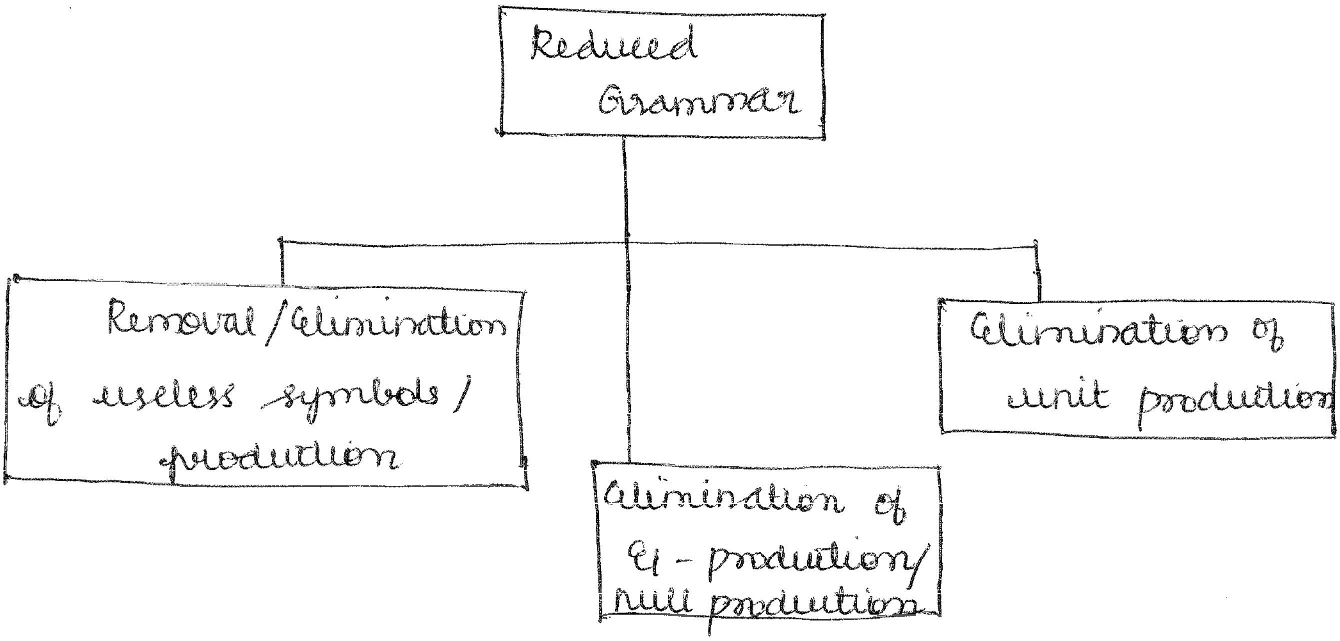
UNIT IV

PROPERTIES OF CONTEXT FREE LANGUAGES

SIMPLICATION OF CFG:

Simplification of CFG means reduction of grammar by removing useless symbols, thus reducing the length of grammar.

The properties of reduced grammar are,



Eliminating Useless symbols / production:

Let $G = (V, T, P, S)$ be a grammar. A symbol 'x' is useful if there is a derivation, $S \overset{*}{\Rightarrow} \alpha x \beta \overset{*}{\Rightarrow} w$ for some α, β and w where w is in T^*, Otherwise it is useless.

There are two ways to find useful production.

(1) Some terminal string must be derived from 'x'.

(2) x must occur in some string derived from S.

Two terms are involved,

(1) Generating symbols (2) Reachable symbols.

Generating symbol: If 'x' is generating if $X \xRightarrow{*} w$ for some T^* in W,

Steps:

(1) every symbol of T is generating, therefore it generate itself

(2) If A tends to a $(A \to a)$, then A is also generating for $\boxed{a \in T \text{ or } a \in \epsilon}$

Reachable symbol: 'x' is reaching if there is a derivation, $\boxed{S \xRightarrow{*} \alpha \, x \, \beta}$ for some α & β.

Steps:

(1) S is a reachable because S is a start symbol.

(2) If A is reachable, then all production with A in the head, all symbols of those production are also reachable.

PROBLEMS:

(1) $S \to AB \mid a$, $A \to BC \mid b$, $B \to aB \mid C$, $C \to aC \mid B$

Solution:

(1) Identify all generating variable:

(2) Generating symbols are $\{a, b, c, S, A\}$

(3) Useless symbol is B, C & eliminate B, C

$S \to a$

$A \to b$

(4) Removing unreachable production / useless.

unreachable production: $A \to b$

(5) Ans : Useful production : $S \to a$.

(2) $S \to aS \mid A \mid C$

$A \to a$

$B \to aa$

$C \to aCb$

Solution : (1) Generating Symbols : $\{a, b, A, B, S\}$

(2) Useless symbol : C

$\therefore$ $S \to aS \mid A$

$A \to a$

$B \to aa$

(3) Unreachable symbol / production : B

Ans : $\boxed{\begin{array}{l} S \to aS \\ A \to a \end{array}}$ → Useful production.

(3) $S \to aA \mid a \mid Bb \mid cC$

$A \to aB$

$B \to a \mid Aa$

$C \to cCD$

$D \to ddd$

Solution : (1) Generating symbols : $\{a, b, c, d, S, A, B, D\}$

(2) Useless symbol : C

$\therefore$ $S \to aA \mid a \mid Bb$

$A \to aB$

$B \to a \mid Aa$

$D \to ddd$

(3) Unreachable symbol / production : D

Ans : $\boxed{\begin{array}{l} S \to aA \mid a \mid Bb \\ A \to aB \\ B \to a \mid Aa \end{array}}$ → Useful production

(4) $S \to aA \mid bB$

$A \to aA \mid a$

$B \to bB$

$D \to ab \mid Ea$

$E \to ac \mid d$

Solution:

(1) Generating symbol : $\{a, b, c, d, S, A, E, D\}$

(2) Useless symbol : B.

$$\therefore \quad S \to aA$$
$$A \to aA \mid a$$
$$D \to ab \mid Ea$$
$$E \to ac \mid d$$

(3) Unreachable symbol : D, E

$$\therefore \text{Ans} = \boxed{\begin{array}{l} S \to aA \\ A \to aA \mid a \end{array}} \to \text{Useful production.}$$

Eliminating ϵ / Null production:

A production which is of the form $A \to \epsilon$ is called ϵ- production. If ϵ is in $L(G)$, it is not possible to eliminate all ϵ- production. The same is possible if ϵ is not in $L(G)$.

For each variable A, if $A \overset{*}{\Rightarrow} \epsilon$, then A is called as nullable variable.

We need to check whether the variable is nullable or not.

If $B \to C_1 . C_2 . C_3 \dots C_n$ where each C_i is nullable, then B is nullable.

PROBLEMS:

(1) $S \to aSa \mid bAb$

$A \to \epsilon$

Solution:

(i) $V = \{S, A\}$

(ii) Null production : $A \to \epsilon$

(iii) Nullable variable : $\{A\}$

(iv) eliminate wherever A is there which should not affect corresponding grammar.

If we remove A, bAb

$\Rightarrow b\epsilon b$

$\Rightarrow bb$

$\therefore S \to aSa \mid b\cancel{A}b \mid bb$

$A \to \cancel{\epsilon}$ (bAb can be eliminated because A is useless symbol)

$\boxed{\therefore S \to aSa \mid bb}$

(2) $S \to AB$

$A \to aAA \mid \epsilon$

$B \to bBB \mid \epsilon$

Solution:

(i) $V = \{S, A, B\}$

(ii) Null production : $\{A \to \epsilon, B \to \epsilon\}$

(iii) Nullable variable : $\{A, B, S\}$

(iv) Find production with & without nullable variable

$S \to AB \mid A \mid B \mid \cancel{\epsilon}$ ($\because AB \to A\epsilon$, A)

$A \to aAA \mid aA \mid \overset{\text{dup}}{\cancel{aA}} \mid a \mid \cancel{\epsilon}$

$B \rightarrow bBB \mid bB \mid \cancel{bB} \mid b \mid \cancel{\epsilon}$ duplicate copy

$$\therefore \boxed{\begin{array}{l} S \rightarrow AB \mid A \mid B \\ A \rightarrow aAA \mid aA \mid a \\ B \rightarrow bBB \mid bB \mid b \end{array}}$$

[Eliminate null & duplicate values]

(3) $A \rightarrow 0B1 \mid 1B1$

$B \rightarrow 0B \mid 1B \mid \epsilon$

Solution :

(1) $V = \{A, B\}$

(2) Null production : $B \rightarrow \epsilon$

(3) Nullable variable : $\{B\}$

(4) find production with & without nullable variables

$A \rightarrow 0B1 \mid 01 \mid 1B1 \mid 11$

$B \rightarrow 0B \mid 0 \mid 1B \mid 1 \mid \cancel{\epsilon}$

$$\therefore \boxed{\begin{array}{l} A \rightarrow 0B1 \mid 01 \mid 1B1 \mid 11 \\ B \rightarrow 0B \mid 0 \mid 1B \mid 1 \end{array}}$$

(4) $S \rightarrow a \mid Ab \mid aBa$

$A \rightarrow b \mid \epsilon$

$B \rightarrow b \mid A$

Solution :

(1) Variable : $\{S, A, B\}$

(2) Null production : $A \rightarrow \epsilon$

(3) Nullable variable : $\{A, B\}$

(4) find production,

$S \to a \mid Ab \mid b \mid aBa \mid aa$

$A \to b \mid \cancel{\epsilon}$

$B \to b \mid A \mid \cancel{\epsilon}$

$\therefore$ $\boxed{\begin{array}{l} S \to a \mid Ab \mid b \mid aBa \mid aa \\ A \to b \\ B \to b \mid A \end{array}}$

Elimination of unit production:

A unit production is a production which is of the form $A \to B$ where both A & B are variables.

