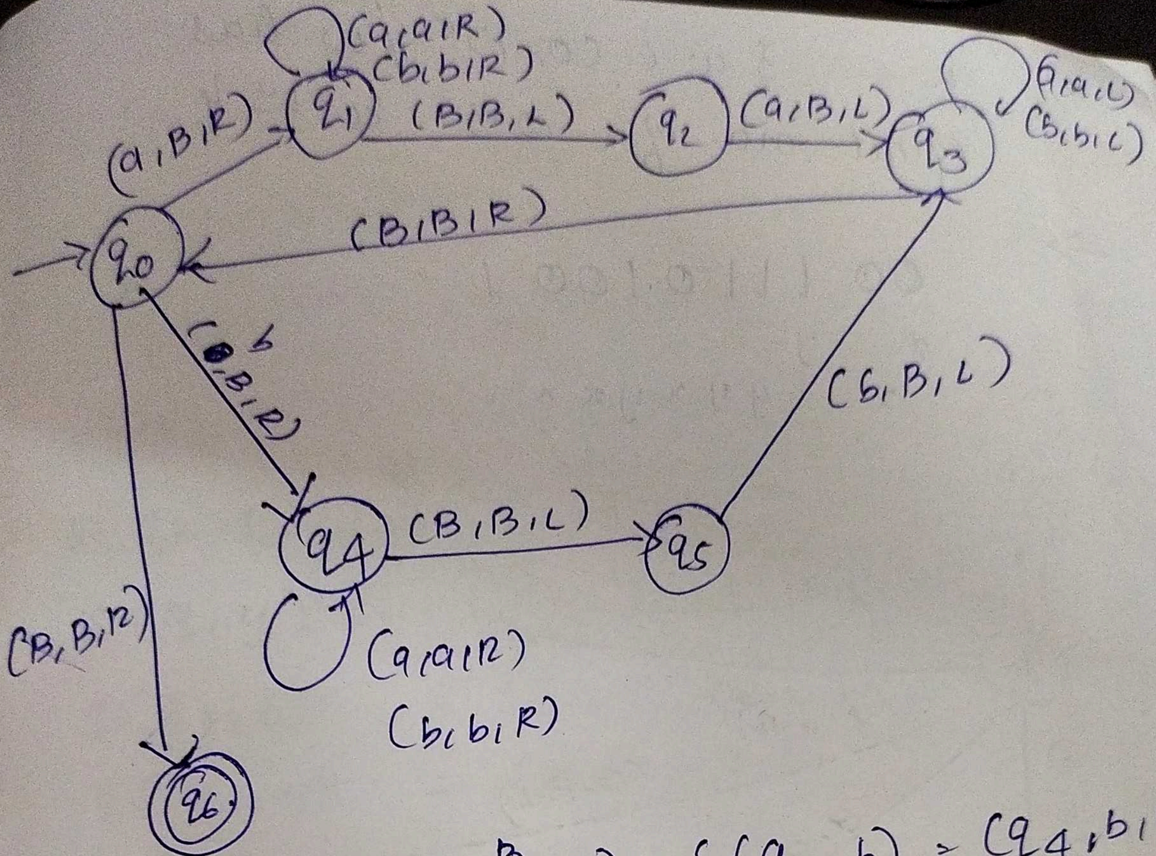


Can start with a or b



B a b b a a b b a B





$\delta(q_0,a) = (q_1, B, R)$

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

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

$\delta(q_1,B) = (q_2,B,L)$

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

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

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

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

$\delta(q_0,b) = (q_4,B,R)$

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

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

$\delta(q_4,B) = (q_5,B,L)$

$\delta(q_5,b) = (q_3,B,L)$

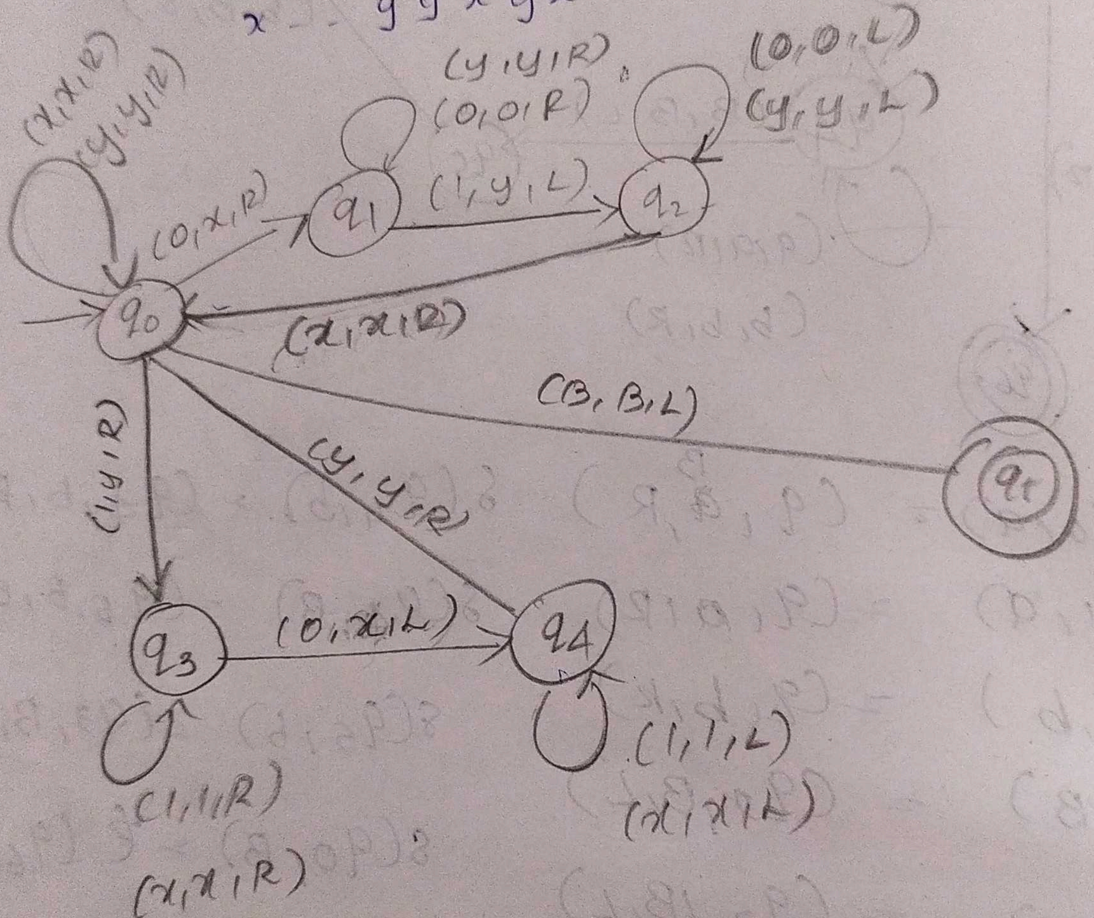
$\delta(q_0,B) = (q_6,B,R)$

Q) $L = \{ w \in (0,1)^* \mid w$ has equal no. of 0's & 1's $\}$

→ 0 0 1 1 1 0 1 0 0 1

x . . y

x . y y x y x x y



0 0 1 1 1 0 1 0 0 1 B

x . . y

x . y . y

. y . X Y X

X . . y . B →

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

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

$\delta(q_0, x) = (q_0, x, R$

$\delta(q_0, 1) = (q_3, y, R)$

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

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

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

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

$\delta(q_2, y) = (q_2, y, L)$

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

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

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

$\delta(q_3, 0) = (q_4, x, L)$

$\delta(q_4, 1) = (q_4, 1, L)$

$\delta(q_4, x) = (q_4, x, L)$

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

$\delta(q_0, B) = (q_5, B, L)$

$q_0 00010111 \vdash x q_1 0010111$

$\vdash x 0 q_1 010111 \vdash x 0 0 q_1 10111$

$\vdash x 0 q_2 0 y 0111 \vdash x q_2 0 0 y 0111$

$\vdash q_2 x 0 0 y 0111 \vdash x q_0 0 0 y 0111$

$\vdash x x q_1 0 y 0111 \vdash x x 0 q_1 y 0111$

$\vdash x x 0 y q_1 0111 \vdash x x 0 y q_1 111$