UNIT PAIR: If the sequence of derivation steps are $A \Rightarrow B_1 \Rightarrow B_2 \ldots B_n \Rightarrow \alpha$, then these unit productions are replaced by a non-unit production, $B_n \to \alpha$ directly from A.

$\therefore \; A \to \alpha$

(A, B) such that $A \overset{*}{\Rightarrow} B$ is called an unit pair.

How to eliminate unit production:

Given a CFG, $G = (V, T, P, S)$ with unit production, then construct a new CFG $G_1 = (V, T, P_1, S)$

(1) Find all the unit pair of G.

(2) For each unit pair (A, B) if there is a production $A \to B$ replace it with $A \to \alpha$ provided $B \to \alpha$ is a production in G.

PROBLEMS :

(1) $S \to Aa \mid B$

$B \to A \mid bb$

$A \to a \mid bc \mid B$

Solution :

(i) Find all unit production

$S \to B$

$B \to A$

$A \to B$

(ii) $S \to B$

$\to A$

$\to a \mid bc$

$S \to B$

$\to bb$

$B \to A$

$\to a \mid bc$

$B \to bb$

$A \to B$

$\to A \mid bb$

$\to a \mid bc \mid bb$

$$\therefore \boxed{\begin{aligned} S &\to Aa \mid a \mid bc \mid bb \\ B &\to a \mid bc \mid bb \\ A &\to a \mid bc \mid bb \end{aligned}}$$

(2) $S \to 0A \mid 1B \mid C$

$A \to 0S \mid 00$

$B \to 1 \mid A$

$C \to 01$

Solution :

(i) Find all unit production

$S \to C$

$B \to A$

(ii) $S \rightarrow C$
$\rightarrow 01$

$B \rightarrow A$
$\rightarrow 0S \mid 00$

$\therefore$ $\boxed{\begin{array}{l} S \rightarrow 0A \mid 1B \mid 01 \\ A \rightarrow 0S \mid 00 \\ B \rightarrow 1 \mid 0S \mid 00 \\ C \rightarrow 01 \end{array}}$ $\Rightarrow$ Remove unreachable production

C is unreachable

$\therefore$ **Ans:** $\boxed{\begin{array}{l} S \rightarrow 0A \mid 1B \mid 01 \\ A \rightarrow 0S \mid 00 \\ B \rightarrow 1 \mid 0S \mid 00 \end{array}}$

(3) $S \rightarrow AB$
$A \rightarrow a$
$B \rightarrow C \mid b$
$C \rightarrow D$
$D \rightarrow E \mid bC$
$E \rightarrow d \mid Ab$

Solution:

(i) Find all unit production

$B \rightarrow C$
$C \rightarrow D$
$D \rightarrow E$

(ii) $B \rightarrow C$
$\rightarrow d \mid Ab \mid bC \mid b$

$D \rightarrow E$
$\rightarrow d \mid Ab$

$C \rightarrow D$
$C \rightarrow d \mid Ab \mid bC$

$\therefore$ $\boxed{\begin{array}{l} S \rightarrow AB \\ A \rightarrow a \\ B \rightarrow d \mid Ab \mid bC \mid b \\ C \rightarrow d \mid Ab \mid bC \\ D \rightarrow d \mid Ab \mid bC \\ E \rightarrow d \mid Ab \end{array}}$

(iii) Remove unreachable production:

D and E

Ans : ∴ $S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow d \mid Ab \mid AC \mid b$

$C \rightarrow d \mid Ab \mid bC$

NORMAL FORM OF CFG :

(1) Chomsky Normal form (CNF)

(2) Greibach Normal Form (GNF)

CONVERSION FROM CFG INTO CNF :

Any CFL without ϵ is generated by a grammar in which all productions are of the form $A \rightarrow BC$ (or) $A \rightarrow a$ where A, B, C are variables and a is a terminal.

Steps :

(1) Write down the rule of CFG

$$\boxed{\begin{array}{rcl} \text{Non-Terminal (NT)} & \rightarrow & \text{NT NT} \\ \text{NT} & \rightarrow & \text{Terminal} \end{array}}$$

(2) Write the given production.

(3) Simplify the CFG

(3.1) elimination of ϵ-production

(3.2) elimination of unit production

(3.3) elimination of useless production.

(4) Convert CFG into CNF

(5) Write down the resultant productions.

PROBLEMS :

(1) construct the grammar $(\{S, A, B\}, \{a, b\}, P, S)$ has the production $S \rightarrow bA \mid aB$ Convert into CNF.

$A \rightarrow bAA \mid aS \mid a$

$B \rightarrow aBB \mid bS \mid b$

Solution :

(1) Rule of CNF :

$$\boxed{\begin{array}{l} NT \rightarrow NT\ NT \\ NT \rightarrow T \end{array}}$$

(2) write the given production

$S \rightarrow bA \mid aB$

$A \rightarrow bAA \mid aS \mid a$

$B \rightarrow aBB \mid bS \mid b$

(3) Simplify the CFG.

(3.1) Eliminate ε- production :

There is no ε-production in the given Grammar. Then CFG is,

$S \rightarrow bA \mid aB$

$A \rightarrow bAA \mid aS \mid a$

$B \rightarrow aBB \mid bS \mid b$

(3.2) Eliminate unit production :

There is no unit production in the given Grammar. Then CFG is,

$S \rightarrow bA \mid aB$

$A \rightarrow bAA \mid aS \mid a$

$B \rightarrow aBB \mid bS \mid b$

(3.3) Elimination of useless production :

There is no useless production in the given Grammar. Then CFG is,

$S \to bA \mid aB$

$A \to bAA \mid aS \mid a$

$B \to aBB \mid bS \mid b$

(4) Simplify CFG to CNF :

$S \to bA$ (Rule 1)

$S \to aB$ (Rule 2)

$A \to bAA$ (Rule 3)

$A \to aS$ (Rule 4)

$A \to a$ (Rule 5) // CNF formate

$B \to aBB$ (Rule 6)

$B \to bS$ (Rule 7)

$B \to b$ (Rule 8) // CNF format

Rule 1 :

$S \to \underline{b}A$

$$\boxed{\begin{array}{l} S \to C_b A \\ C_b \to b \end{array}}$$

Rule 2 :

$S \to \underline{a}B$

$$\boxed{\begin{array}{l} S \to C_a B \\ C_a \to a \end{array}}$$

Rule 3 :

$A \to \underline{b}AA$

$\to C_b \underline{AA}$

$$\boxed{\begin{array}{l} A \to C_b D_1 \\ C_b \to b \\ D_1 \to AA \end{array}}$$

Rule 4 :

$A \to \underline{aS}$

$$\boxed{A \to C_a S}$$

Rule 6 :

$B \to \underline{a}BB$

$\to C_a \underline{BB}$

$$\boxed{\begin{array}{l} B \to C_a D_2 \\ D_2 \to BB \end{array}}$$

Rule 7 :

$B \to \underline{b}S$

$$\boxed{B \to C_b S}$$

(5) Resultant productions are,

$$\boxed{\begin{aligned} S &\rightarrow C_b A \mid C_a B \\ A &\rightarrow C_b D_1 \mid C_a S \mid a \\ B &\rightarrow C_a D_2 \mid C_b S \mid b \\ D_1 &\rightarrow AA \\ D_2 &\rightarrow BB \\ C_a &\rightarrow a \\ C_b &\rightarrow b \end{aligned}}$$

---

(2) $S \rightarrow ASB \mid \epsilon$

$A \rightarrow aAS \mid a$. Convert into CNF.