$\vdash x x 0 y 0 q_1 111 \vdash x x \cancel{x} 0 y q_2 0 y 11\cancel{0}$

$\vdash x x \cancel{x} 0 q_2 y 0 y 11\cancel{0} \vdash \cancel{x x q x}$

$x x q_2 0 y 0 y 11\cancel{0} \vdash x q_2 x 0 y 0 y 11$

$\vdash x x q_0 0 y 0 y 11 \vdash x x x q_1 y 0 y 11$

$\vdash x x x y q_1 0 y 11 \vdash x x x y 0 q_1 y 11 \vdash$

$x x x y 0 y q_1 11 \vdash x x x y 0 q_2 y y 1 \vdash$

$x x x y q_2 0 y y 1 \vdash x x x q_2 y 0 y y 1 \vdash$

$x x \cancel{x} q_2 \overset{}{\underset{y}{x}} 0 y y 1 \vdash x x \cancel{x} \; x x x q_0 y 0 y y 1$

$\vdash x x x y q_0 0 y y 1 \vdash x x x y x q_1 y y 1$

$\vdash xxxyx\,yq_1y$ $\vdash$

$xxxyxyy\,q_1$ $\vdash$ $xxxxyxyyq_2$

$xxxyxyq_2yy \vdash xxxyxq_2yyy$

$\vdash xxxyq_2xyyy \vdash xxxyxq_0yyy$

$\vdash xxxyxyq_0yy \vdash xxxyxyyq_0y$

$\vdash xxxyxyyyq_0B \vdash xxxyxyyyq_5$

Turing machine as Transducer.

A turing machine can function as a transducer i.e, when a string is given as input to turing machine it produces some output.

Input to the computation is a set of symbols on the tape. At the end of computation whatever remains on the tape is the output.

A function f is said to be computable or turing computable if there exists a turing machine M

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

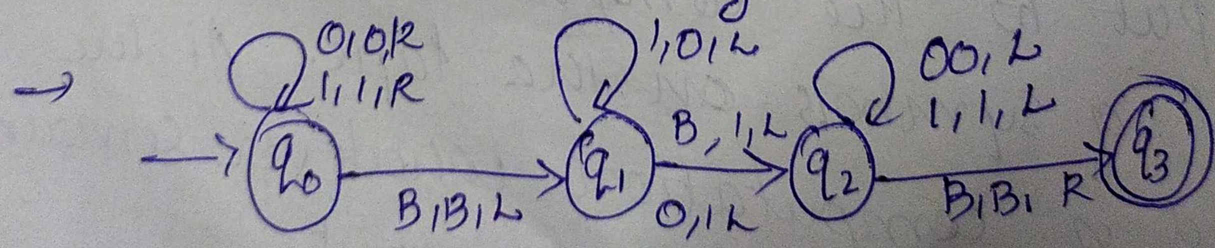
such that $q_0 w \vdash^{*} q_f f(w)$

where . w is the domain of f.

All common mathematical functions are turing computable basic operations like addition substraction multiplication, division can be performed on it, This means that a turing machine is an abstract model of modern computer system.

Q. Construct a turing machine that increments a binary no.

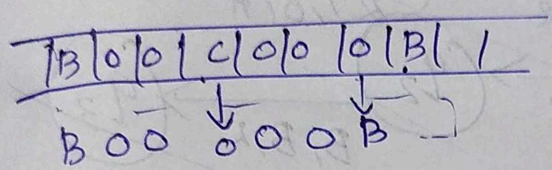
$$\begin{array}{r} 1000 \\ +1 \\ \hline 1001 \end{array}$$



Q2 Design a turing machine that computes the function

$f(m, n) = (m+n)$

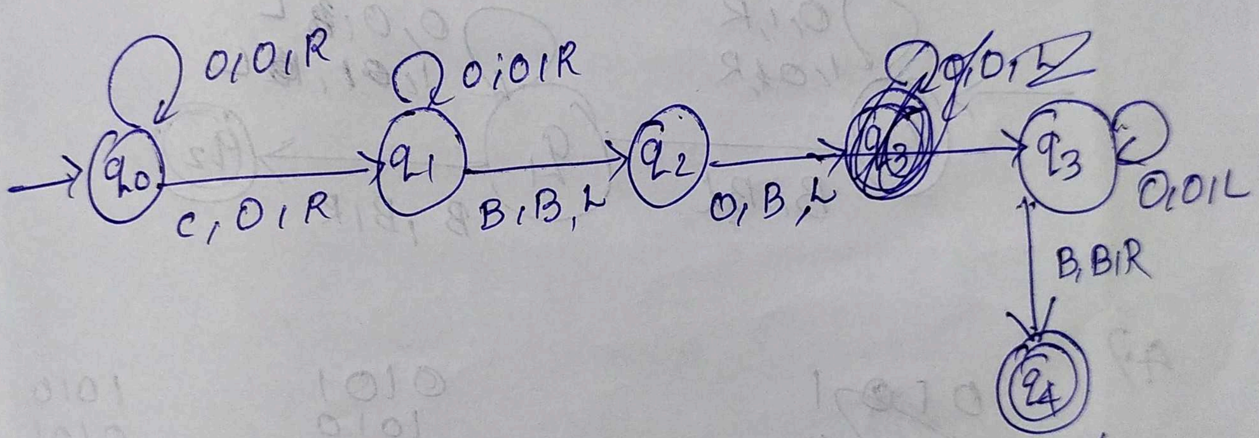
→ 2 + 3



2 represented as 00

3 " 000

~~c~~ c is used to differentiate



Q3 Construct a turing machine that computes the 1's complement of given binary number.