$B \rightarrow SbS \mid A \mid bb$

Solution :

(1) Rule of CNF :

$$\boxed{\begin{aligned} NT &\rightarrow NT\ NT \\ NT &\rightarrow T \end{aligned}}$$

(2) Given production :

$$S \rightarrow ASB \mid \epsilon$$
$$A \rightarrow aAS \mid a$$
$$B \rightarrow SbS \mid A \mid bb.$$

(3) Simplify CFG :

(3.1) Elimination of ϵ - production :

$$V = \{S, A, B\}$$

Null production : $S \rightarrow \epsilon$

Nullable variable : $\{S\}$

$S \rightarrow ASB \mid AB \mid \cancel{\epsilon}$

$A \rightarrow aAS \mid aA \mid a$

$B \rightarrow Sbs \mid bs \mid Sb \mid b \mid A \mid bb$

$\therefore$
$$\boxed{\begin{aligned} S &\rightarrow ASB \mid AB \\ A &\rightarrow aAS \mid aA \mid a \\ B &\rightarrow Sbs \mid bs \mid Sb \mid b \mid A \mid bb \end{aligned}}$$

(3.2) Eliminate unit production:

- Find all unit production.

  $B \rightarrow A$

- $B \rightarrow A \rightarrow aAS$ $B \rightarrow A \rightarrow a$ $B \rightarrow A \rightarrow aA$

$\therefore$ There is no unreachable production

$\therefore$
$$\boxed{\begin{aligned} S &\rightarrow ASB \mid AB \\ A &\rightarrow aAS \mid aA \mid a \\ B &\rightarrow Sbs \mid bs \mid Sb \mid b \mid aAS \mid aA \mid a \mid bb \end{aligned}}$$

(3.3) Eliminate useless production:

There is no useless production in the given Grammar G.

Then CFG,
$$\boxed{\begin{aligned} S &\rightarrow ASB \mid AB \\ A &\rightarrow aAS \mid aA \mid a \\ B &\rightarrow Sbs \mid bs \mid Sb \mid b \mid aAS \mid aA \mid a \mid bb \end{aligned}}$$

(4) Simplify CFG to CNF:

- $S \rightarrow ASB$

$\boxed{\begin{array}{l} S \rightarrow AD_1 \\ D_1 \rightarrow SB \end{array}}$

$S \rightarrow AB$

$\boxed{S \rightarrow AB}$

- $A \rightarrow aAS$

$\rightarrow C_a AS$

$\boxed{\begin{array}{l} A \rightarrow C_a D_2 \\ C_a \rightarrow a \\ D_2 \rightarrow AS \end{array}}$

$A \rightarrow aA$

$\boxed{A \rightarrow C_a A}$

$A \rightarrow a$

- $B \rightarrow SbS$

$\rightarrow S C_b S$

$\boxed{\begin{array}{l} B \rightarrow SD_3 \\ D_3 \rightarrow C_b S \\ C_b \rightarrow b \end{array}}$

$B \rightarrow Sb$

$\boxed{B \rightarrow S C_b}$

$B \rightarrow bS$

$\boxed{B \rightarrow C_b S}$

$\boxed{B \rightarrow b}$

$B \rightarrow aAS$

$\rightarrow C_a AS$

$\boxed{B \rightarrow C_a D_2}$

$B \rightarrow aA$

$\boxed{B \rightarrow C_a A}$

$\boxed{B \rightarrow a}$

$B \rightarrow bb$

$\rightarrow C_b b$

$\boxed{B \rightarrow C_b C_b}$

(5) Final productions are,

$$\boxed{\begin{array}{l} S \rightarrow AD_1 \mid AB \\ A \rightarrow C_a D_2 \mid C_a A \mid a \\ B \rightarrow SD_3 \mid SC_b \mid C_b S \mid b \mid C_a D_2 \mid C_a A \mid a \mid C_b C_b \\ D_1 \rightarrow SB \\ D_2 \rightarrow AS \\ D_3 \rightarrow C_b S \\ C_a \rightarrow a \\ C_b \rightarrow b \end{array}}$$

GREIBACK NORMAL FORM :

A CFG is in GNF if the productions are in the following form,

$$\boxed{\begin{array}{l} A \rightarrow b \quad \text{(or)} \\ A \rightarrow b\, C_1\, C_2 \cdots\cdots C_n \end{array}}$$

where $A, C_1, C_2 \cdots C_n$ are variables and b is a terminal.

Note :

Rule : $\boxed{\begin{array}{l} NT \rightarrow T \\ NT \rightarrow T\ NT \cdots\cdots NT \end{array}}$

CONVERSION FROM CFG INTO GNF :

Steps :

(1) Simplify CFG (eliminating ϵ - production, unit production, useless production)

(2) Check whether the simplified CFG in CNF format or not. If not convert it into CNF.

(3) Change the names of the non-terminal symbols into some A_i in ascending order of i.

(4) Alter the rules so that, non-terminal symbols are in ascending order such that if the production is of the form $A_i \rightarrow A_j \alpha$ then $i < j$ & should never be $i \geq j$.

(5) Remove left recursion production $\boxed{A_i \rightarrow A_i \alpha}$

Rules : By Introducing new variable, $B_i \rightarrow \alpha B_i \mid \alpha$

(6) Check whether the production is in GNF format or not. If it is not, then convert it into GNF.

(7) Write the final set of production in given CFG order

PROBLEM:

(1) $S \to CA \mid BB$

$B \to b \mid SB$

$C \to b$

$A \to a$

Solution:

(1.1) Eliminate ϵ - production:

There is no ϵ - production in given Grammar.

(1.2) Eliminate unit production:

No unit production

(1.3) Eliminate useless production:

No useless production.

(2) Check whether simplified CFG is CNF format or not.

All are in CNF.

$$\therefore \left.\begin{array}{l} S \to CA \mid BB \\ B \to b \mid SB \\ C \to b \\ A \to a \end{array}\right\} \text{All production are in CNF format}$$

(3) Change the name of NT (Non-terminal) symbols.

Replace S by A_1

C by A_2

A by A_3

B by A_4.

we get,

$A_1 \to A_2 A_3 \,|\, A_4 A_4$

$A_4 \to b \,|\, A_1 A_4$

$A_2 \to b$

$A_3 \to a$

(4) Alter the rules so that, NT symbols are in ascending order. $[A_i \to A_j \alpha]$

$A_1 \to A_2 A_3$

$i = 1,\ j = 2 \qquad i < j$

$A_1 \to A_4 A_4$

$i = 1,\ j = 4 \qquad i < j$

$A_4 \to b$ // GNF format

$A_4 \to A_1 A_4 \to ①$

$i = 4,\ j = 1 \qquad i > j$

Sub $A_1 = A_2 A_3 / A_4 A_4$ in ①

$\therefore A_4 \to A_2 A_3 A_4 \,|\, A_4 A_4 A_4 \to ②$

$i = 4,\ j = 2 \qquad i = 4,\ j = 4$

$i > j \qquad i \geq j$

Sub $A_2 = b$ in ②

$A_4 \to b A_3 A_4 \,|\, A_4 A_4 A_4 \to ③$

(5) Here $i = j$ & then remove left recursion

$A_4 \to b A_3 A_4$ (T NT NT) // GNF format

$\overset{A}{A_4} \to \overset{A}{A_4}\, \overset{\alpha}{A_4 A_4}$

since $\boxed{\begin{array}{l} A \to \alpha \\ B \to \alpha B \,|\, \alpha \end{array}}$

Introducing B_4 as new variable

$$B_4 \rightarrow A_4 A_4 B_4 \mid A_4 A_4 .$$

$A_4 \rightarrow b \mid bA_3A_4 \mid bB_4 \mid bA_3A_4B_4$ // GNF format

$\therefore$

$$\boxed{\begin{array}{l} A_1 \rightarrow \underline{A_2} A_3 \mid \underline{A_4} A_4 \\ A_4 \rightarrow b \mid bB_4 \mid bA_3A_4 \mid bA_3A_4B_4 \\ A_2 \rightarrow b \\ A_3 \rightarrow a \end{array}}$$

(6) Check whether given production are in GNF or not

$$\boxed{\begin{array}{l} A_1 \rightarrow bA_3 \mid bA_4 \mid bB_4A_4 \mid bA_3A_4B_4 \mid bA_3A_4B_4A_4 \\ A_4 \rightarrow b \mid bB_4 \mid bA_3A_4 \mid bA_3A_4B_4 \\ A_2 \rightarrow b \\ A_3 \rightarrow a \\ B_4 \rightarrow A_4A_4B_4 \mid A_4A_4 \end{array}}$$

② $S \rightarrow AB$

$A \rightarrow BS \mid b$. convert into GNF .

$B \rightarrow SA \mid a$

<u>Solution</u> :

(1.1) <u>eliminate ϵ - production</u> : No ϵ - production

(1.2) <u>eliminate unit production</u> : No unit production

(1.3) <u>eliminate useless production</u> : No useless production

(2) Check whether the simplified CFG in CNF format or not,

$S \rightarrow AB$
$A \rightarrow BS \mid b$
$B \rightarrow SA \mid a$

All are in CNF.