→ ~~B1011~~

Q.4 2's compliment

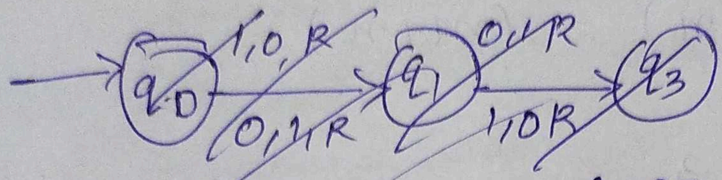
Q.5 Design a turing machine that perform the computation

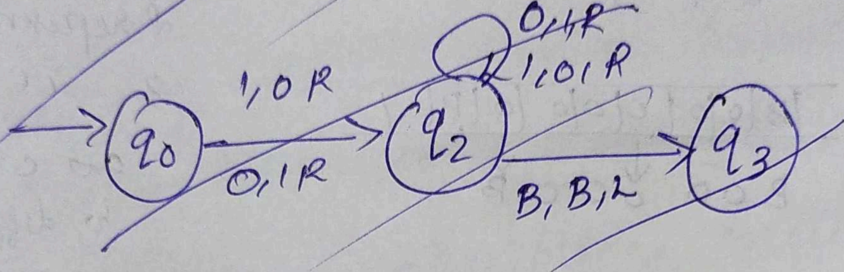
$q_0 w \vdash^* q_f ww$

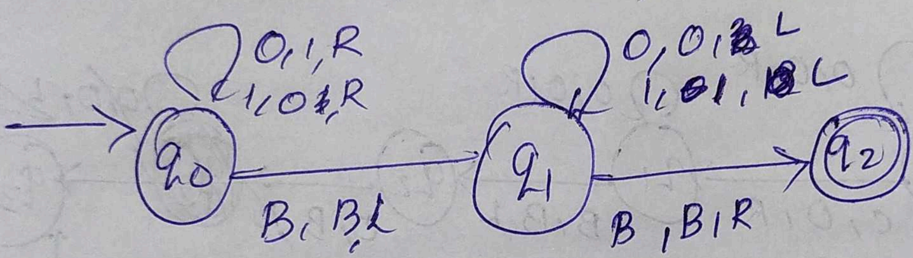
~~Q6) Design a turing machine that~~

3)

B 1 0 1 0 B
$\downarrow\ \downarrow\ \downarrow\ \downarrow$
0 1 0 1





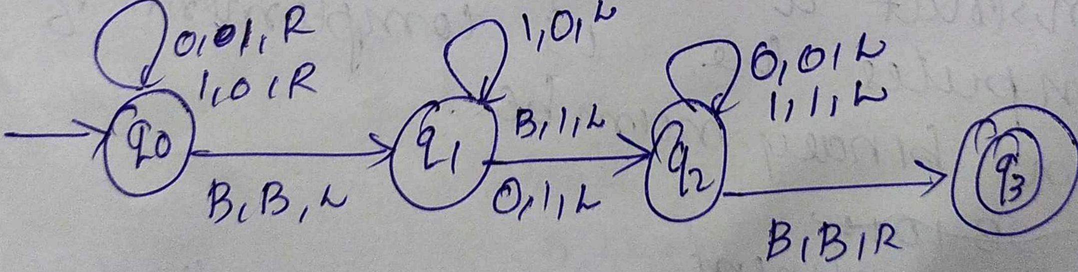


4)

0 1 0 1
1 0 1 0

0101
1010
1011

1010
0101
+1
0



1000
0111
0

1000
0111
+ 1
1000

Q-6)

Q-7)

5)

110010 } R upto first 1, take ones complement
001110 } ie after same.

(6) Construct a turing machine that decrements a binary no.

(7) construct a ~~two~~ turing machine that computes x mode 2

5) B 1111 B ~~1111~~

1111 1111

B 1111 B

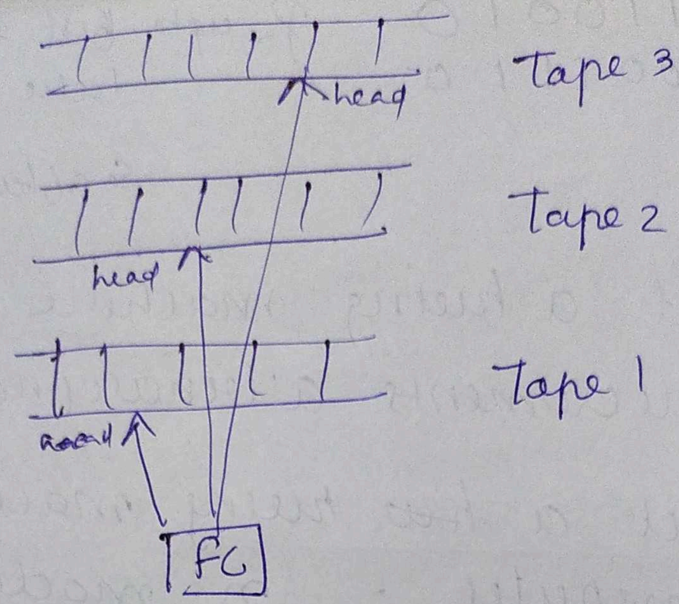
xxxx

B 1111 B

xxxx

Types of Turing machine or variance of Turing machine.

1.) Multitape turing machine :-



A multitape turing machine consist of multiple tapes, each tape has a seperate head. There are n tapes each divided into cells, the first tape holds the input string. Initially all other tapes hold the blank symbol B. A head has 3 possible options

1) To move towards left,
2) To move " right
3) To remain stationary.