(3) Change the names of NT symbols into some A_i in ascending order of i.

Replace S by A_1
A by A_2
B by A_3

we get:

$A_1 \rightarrow A_2 A_3$
$A_2 \rightarrow A_3 A_1 \mid b$
$A_3 \rightarrow A_1 A_2 \mid a$

(4) Alter the rules so that, NT symbols are in ascending order. $A_i \rightarrow A_j \alpha$

$A_1 \rightarrow A_2 A_3$
$i=1,\ j=2 \qquad i<j$

$A_2 \rightarrow A_3 A_1 \mid b$
$i=2,\ j=3 \qquad i<j$

$A_3 \rightarrow A_1 A_2 \mid a \rightarrow ①$
$i=3,\ j=1 \qquad i>j$

Sub $A_1 = A_2 A_3$ in ①

$\therefore A_3 \rightarrow \underline{A_2} A_3 A_2 \mid a \rightarrow ②$
$j=2,\ i=3 \qquad i>j$

Sub $A_2 = A_3 A_1 \mid b$ in ②

$A_3 \rightarrow A_3 A_1 A_3 A_2 \mid \overset{T}{b} \overset{NT}{A_3} \overset{NT}{A_2} \mid \overset{T}{a}$
$i=3,\ j=3 \qquad i=j$

(5) Here i = j & then remove left recursion

$A_3 \rightarrow bA_3A_2 \mid a$ // GNF format

$A_3 \rightarrow \overset{A}{A_3} \mid \overset{\alpha}{A_1 A_3 A_2}$

Since $\boxed{\begin{array}{l} A \rightarrow \alpha \\ B \rightarrow \alpha B \mid \alpha \end{array}}$ Introduce B_3 new variable

$\therefore B_3 \rightarrow A_1A_3A_2B_3 \mid A_1A_3A_2$

$A_3 \rightarrow bA_3A_2 \mid a \mid bA_3A_2B_3 \mid aB_3$

$\therefore A_1 \rightarrow A_2A_3$

$A_2 \rightarrow A_3A_1 \mid b$

$A_3 \rightarrow bA_3A_2 \mid a \mid bA_3A_2B_3 \mid aB_3$

(6) Check whether given production are in GNF or not.

$A_1 \rightarrow \underline{A_2}A_3$

$\rightarrow \underline{A_3}A_1A_3 \mid b$

$A_1 \rightarrow bA_3A_2A_1A_3 \mid aA_1A_3 \mid bA_3A_2B_3A_1A_3 \mid aB_3A_1A_3 \mid b$

$A_2 \rightarrow \underline{A_3}A_1 \mid b$

$\rightarrow bA_3A_2A_1 \mid aA_1 \mid bA_3A_2B_3A_1 \mid aB_3A_1 \mid b$

$A_3 \rightarrow bA_3A_2 \mid a \mid bA_3A_2B_3 \mid aB_3$

$\therefore$

$A_1 \rightarrow bA_3A_2A_1A_3 \mid aA_1A_3 \mid bA_3A_2B_3A_1A_3 \mid aB_3A_1A_3 \mid b$

$A_2 \rightarrow bA_3A_2A_1 \mid aA_1 \mid bA_3A_2B_3A_1 \mid aB_3A_1 \mid b$

$A_3 \rightarrow bA_3A_2 \mid a \mid bA_3A_2B_3 \mid aB_3$

$B_3 \rightarrow bA_3A_2A_1A_3A_2B_3 / aA_1A_3A_3A_2B_3 / bA_3A_2B_3A_1A_3A_3A_2B_3 /$
$aB_3A_1A_3A_3A_2B_3 / bA_3A_2B_3 / bA_3A_2A_1A_3A_3A_2 / aA_1A_3A_3A_2 /$
$bA_3A_2B_3A_1A_3A_3A_2 / aB_3A_1A_3A_3A_2 / bA_3A_2$

UNIT - IV TURING MACHINE :

Definitions of Turing Machine - Models - Computable Languages & Function - Techniques For Turing Machine - Construction - Multihead & Multi Tape Turing Machine - The Halting problem - Partial solvability - Problems about Turing Machine - Chomskian Hierarchy of Languages.

INTRODUCTION - TURING MACHINE (TM):

- During the year 1936, Alan Turing introduced a new mathematical model called Turing Machine
- Turing Machine is an abstract machine (or) mathematical model to represent a real computer.
- Turing Machine is a tool, for studying the computability of mathematical function.
- Turing Hypothesis believed that a function is computable if and only if it can be computed by turing machine.
- Turing machine can solve any problem that a modern computer can solve.
- Turing machine is used to define the language and to compile the integer functions.
- Turing machine accepts recursive language or recursive enumerable language.
- Turing machine differs from PDA and FA.
- FA has finite memory and PDA has infinite memory and access in LIFO order
- But TM has both infinite memory and no restriction in accessing the input.

• TM has infinite tape memory & the tape head can move either left or right to access the input

MODEL OF TURING MACHINE:

Turing Machine has

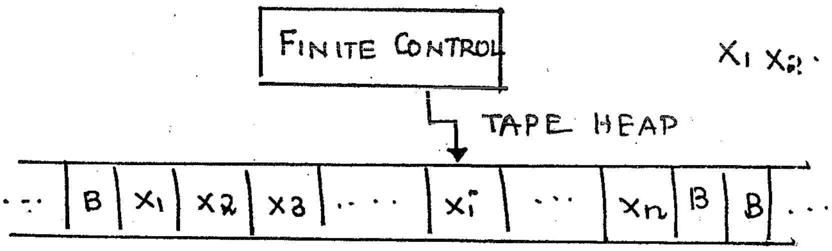
1. Finite control - which contains set of states and transitions between the states.

2. Turing Machine has an input tape (i.e) divided into cells & each cell can hold any one of the finite number of symbols over alphabet.

• ~~If~~ It has a tape head that scans one cell on the input tape at a time.

B → BLANK SYMBOL

$X_1 X_2 \cdots X_n$ → INPUT SYMBOL



INPUT TAPE

WORKING OF TURING MACHINE:

• The Turing Machine, the input initially consists of a finite length string of symbols chosen from the I/p alphabet & the i/p is placed on the input tape.

• All other tape cells extending infinitely into the left & right of the input tape contains the spirial symbol called "Blank symbol"

• The tape head is positioned at one of the tape cells for scanning the input symbol from the input tape.

• Initially the tape head points at the left most cell of the input tape

FORMAL NOTATION | DEFINITION OF A TURING MACHINE:

Turing Machine has 7-tuple:

$$\boxed{M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)}$$ where

$Q \rightarrow$ The Finite set of states of the Finite control.

$\Sigma \rightarrow$ The Finite set of input symbols.

$\Gamma \rightarrow$ The complete set of tape symbols, Σ is always a subset of Γ.
$(\Sigma \subseteq \Gamma)$

$\delta \rightarrow$ The Transition Function $\boxed{\delta(q, x) = (P, Y, D)}$.

where $q \rightarrow$ a state, $x \rightarrow$ a tape symbol, $P \rightarrow$ new state | same state in Q, $Y \rightarrow$ symbol in Γ, written in the cell being scanned, replacing whatever symbol was there.

$D \rightarrow$ Direction, either Left or Right and telling us the direction in which the head moves.

$q_0 \rightarrow$ The start state, a member in Q, in which the Finite control is found initially.

$B \rightarrow$ The blank symbol. This symbol is in Γ but not in Σ

$F \rightarrow$ The set of Final / Accepting states i.e $F \subseteq Q$.

PROCESSING OF MOVE IN A TURING MACHINE:

- The single move of a Turing Machine depends on the current state of Finite control and the tape symbol present in the input tape.
- The Following changes happen in one ~~one~~ move of a TM.

$\rightarrow$ Changes the state after consuming an i/p symbol. It may also be in the same state or transfer to any new state

$\rightarrow$ The Tape symbol to be replaced for the scanned i/p tape symbol

→ Deciding the move of the tape head to left or right of i/p tape

→ Whether to halt the TM or not.

INSTANTANEOUS DESCRIPTIONS OF A TM: (ID).

- The execution sequence of an i/p string is represented by the ID of a TM.
- Each move of TM is represented by the ID.
- ID of a TM describes the current configuration and it can be of following types

✓	Accepting configuration
✓	Rejecting Configuration

- A move of TM can be represented as a pair of ID separated by the symbol $\vdash$;
- Each move is represented by $\alpha_1 q \alpha_2$ where

  α_1 & α_2 are the strings from T^* and q is the state of TM
- The move can be of single move or zero or more moves as

  $\vdash_m$ = single move $\vdash_m^*$ = zero/more moves.

Let us use the string

$\underline{x_1 x_2 \ldots x_{i-1}\, q\, x_i\, x_{i+1} \ldots x_n}$ to represent ID.

where 1. q is the state of TM.

2. The Tape head is scanning the ith symbol from left.

3. $x_1 x_2 \ldots x_n$ is the portion of the tape between the leftmost & rightmost non-Blank.

If the transition function of TM is

CASE 1: $\boxed{\delta(q, x_i) = (P, Y, L)}$

i.e. the next move is leftward. Then

$x_1 x_2 \ldots x_{i-1}\, q\, x_i\, x_{i+1} \ldots x_n \vdash_m x_1\, x_2 \ldots x_{i-2}\, P\, x_{i-1}\, Y\, x_{i+1} \ldots x_n$

NOTE: This move reflects the change to state P and the fact that the tape head is now positioned at cell $i-1$.

There are 2 important exceptions

1. If $i=1$, then M moves to the blank to the left of $x_1 \cdots x_n$. In that case, $x_1 x_2 \cdots x_{i-1}\, q\, x_i\, x_{i+1} \cdots x_n \vdash_m PBYx_2 \cdots x_n$.
2. If $i=n$, then the symbol B written over x_n joins the infinite sequence of trailing blanks and doesn't appear in next ID.

$$x_1 x_2 \cdots x_{n-1}\, q\, x_i \cdots x_n \vdash_m x_1 x_2 \cdots x_{n-2}\, P\, x_{n-1} Y$$

CASE 2: $\boxed{\delta(q, x_i) = (P, Y, R)}$ i.e, the next move is Rightward, then

$x_1 x_2 \cdots x_{i-1}\, q\, x_i\, x_{i+1} \cdots x_n \vdash_m x_1 x_2 \cdots x_{i-1} Y P x_{i+1}$. Here the move reflects the fact that the head is $\cdots x_n$ moved to cell etc.

AGAIN THERE ARE 2 IMPORTANT EXCEPTIONS:

1. If $i=n$, then the $i+1^{st}$ cell holds a blank and that cell was not part of the previous ID. Thus we insert,

$$x_1 x_2 \cdots x_{i-1}\, q\, x_i\, x_{i+1} \cdots x_n \vdash_m x_1 x_2 \cdots x_{n-1} Y P B$$

2. If $i=1$ & $Y=B$, then the symbol B written over x_1 joins the infinite sequence of leading blanks & doesn't appear in next ID

i.e $x_1 x_2 \cdots q\, x_i \cdots x_n \vdash_m YPx_2 \cdots x_n$.

LANGUAGE OF A TM:

- The set of languages accepted by TM is recursively enumerable language.
- The input string is placed on the input tape & the tape head begins at the leftmost input symbol

If the TM enters an accepting state, then i/p is accepted else the i/p string is not accepted.

The languages accepted by TM M is defined as L(M) and it is denoted by $L(M) = \{ w \mid w \text{ is in } \Sigma^* \; q_0 w \vdash_m^* \alpha_1 P \alpha_2$ for some state P in F & α_1 and α_2 is in $\Gamma^* \}$.

HALTING OF TM:

- There is another notion of "acceptance" i.e commonly used for TM: acceptance by halting.
- We say a TM halts if it enters a state q, scanning a i/p tape symbol X, and there is no move in this situation (i.e) $\delta(q, X)$ is undefined.
- TM always halts when it is an accepting state. Unfortunately, it is not always possible to require that a TM halts even if it doesn't accept.
- Those lang with TM that donot halt eventually, regardless of whether or not they accept are called recursive.
- TM that always halt, regardless of whether or not they accept, are a good model of an "algorithm". If an algorithm to solve a given problem exists, then we say the problem is "decidable". So TM's that always halt.

COMPUTABLE LANGUAGE AND FUNCTIONS:

DESIGN A TM FOR COMPUTABLE FUNCTIONS

PROBLEMS:

1. DESIGN a TM to process zero function such that $f(x) = 0$. where x is input.

SOLUTION :

STEP 1 : IDEA OF CREATION :

The idea to design this TM is that X is the i/p, if X=5, then i/p tape contains 5 no. of 1's in the input and steps are as follows.

(i) The TM initially in the state q_0 and if it reads '1' as the left most symbol, it replaces '1' to 'B' & moves to right without changing the state.

(ii) The TM remains in the same state q_0 and replaces all 1's to 'B' until it sees 'B'.

(iii) At state q_0, if it finds 'B' it enters the final state q_1, then halt the TM.

STEP 2 : DIAGRAMMATIC REPRESENTATION.

EXAMPLE X = 3.

INPUT TAPE

1	1	1	B	..

$\uparrow q_0$

$(q_0, 1) = (q_0, B, R)$.

B	1	1	B

$q_0 \rightarrow$

$(q_0, 1) = (q_0, B, R)$

B	B	1	B

$\uparrow q_0$

$(q_0, 1) = (q_0, B, R)$.

B	B	B	B

$\uparrow q_0$

$(q_0, B) = (q_1, B, L)$ halts.

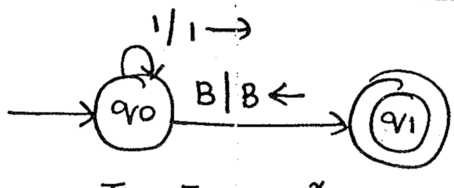
B	B	B	B

$\leftarrow q_1$

STEP 3 : TRANSITION TABLE

STATE	1	B
$\rightarrow q_0$	(q_0, B, R)	(q_1, B, L)
$* q_1$	–	–

STEP 4 : TRANSITION DIAGRAM.



TM FOR $f(x) = 0$.

STEP 5: TM DEFINITION IS $M = (\{q_0, q_1\}, \{1\}, \{1, B\}, \delta, q_0, B, \{q_1\})$

δ : $\delta(q_0, 1) = (q_0, B, R)$

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

STEP 6 : INSTANTANEOUS DESCRIPTION :

EXAMPLE $x = 2$ $\delta(q_0, 11B) \vdash_m (Bq_0 1B) \vdash_m (BBq_0 B) \vdash_m (Bq_1 BB)$

String accepted and all 1's changed to Blank and the zero function is implemented.

2. Design a TM to implement the Function $f(n) = x+1$.

SOLUTION : If $x = 3$ then

Input Tape

1	1	1	B	...

output Tape

1	1	1	1	B	...

STEP 1 :

1. TM is initially in the state q_0 and it reads '1' in the leftmost input tape.

2. At state q_0 when it reads '1' it remains in the same state, without changing '1' and just move the tape head to right

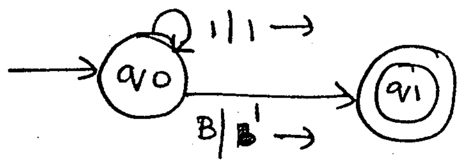
3. At state q_0, it skips all 1's and searches for the 1st blank symbol B.

4. At state q_0, when it finds 1st 'B', it enters the final state q_1 & changes 'B' to '1'.

STEP 2: TRANSITION TABLE

	1	B
$\rightarrow q_0$	$(q_0, 1, R)$	$(q_1, 1, R)$
$* q_1$	–	–

STEP 3: TRANSITION DIAGRAM.



TM for $f(x) = x+1$.

STEP 4: TM Definition $M = (\{q_0, q_1\}, \{1\}, \{1, B\}, \delta, q_0, B, \{q_1\})$

δ: $\delta(q_0, 1) = (q_0, 1, R)$

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

STEP 4: INSTANTANEOUS DESCRIPTION: $x = 3$

$\delta(q_0, 111B) \vdash_m (1q_011B) \vdash_m (11q_01B) \vdash_m (111q_0B) \vdash_m (1111q_1B)$

String is accepted.

3. Design a TM to implement the function $f(x) = x+2$.

SOLUTION: EXAMPLE: $X=3$

Input tape:

1	1	1	B	...

output tape:

1	1	1	1	1	B	..

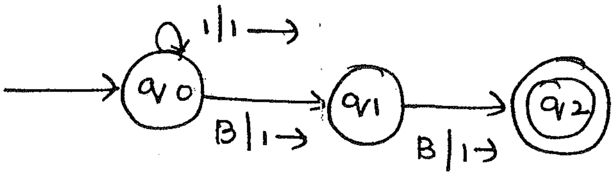
STEP 1:

1. At state q_0, the initial state of TM, it reads the leftmost 1, it skips 1 and searches for the 1st Blank symbol 'B' and moves to right
2. At state q_0, when it reads 1st B, it changes B to '1' and moves to right to see the next Blank symbol 'B' and changes to 'q_1'
3. At state q_1, when it finds the 2nd 'B' blank symbol, it changes B to '1' and moves to right and enters the accepting state q_2.

STEP 2: TRANSITION TABLE:

	1	B
$\rightarrow q_0$	$(q_0, 1, R)$	$(q_1, 1, R)$
q_1	-	$(q_2, 1, R)$
$* q_2$	-	-

STEP 3: TRANSITION DIAGRAM.