All the heads are ~~cont~~ connected to a finite control. Finite control

is in a state at an instant
when a ~~frans~~ transition occurs,
1) finite control may change its state.
2) Head reads the symbol from the current
cell in each tape and writes a
symbdol on it.
3) Each head can move towards
left, right or stay stationary

δ for multitape turing is defined
as ~~$Q \times F^k$~~

$$Q \times \Gamma^k \longrightarrow Q \times F^k \times \{L, R, S\}^k$$

A move depends on the current
state and k tape symbols under
k tape heads.

eg:- A transition function for a
4 tape turing machine is

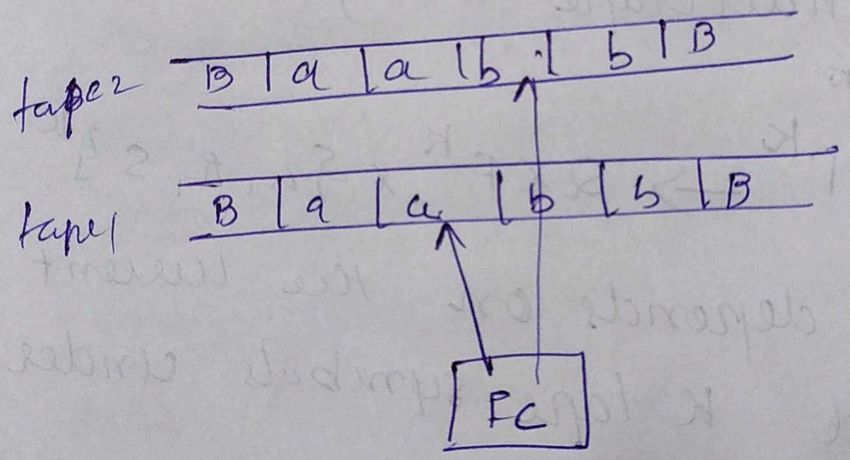
$$\delta(q_0, a_1 a_2 a_3 a_4) = (q_1, \{b_1 b_2 b_3 b_4\}, \{L R L S\})$$

Q Consider the language

$$L = \{a^n b^n / n \geq 1\}$$

→ In a normal turing machine head has to move back and forth to match each pair of symbols a & b. On a multitape turing machine, no such moves are required. This is done by making a copy of input string on another ~~e~~ string

Let i/p string be aabb



Head of tape 1 is positioned on first a of i/p string, head of tape 2 is positioned on first b of i/p string. Now the heads advance on ~~ta~~ both tapes simultaneously towards right &

the string is accepted if there are equal no. of a's & b's in the string. This will happen if head on tape 1 encounters the first b & head on tape encounters first B simultaneously.

Q. Design a multitape turing machine that determines the ones complement of given binary no.

→ ~~q δ(q0, 01~~ 0110

1001

Tape 1 symbol - $\{0,1\}$

Tape 2 " $= \{B\}$ initially

$\delta(q_0, 0B) = \delta(q_0, 01, RR)$

$\delta(q_0, 1B) = \delta(q_0, 10, RR)$

$\delta(q_0, BB) = \delta(q_F, BB, SS)$

Q.5) Construct a turing machine ~~such~~ to perform the computation

$$q_0 w \vdash^{*} q_f w w$$

→ B 1 1 1 1 B

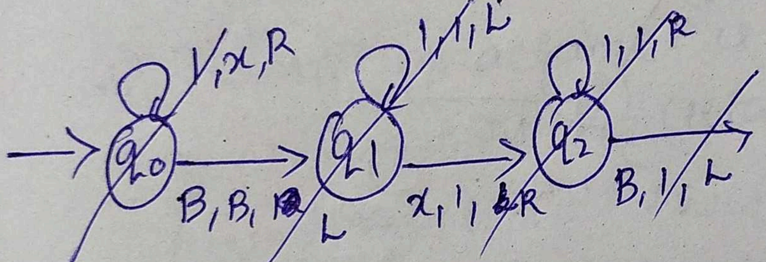
x x x x B

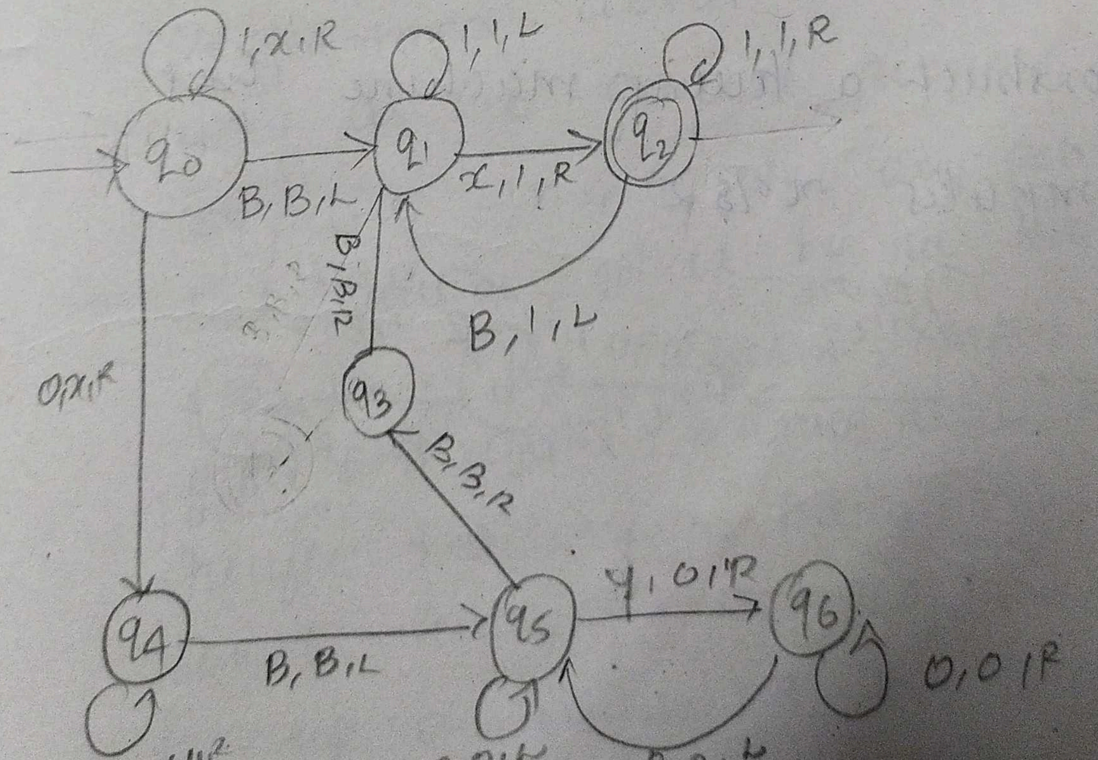
1 1 B

1 1 1 1 B

1 1 1 1 1 1 B

1 1 1 1 1 1 1 1 B



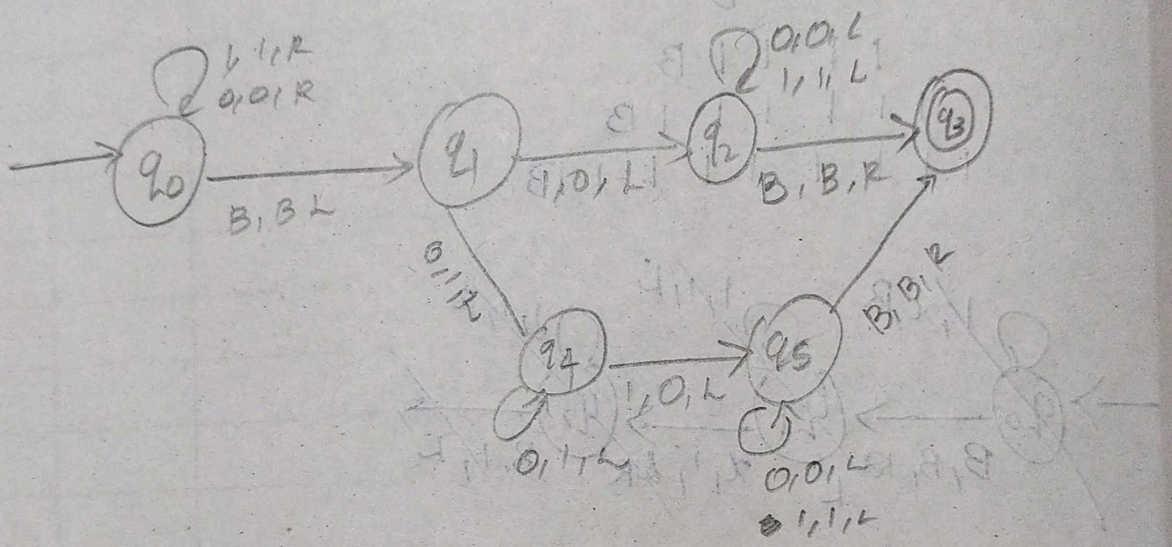


6. Construct a turing machine that decrements a binary no.

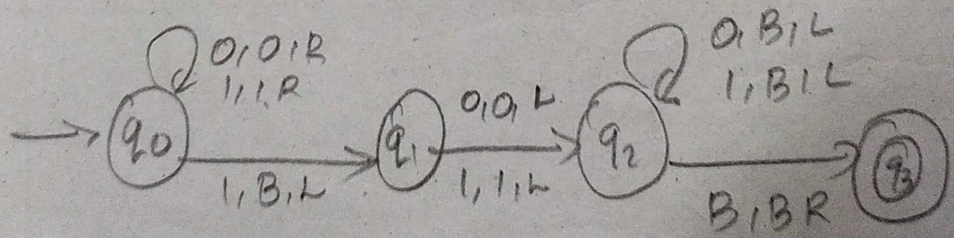
→

10|1   11|0   11|1   01|1   011|1   1000
10|0   10|1   11|0   01|0   011|0   0111

000   00|1   100|1   01|0   010|0
000   00|0   100|0   00|1   001|1



7. Construct a turing machine that computes $n \% 2$.



Universal Turing Machine

Turing machine can be thought of in 2 ways

1) ~~Two~~ Turing machine gives an unprogrammable piece of h/w specialised at solving one particular problem with instructions that are ~~h/w~~ hardwired at the factory.

2) Turing machine is a software. i.e. there is a certain ~~x~~ generic turing machine that can be programmed about the same way as the general purpose computers can., to solve any problem that can be solved by the turing ~~t~~ machine. The program makes the generic machine behave like a specific machine. i.e, turing machine can be thought of as programming language in which we can write programs. Programs written in this

language can be ~~autoc~~ interpreted by some universal machine

Universal turing machine U takes 2 arguments, a description of machine (binary encoded) say m. & input string say "w".

$U(\cancel{m} M, w) = m(w)$

It is the functional notation of universal ~~of~~ TM.

U ~~halts~~ halts on i/p w if & only if TM halts on i/p w.

Consider a TM which is defined as $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

$= (\{q_0, q_f\}, \{a, b\}, \{a, b, B\}, \delta, q_0, B, q_f)$

$\delta(q_0, a) = (q_0, a, R)$

$\delta(q_0, b) = (q_0, b, L)$

$\delta(q_0, B) = (q_F, B, L)$

First we encode all the components of this TM using binary coding.
In binary coding, only symbols 0 & 1 are available. Here 0 is used to quote all the transitions function and 1 is used as the separators of states of TM.

$Q = \{ q_0, q_f \}$ = 0100

$\Sigma = \{ a, b \}$ = 0100

$F = \{ a, b, B \}$ - 01001000

$\delta(q_0, a) = (q_0, a, R)$ - 010101010

$\delta(q_0, b) = (q_0, b, L)$ - 0101010100100

$\delta(q_0, B) = (q_f, B, L)$ - 01000100100010 0

~~(0100~~

Let $w = ab$ be the string to be checked on turing machine M the input to universal turing machine is encoded as,

$\underline{Q}$

0100011

$q_0 = 0$

$q_f = 00$

$\underline{\Sigma}$

$a = 0$

$b = 00$

$B = 000$

$\underline{\Gamma}$

$a = 0$

$b = 00$

$B = 000$

$\delta(q_0, a) = (q_0, a, R)$ - 0101010 10

$\delta(q_0, b) = (q_0, b, L)$ - 0100 100100100

$\delta(q_0, B) = (q_f, B, L)$: 01000100100010100

Let $w = \{a, b\}$ be string to be checked on the TM. M. The i/p to universal TM will be,

(0100110100110100100011010101010101
 01001010010101000100100010011
 01100011100)

Universal turing machine used binary code of turing machine M on string AB & will check if AB is recognised by M.

If true, universal turing machine will halt to say 'Yes'.

If false, UTM will stop to say 'No'.