TM for $f(x) = x+2$.

STEP 4: TM definition $M = (\{q_0, q_1, q_2\}, \{1\}, \{1, B\}, \delta, q_0, B, \{q_2\})$

δ: $\delta(q_0, 1) = (q_0, 1, R)$

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

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

STEP 5: ID $x=3$.

$$\delta(q_0, 111B) \vdash_m (q_0 111B) \vdash_m (1q_0 11B) \vdash_m (11q_0 1B) \vdash_m (111q_0 B) \vdash_m (1111q_1 B) \vdash_m (11111q_2 B)$$

String is accepted.

4. Design a TM to implement the concatenation function $f(x, y) = xy$ (or) to implement addition function $f(x, y) = x+y$

SOLUTION:

STEP 1:

Let us assume that x is represented by the 1^x and y is represented by 1^y in the input tape. The 1^x and 1^y is separated by the separator symbol '#' and is shown below.

$x=2 \quad y=3$

Input:

1	1	#	1	1	1	B	...

output:

1	1	1	1	1	B	...

$x+y = 2+3 = 5$

The sum of 2 values are performed by replacing the last '1' by Blank symbol and the steps are as follows :

a. At initial state q0, when it reads '1', it skips the 1's and remain in the same state.

b. At state q0, when it reads '#' it reaches the state q1 and changes '#' to '1' and moves right

c. At state q1, it skips all 1's and searches for 'B' by moving righ

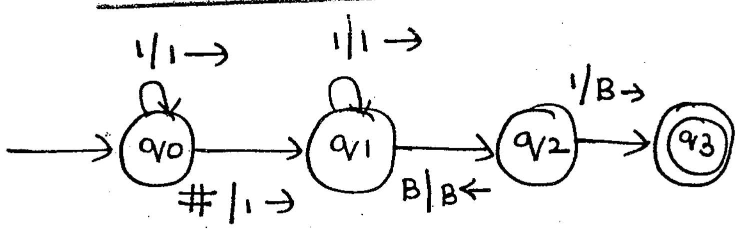
d. At state q1, when it sees blank symbol, it moves left and changes state to q2.

e. At state q2, when it finds '1' it replaces '1' to B and enters the Final state q3.

STEP 2: TRANSITION TABLE

State	1	#	B
→ q0	(q0,1,R)	(q1,1,R)	–
q1	(q1,1,R)	–	(q2,B,L)
q2	(q3,B,R)	–	–
* q3	–	–	–

STEP 3: TRANSITION DIAGRAM.



TM for $f(x,y) = x+y$.

STEP 4: TM definition $M = (\{q0,q1,q2,q3\}, \{1\}, \{1,\#,B\}, \delta, q0, B, \{q3\})$

STEP 5: ID EXAMPLE $x=2$ $y=3$

$\delta(q0, 11\#111B) \vdash_m (q011\#111B) \vdash_m (1q01\#111B) \vdash_m (11q0\#111B)$

$\vdash_m (111q1111B) \vdash_m (1111q111B) \vdash_m (11111q11B) \vdash_m (111111q_1B)$

$\vdash_m (11111q_21) \vdash_m (11111Bq_3B)$

String accepted. The function $f(x,y) = x+y$ is implemented.

5. Design a TM to perform subtraction $f(x,y) = \begin{cases} x-y & \text{if } x>y \\ 0 & \text{if } x \le y \end{cases}$

SOLUTION :

The idea to create a TM to perform subtraction is, the i/p is represented as $1^m \# 1^n$. The value 1^m and 1^n is separated by a separator symbol '#' and $1^m \# 1^n$ is surrounded by B.

This proper subtraction function say that

$$f(m,n) = \begin{cases} m-n, & \text{if } m > n \\ 0, & \text{if } m \le n \end{cases}$$

So we have to design a TM such that if $m > n$ the subtracted value that is $1^m - 1^n$ should be on the tape. And if $m \le n$, then tape should have only 'B'

If $m = 4, n = 2$ (i.e) $m > 2$

Input:

1	1	1	1	#	1	1	B	...

output:

B	B	1	1	B	B	...

$m - n = 2$

If $m = 2, n = 4, m \le n$.

Input:

1	1	#	1	1	1	1	B	...

output:

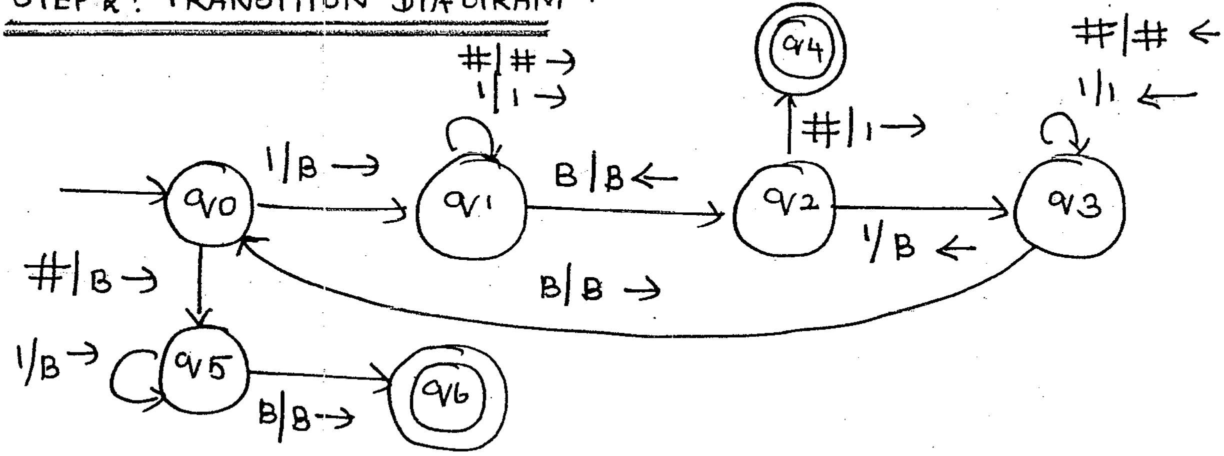
B	B	B	B	...

$m - n = 0$

- The idea to design this TM is that the TM process in such a way that for each '1' on the Leftmost side, it replaces '1' on the right most side to 'B'. ['1' appearing before 'B']
- After replacing with 1's to the left and right when the m/c encounters separator symbol on right side, it is clear that n value ends.
- When 'n' value ends, it starts replacing '#', to '1' and enters final | accepting state.
- Similarly if $m \le n$, then m/c encounters the symbol '#'

from initial state then it starts replacing all 1's and '#' to Blank and enter the Final state.

STEP 2: TRANSITION DIAGRAM:



STEP 3: TRANSITION TABLE:

	1	#	B
→ q_0	(q_1, B, R)	(q_5, B, R)	–
q_1	$(q_1, 1, R)$	$(q_1, \#, R)$	(q_2, B, L)
q_2	(q_3, B, L)	$(q_4, 1, R)$	–
q_3	$(q_3, 1, L)$	$(q_3, \#, L)$	(q_0, B, R)
* q_4	–	–	–
q_5	(q_5, B, R)	–	(q_6, B, R)
* q_6	–	–	–

STEP 4: TM definition $M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{1\}, \{1, \#, B\}, \delta, q_0, B, \{q_4, q_6\})$

STEP 5: ID: $m = 2 \quad n = 1$

$\delta(q_0, 11\#1B) \vdash_m (q_0 11\#1B) \vdash_m (Bq_1 \#1B) \vdash_m (B1q_1 \#1B) \vdash_m$
$(B1\#q_1 1B) \vdash_m (B1\# 1q_1 B) \vdash_m (B1\# q_3 B) \vdash_m (B1q_3\#B) \vdash_m (Bq_3 1\#B$
$\vdash_m (q_3 B1\#B) \vdash_m (Bq_0 1\#B) \vdash_m (BBq_1\#B) \vdash_m (BB\# q_1 B) \vdash_m (BBq_2\#B$
$\vdash (BB1q_4 B)$

String accepted and now the input tape contain one 1's and the function $f(m-n) = m-n$ is implemented.

Eg:2 $m=1, n=2$.

$\delta(q_0, 1\#11B) \vdash_m (q_0 1\#11B) \vdash_m (Bq_1\#11B) \vdash_m (B\#q_1 11B) \vdash_m (B\#1q_1 1$

$\vdash_m (B\#11q_1B) \vdash_m (B\#1q_2 1B) \vdash_m (B\#q_3 1B) \vdash_m (Bq_3\#1B)$

$\vdash_m (q_3 B\#1B) \vdash_m (Bq_0\#1B) \vdash_m (BBq_5 1B) \vdash_m (BBBq_5 B) \vdash_m (BBBBq_6$

String accepted. Since m is less than n, then the i/p tape contains zero value.