Non-Deterministic Turing Machine

A non-deterministic turing machine NDTM or NTM differs from deterministic turing machine by having transition function δ such that for a ~~st~~ state q & a tape symbol $\delta(q, x)$ is a set of triples ie $\delta(q, x) = \{(q_1, y, L), (q_2, y, R), (q_3, z, L)\}$

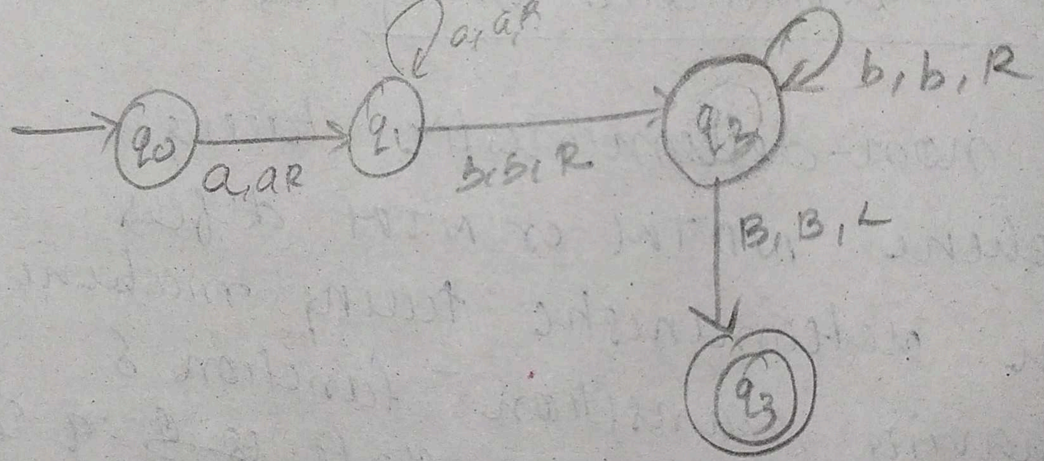
~~ND~~ NTM can choose, at each step, any of the triples to be the next move, the transition ~~funct~~ function for NDTM is given as

$\delta: Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L, R\}}$

A string w in Σ^* is said to be accepted by NDTM if there exists sequence of moves starting from initial id to an accepting ~~st~~ condition.

Q) Construct a NDTM which accepts the language $L = \{a^n b^m / n \geq 1, m \geq 1\}$

→



Enumeration Machine

An enumeration machine is a turing machine with a printer i.e, it has infinite tape & finite state control as of turing machine plus a printer i.e used to generate strings of the language.

Recursive ~~Enumerable~~ Languatage (REL)

A language L is recursively enumerable if it is possible to design a turing ~~for~~ machine for the language L such that for any string ~~w∈L~~ w∈L, turing machine accepts w by entering into final state. and for any string w'∉L turing machine rejects w' by halting in a non final stage. or by looping forever. i.e, ~~if w∈L~~

if w∈L => M accepts by reaching final state. if ~~w~~

if w ~~∉L~~ ∉L => M halts at non final state OR M enters infinite loop.

Recursive Language

A language L is recursive language, if it is ~~design~~ possible to design the turing machine for the language L such that for any string w∈L, turing machine accepts w by entering into final state. For any string w' ∉ L, Turing machine rejects w' by halting in a non final state. ie, If w∈L, M accepts by entering final state. If w∉L, M halts by entering non-final state.

Properties of Recursively Enumerable & Recursive Language

1) Union :-

If L_1 & L_2 are 2 recursive languages, then their union $L_1 \cup L_2$ ~~is also~~ will also be recursive. ~~is~~ ~~beco~~ because TM halts for L_1 & halts for L_2, it will also halt for $L_1 \cup L_2$

2) Concatenation:-

If L_1 & L_2 are 2 recursive languages, then their concatenation ~~L~~. $L_1 L_2$ will also be recursive.

eg: $L_1 = \{a^n b^n ; n \geq 0\}$ is recursive

$L_2 = \{d^m c^m / m \geq 0\}$ "

$L_1 L_2 = \{a^n b^n d^m c^m / n \geq 0, m \geq 0\}$

is also recursive

3) Kleen Closure :-

If L_1 is recursive, its Kleen closure L_1^* will also be recursive.

eg: $L_1 = \{a^n b^n / n \geq 0\}$

$L_1^* = \{a^n b^n / n \geq 0\}^*$

4) Intersection :-

If L_1 & L_2 are 2 recursive languages, then their intersection $L_1 \cap L_2$ will also be recursive.

eg:- $L_1 = \{a^n b^n d^{n+m} / n \geq 0 \text{ \& } m \geq 0\}$

$L_2 = \{a^n b^n d^n / n \geq 0\}$

$L_1 \cap L_2 = \{a^n b^n d^n / n \geq 0\}$

is also recursive.

5) compliment of recursive language is recursive.

6) If L_1 & L_2 are 2 recursively enumerable languages, then

1) $L_1 \cup L_2$ is ~~re~~ REL.

2) $L_1 \cap L_2$ is REL.

3) If L is a REL & its compliment L' is also REL; Then L is recursive language.

7) Compliment of REL L need not be Recursevely enumerable.

Decidability

The problem solved by TM can be catagoriesed into 2 catagories

1) The Problems in which TM can ~~hold~~ halt in the accepting or rejecting ~~s~~tate are called decidable problems. In other words, a problem

is said to be decidable. If it is ~~so~~ solvable.

A language L is decidable, if there exists a turing machine M such that for all strings w if w∈L, M enters q_{accept}. If w∉L, ~~Fo~~ M enters q_{reject}.

The problems in which TM may not halt at all if ~~they~~ they do not accept its input i.e. the problems that do not have ~~a the~~ p. algorithms are called ~~at~~ undecidable problems.

A problem is said to be undecidable if it is unsolvable

decidable or not

* Halting Problem

7) If L is a Recursively Enumerable Language and its complement L' is also Recursively Enumerable, then L is Recursive language.

8) Complement of Recursively Enumerable Language L need not be Recursively Enumerable.

Decidability

The problems solved by Turing Machine can be categorized into two categories:

1) The problems in which TM can halt in the accepting or rejecting state are called decidable problems. In other words, a problem is said to be decidable if it is solvable. A Language L is decidable, if there exists a TM, M such that for all strings w:
   If $w \in L$, M enters q_{accept}.
   If $w \notin L$, M enters q_{reject}.

2) The problems in which TM may not halt at all if they do not accept its input ie the problems that do not have algorithms are called undecidable problems. A problem is said to be undecidable if it is unsolvable.

⇒ A Recursively Enumerable Language is also known as Turing Recognisable (or partially decidable), while a Recursive language is known as Turing decidable. So deciders always terminates, while recognizers can run forever without deciding.

Halting Problem

⇒ Halting problem is the problem of determining, from a description of a machine and an input, whether the program or machine will halt (ie finish running) or continue to run forever.

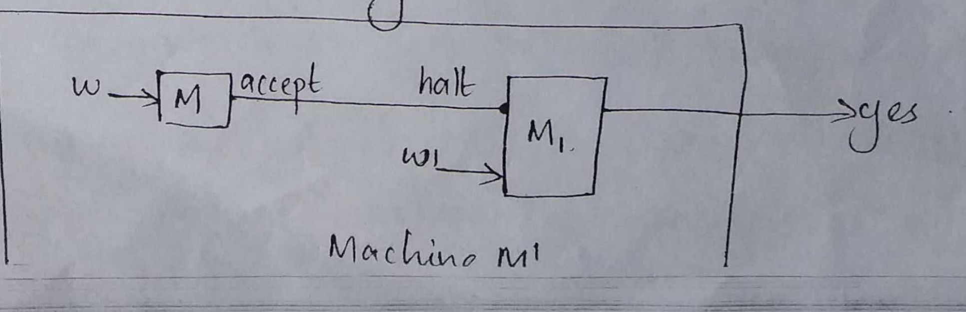
⇒ Halting problem is an undecidable problem.

⇒ Alan Turing proved in 1936 that a general algorithm to solve the halting problem for all possible program-input pairs cannot exist.

Theorem : Halting problem of a Turing Machine is undecidable.

Proof: To show this, we reduce problem of halting to a problem of acceptance.

Consider the following TM.



Here we take an instance (M, w) and construct another instance (M_1, w_1)

It is taken such that M_1 halts on input w_1, if and only if M accepts w.

The machine M' stops when M_1 halts.

Initially, the string w is fed to the TM, M and w_1 is fed to the TM, M_1.

If M accepts w, then it sends a halt signal to M_1. Then TM, M_1 halts on input w_1.

If M rejects w, then M_1 does not halt on w_1.

Thus halting of M_1 depends on the acceptance behaviour of M. Acceptance behaviour of TM is undecidable, Hence halting of M_1 or M' is undecidable.

Chomsky Hierarchy

A number of language families are present.

Noam Chomsky, a founder of Formal Language Theory provided an initial classification into 4 Language types: Type 0, Type 1, Type 2, Type 3

→ Four types of languages and their associated grammars are defined in Chomsky Hierarchy.

The Languages are:

Type 0 Languages or Unrestricted Languages.

Type 1 Languages or Context Sensitive Languages.

CLOSURE PROPERTIES OF RECURSIVE & RECURSIVELY ENUMERABLE LANGUAGE

PROPERTY 1

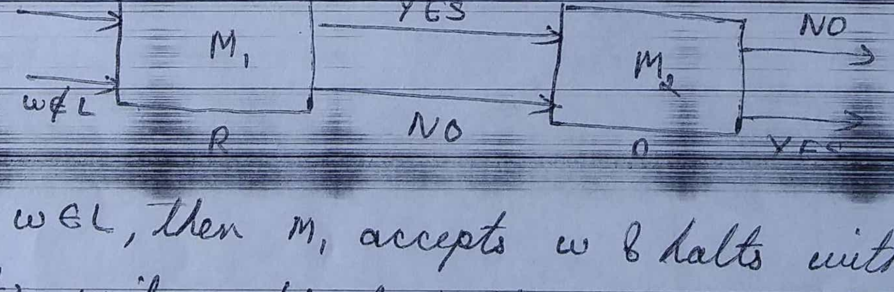
The complement of a recursive language is also recursive, ie, recursive languages are closed under complementation.

Proof.

Let 'L' be a recursive language accepted by the Turing m/c M_1 where 'YES' is an accepting state & 'NO' is a non-accepting state.

Let '$\bar{L}$' be a recursive language accepted by the TM M_2 where 'NO' is an accepting state & 'YES' is a non-accepting state.

The construction of M_1 & M_2 are given as follows,



Let $w \in L$, then M_1 accepts w & halts with 'YES'. M_1 rejects w if $w \notin L$ & halts with 'NO'. M_2 is activated once M_1 halts. M_2 works on $\bar{L}$ & hence if M_1 returns 'YES', M_2 halts with 'NO'. If M_1 returns 'NO', M_2 halts with 'YES'.

Thus for all w, where $w \in L$ or $w \notin L$, M_2 halts with either 'Yes' or 'No'. Hence the complement of a recursive lang is

Property 2

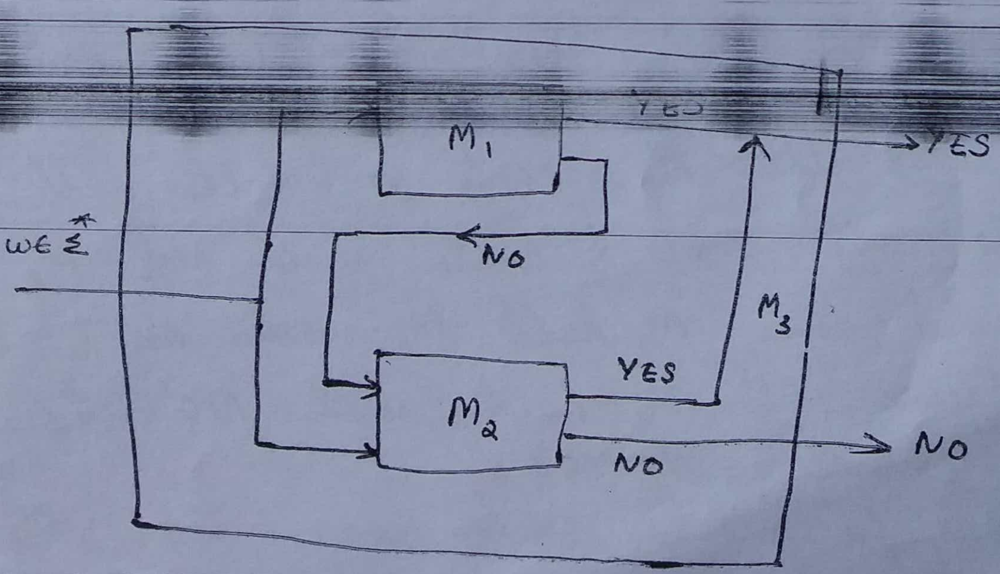
The union of two recursive language is also recursive, ie, recursive languages are closed under union.