6. Design a TM to implement multiplication function $f(x,y) = x*y$.

STEP 1:

The idea to design this TM is that we place the input as $1^x\#1^y\#$ on the TM. Now the multiplication is done by performing successive addition and it is shown below.

$x=2 \quad y=3$

Input:

1	1	#	1	1	1	#	B	..

$x=2 \quad y=3$

$x*y = 2*3 = 6.$

output:

B	B	B	B	B	1	1	1	1	1	1	B	..

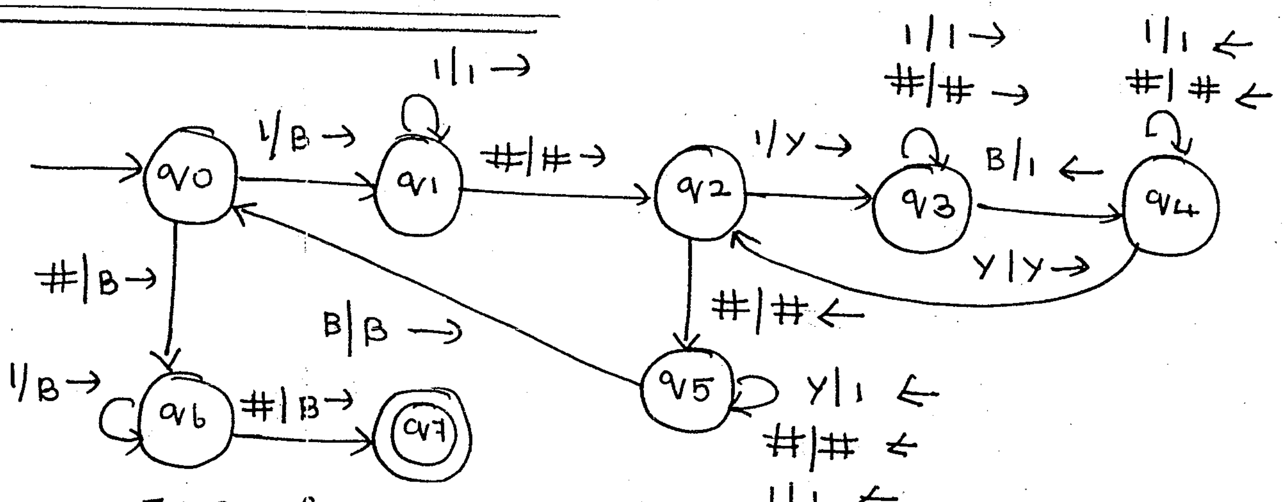
STEPS:

a. At initial state when '1' finds in the i/p, replace it to 'B' and move right for searching #

b. After Finding '#', copy the 'Y' no. of 1's for 'x' no. of times in B symbols

c. After performing 'x' no. of copy with 'y' no. of 1's we replac $\#1^y\#$ to 'B' then reach to final state and the tape contain 1^{xy}.

STEP 2: TRANSITION DIAGRAM:



TM for $f(x,y) = x * y$.

STEP 3: Transition Table.

states	1	#	B	Y
→ q0	(q1, B, R)	(q6, B, R)	–	–
q1	(q1, 1, R)	(q2, #, R)	–	–
q2	(q3, Y, R)	(q5, #, L)	–	–
q3	(q3, 1, R)	(q3, #, R)	(q4, 1, L)	–
q4	(q4, 1, L)	(q4, #, L)	–	(q2, Y, R)
q5	(q5, 1, L)	(q5, #, L)	(q0, B, R)	(q5, 1, L)
q6	(q6, B, R)	(q7, B, R)	–	–
q7	–	–	–	–

STEP 4: INSTANTANEOUS DESCRIPTION: $x = 2,\ y = 1$.

$\delta(q_0, 11\#1\#B) \vdash_m (q_0 11\#1\#B) \vdash_m (B q_1 1\#1\#B) \vdash_m (B1 q_1 \#1\#B)$

$\vdash_m (B1\# q_2 1\#B) \vdash_m (B1\#Y q_3 \#B) \vdash_m (B1\#Y\# q_3 B) \vdash_m (B1\#Y q_4 \#.$

$\vdash_m (B1\# q_4 Y\#1) \vdash_m (B1\#Y q_2 \#1) \vdash_m (B1\# q_5 Y\#1) \vdash_m (B1 q_5 \#1\#1)$

$\vdash_m (B q_5 1\#1\#1) \vdash_m (q_5 B1\#1\#1) \vdash_m (B q_0 1\#1\#1) \vdash_m (BB q_1 \#1\#1)$

$\vdash_m (BB\# q_2 1\#) \vdash_m (BB\#Y q_3 \#1B) \vdash_m (BB\#Y\# q_3 1B)$

$\vdash_m (BB\#Y\#1q_3B) \vdash_m (BB\#Y\#q_4 11) \vdash_m (BB\#y\,q_4\#11)$

$\vdash_m (BB\#q_4 y\#11) \vdash_m (BB\#y q_2\#11) \vdash_m (BB\#q_5 y\#11) \vdash_m (BB q_5\#\#11)$

$\vdash_m (Bq_5 B\#1\#11) \vdash_m (BB q_0\#1\#11) \vdash_m (BBB q_6 1\#11) \vdash_m$

$(BBBB q_6\#11) \vdash_m (BBBBB q_2 11)$

String is accepted and the $f(x,y) = f(x * y$ is implemented.

7. Design a TM to perform 1's complement of a no. over $\Sigma = \{0,1\}$.

SOLUTION:

On Reading the i/p;

→ If the symbol = 0 replaces it by '1' & move right

→ If the symbol = 1 replace it by '0' & move right

→ Perform step 1 & 2 until the i/p symbols are processed from left to right

→ Halt the m/c when it encounters the 1st Blank symbol.

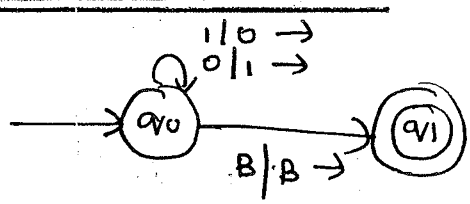
Example: 1011 → 0100

i/p o/p.

STEP 2: TRANSITION TABLE:

	0	1	B
$\rightarrow q_0$	$(q_0,1,R)$	$(q_0,0,R)$	(q_1,B,R)
$* q_1$	(–	–	–

STEP 3: TRANSITION DIAGRAM.



STEP 4: TM Definition $M = (\{q_0,q_1\}, \{0,1\}, \{0,1,B\}, \delta, q_0, B, \{q_1\})$

STEP 5: I/p. $w = 101$

$\delta(q_0, 101B) \vdash_m (q_0 101B) \vdash_m (0 q_0 01B) \vdash_m (01 q_0 1B) \vdash_m (010 q_0 B)$

$\vdash_m (010B q_1)$

String accepted and 1's complement is implemented.

8. Design a TM to perform 2's complement of a no over $\Sigma = \{0, 1\}$.

NOTE: Don't change the bits from the right towards left until the 1st 1 has been processed perform complementation to the rest of the bits from right to left [after 1st 1 is processed]

SOLUTION:

STEP 1:

a. Traverse Right & Locate Right most bit.

b. If the bit = 0, perform no replaces & move left.

c. If the bit = 1, perform no change & move left.

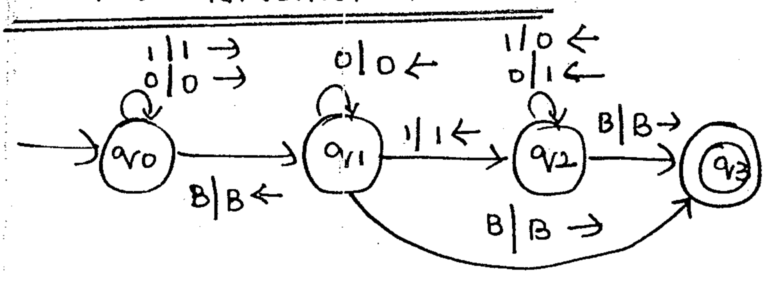
d. If the next bit symbol = '0' replace it by '1' and move left.

e. Else if the next bit symbol = '1' replace it by '0' & move left.

f. Perform steps until all the i/p symbols are processed [From Right to Left]

g. Halt the m/c.

STEP 2:- TRANSITION DIAGRAM:



STEP 3: TRANSITION TABLE

	0	1	B
$\rightarrow q_0$	$(q_0, 0, R)$	$(q_0, 1, R)$	$(q_1, B$
q_1	$(q_1, 0, L)$	$(q_2, 1, L)$	$(q_3, B,$
q_2	$(q_2, 1, L)$	$(q_2, 0, L)$	$(q_3, B,$
$* q_3$	–	–	–

STEP 4: TM Definition:

$$M = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, B, \{q_3\})$$

STEP 5: ID. $w = 101$.

$$\delta(q_0, 101B) \vdash_m (q_0 101B) \vdash_m (1 q_0 01B) \vdash_m (10 q_0 1B) \vdash_m (101 q_0 B)$$

$$\vdash_m (10 q_1 1B) \vdash_m (1 q_2 01B) \vdash_m (q_2 111B) \vdash_m (q_2 B011B) \vdash_m (B q_3 011B)$$

String is accepted and function is implemented.