Proof

Let L_1 & L_2 are two recursive languages (ie $L_1, L_2 \in R$) then there exist two TMs M_1 & M_2 that compute L_1 & L_2 respectively, because recursive languages are decided by TM.

For deciding $L_1 \cup L_2$, construct a TM M_3 which can simulate M_1 & M_2 parallel to each other one step at a time. It first stimulates M_1, that is starts M_1 on w first. If it decides on the given i/p w, ie M_1 outputs YES, then M_3 also o/p's YES & terminates.

On the other hand, if M_1 o/p's NO, M_3 starts M_2 on i/p 'w' & o/p's whatever M_2 o/p's. It is clear that M_3 decides $L_1 \cup L_2$ as shown in fig. below



Since both M_1 & M_2 are algorithms, M_3 is guaranteed to halt. This simulation causes M_3 to decide if and only if at least one of the two m/c's M_1 & M_2 decide.

Thus we can conclude that M_3 decides or computes $L_1 \cup L_2$, ie $L_1, L_2 \in R$.

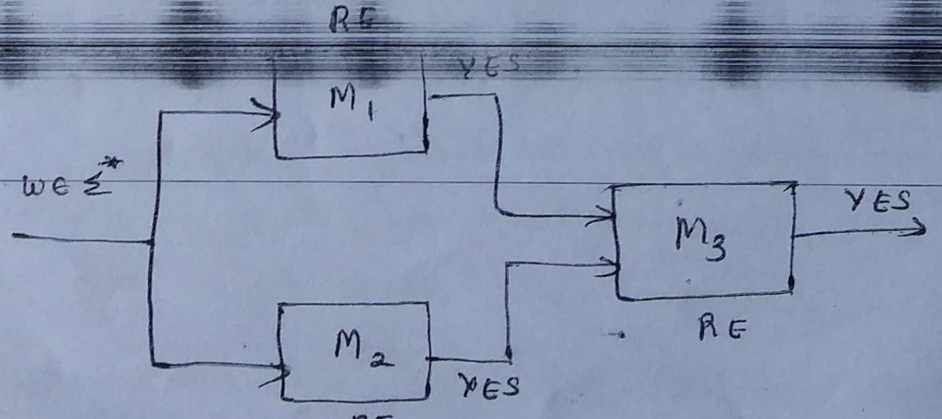
→ <u>Property 3</u>

The union of two recursively enumerable language is also recursively enumerable, ie RE lang's are closed under union.

<u>Proof</u>

Suppose L_1 & L_2 are two recursively enumerable lang's (ie $L_1, L_2 \in RE$) then there exist two TM's M_1 & M_2 corresponding to L_1 & L_2 respectively, because R.E lang's are accepted by TM.

∴, M_1 will accept any string w if $w \in L_1$ & similarly M_2 will accept w, if $w \in L_2$.



Now, we can form the m/c M_3 which can simulate M_1 & M_2 parallel to each other one step at a time. If either accepts, then M_3 accept & recognizes $L_1 \cup L_2$, as in the fig.

The m/c can choose whether to simulate M_1 or M_2 first so it is a NDTM, however it will not make any difference in computation because NDTM & DTM's have the same expressive power.

The possible outcomes are,

1) If M_1 & M_2 both reject, the string simultaneously only then M_3 rejects.
2) If M_1 accepts the string, then M_3 accepts
3) If M_2 accept the string, then M_3 accepts.

This simulation causes M_3 to accept iff at least one of the two m/c's M_1 & M_2 accepts. So we can conclude that M_3 recognizes $L_1 \cup L_2$, ie $L_1 L_2 \in RE$

→ Property 4

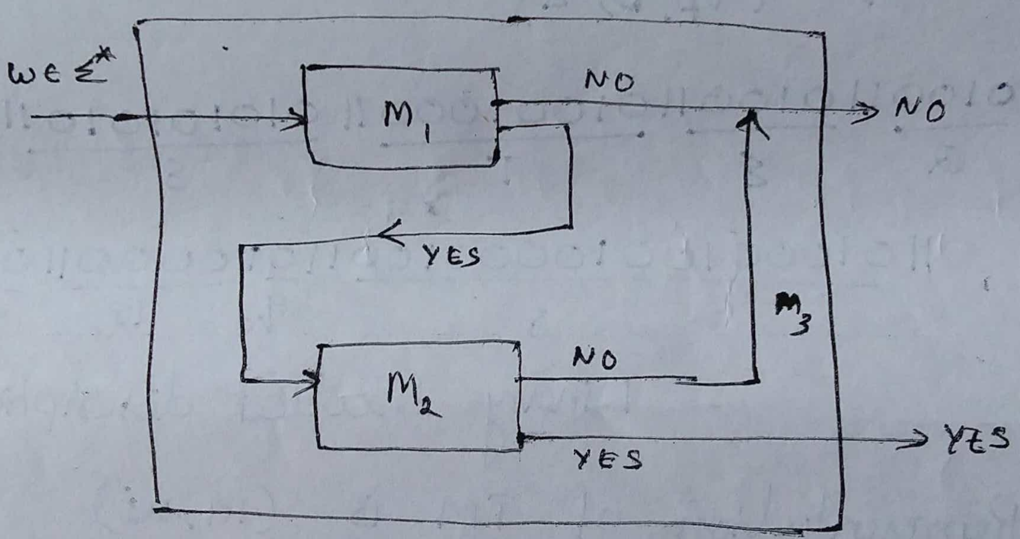
The intersection of two recursive language is also recursive. ie recursive languages are closed under intersection.

Proof

Let L_1 & L_2 are two recursive languages (ie $L_1, L_2 \in R$) then there exist two TM's M_1 & M_2 that compute L_1 & L_2 respectively, because recursive languages are decided by TM.

$$L(M_1) = L_1 \quad \& \quad L(M_2) = L_2$$

Let M_3 be the TM, ie constructed by the intersection of M_1 & M_2. M_3 is constructed as follows,



The TM M_3 simulates M_1 with i/p string w. If $w \notin L_1$, then M_1 halts along with M_3 with ~~answer 'NO' since since~~ $\cancel{L(M_3) = L(M_1) + L(M_2)}$ ~~. If then~~ $\cancel{M_1}$ ~~accepts with the answer 'YES' &~~ $\cancel{M_3}$ ~~simulates~~ $\cancel{M_2}$

If M_2 accepts the string, then the answer of M_2 & M_3 are 'YES' & halts. Else M_2 & M_3 halts with answer 'NO.'

Thus the intersection of two recursive lang. is recursive.