COMPUTABLE LANGUAGE

1. Design a TM that accepts the language $L = \{a^n b^n \mid n \geq 1\}$.

SOLUTION:

STEP 1: IDEA OF CREATION:

a. The idea to create this TM is to place $a^n b^n$ in the I/p tape

b. Let the TM initially be in the state q_0 (initial state).

c. while in q_0, the machine reads 'a' and changes ~~to~~ 'a' to X and moves to the right and changes its state to q_1 and starts scanning the next input

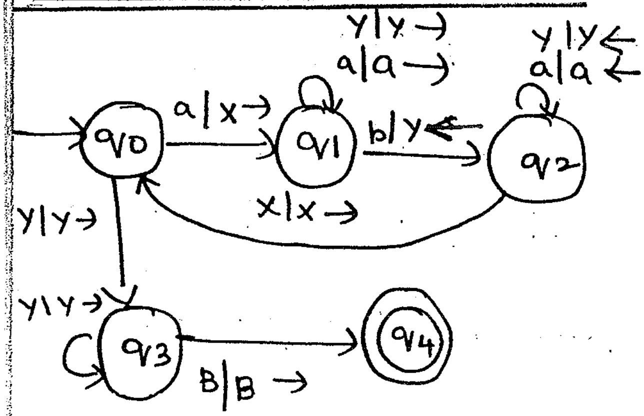
d. From the q_1, while reading 'a' it doesnot change state but simply moves to the right until seeing 1st 'b'

e. When seeing 'b' from state q_1, it reach the state q_2 and change 'b' to 'y' and moves to left to see 'X'

f. The From state q_2 when it sees X, it the state to q_0 and repeat the process.

g. The major idea is that for each 'a', we try to 'b' and alternatively, the process is repeated

STEP 2: TRANSITION DIAGRAM.



REJECTING STATE.

$(q_3, b) = (q_{reject}, b, R)$ [b>a]

$(q_3, a) = (q_{reject}, a, R)$ [ba]

$(q_3, b) = (q_{reject}, B, R)$ [a>b]

STEP 3: TRANSITION TABLE.

	a	b	Y	X	B
$\rightarrow q_0$	(q_1, X, R)	–	(q_3, Y, R)	–	–
q_1	(q_1, a, R)	(q_2, Y, L)	(q_1, Y, R)	–	–
q_2	(q_2, a, L)	–	(q_2, Y, L)	(q_0, X, R)	–
q_3	–	–	(q_3, Y, R)	–	(q_4, B, R)
$* q_4$	–	–	–	–	–

STEP 4: TM definition $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{a, b\}, \{a, b, X, Y, B\}, \delta, q_0, B, \{q_4\})$

STEP 5: ID $w_1 = aabb$.

$\delta(q_0, aabb) \vdash_m (q_0 aabb) \vdash_m (X q_1 abb) \vdash_m (X a q_1 bb) \vdash_m (X q_2 a Y 1)$
$\vdash_m (q_2 X a Y b) \vdash_m (X q_0 a Y 1) \vdash_m (X X q_1 Y 1) \vdash_m (X X Y q_1 1) \vdash_m (X X q_2 Y Y)$
$\vdash_m (X q_2 X Y Y) \vdash_m (X X q_0 Y Y) \vdash_m (X X Y q_3 Y) \vdash_m (X X Y Y q_3 B) \vdash_m (X X Y Y B q_4)$

String "aabb" is accepted.

ID $w_2 = aab$.

$\delta(q_0, aab) \vdash_m (q_0 aab) \vdash_m (X q_1 ab) \vdash_m (X a q_1 b) \vdash_m (X q_2 a Y)$
$\vdash_m (q_2 X a Y) \vdash_m (q_0 a Y) \vdash_m (X X q_1 Y) \vdash_m (X X Y q_1 B)$

String "aab" is rejected.

2. Design a TM that accepts the language $L = \{a^n b^n c^n \mid n \geq 1\}$.

SOLUTION:

The Construction is similar to the design $a^n b^n$. Here we have to replace each 'a' by 'x' & 'b' by 'y' and 'c' by 'z' respectively.

IDEA :

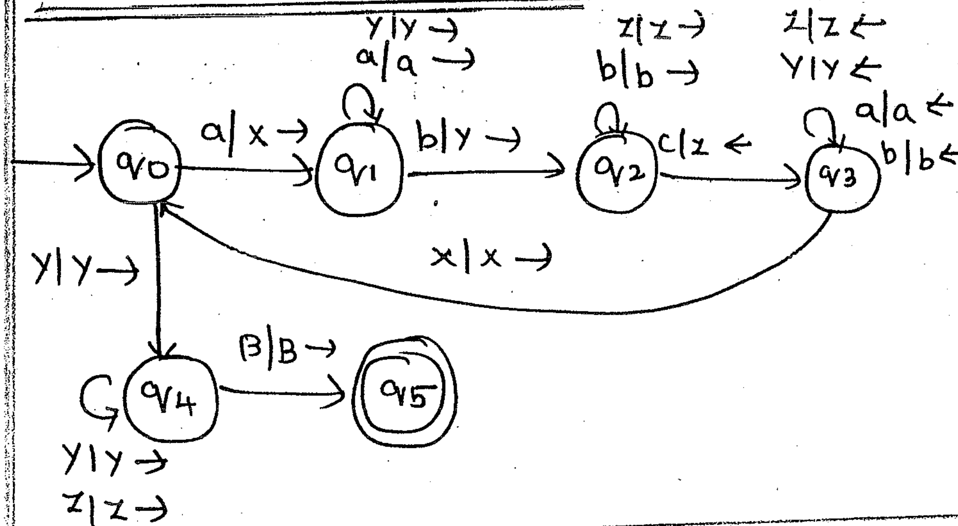
a. Initially the TM is at q_0. At q_0 if it finds a's replace it by x's and move right with state q_1.

b. At q_1, if it finds b's, replace it by y's and moves right with state q_2.

c. At state q_2, if it finds c's replace it by z and enters q_3 by moving left

d. At q_3, if it finds the leftmost x by skipping z by a then it goes to state q_0. Repeat the process till at q_0, if finds y.

STEP 2: TRANSITION DIAGRAM.



REJECTING STATE.

$(q_3, \overset{c}{\cancel{b}}) = (q_{reject}, \overset{c}{\cancel{b}}, R)$

$(q_3, a) = (q_{reject}, a, R)$

$(q_3, b) = (q_{reject}, b, R)$

~~$(q_3, b$~~

STEP 3: TRANSITION TABLE.

	a	b	c	x	y	z	B
q_0	(q_1, Y, R)	–	–	–	(q_4, Y, R)	–	–
q_1	(q_1, a, R)	(q_2, Y, R)	–	–	(q_1, Y, R)	–	–
q_2	–	(q_2, b, R)	(q_3, Z, L)	–	--	(q_2, Z, L)	–
q_3	(q_3, a, L)	(q_3, b, L)	–	(q_0, X, R)	(q_3, Y, L)	(q_3, Z, L)	–
q_4	–	–	–	–	(q_4, Y, R)	(q_4, Y, R)	(q_5, B, R)
q_5	–	–	–	–	–	–	–

STEP 4: TM Definition $M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{a, b, c\}, \{a, b, c, x, y, z, B\}, \delta, q_0, B, \{q_5\})$.

STEP 5: ID $w_1 = aabbcc$.

$\delta(q_0, aabbcc) \vdash_m (q_0 aabbcc) \vdash_m (xq_1abbcc) \vdash_m (xaq_1bbcc)$
$\vdash_m (xayq_2bcc) \vdash_m (xaybq_2cc) \vdash_m (xayq_3bzc) \vdash_m (xaq_3ybzc)$
$\vdash_m (xq_3aybzc) \vdash_m (q_3xaybzc) \vdash_m (xq_0aybzc) \vdash_m (xxq_1ybzc)$
$\vdash_m (xxyq_1bzc) \vdash_m (xxyyq_2zc) \vdash_m (xxyyzq_2c) \vdash_m (xxyyq_3zz)$
$\vdash_m (xxyq_3yzz) \vdash_m (xxq_3yyzz) \vdash_m (xq_3xyyzz) \vdash_m (xxq_0yyzz)$
$\vdash_m (xxyq_4yzz) \vdash_m (xxyyq_4zz) \vdash_m (xxyyzq_4z)$
$\vdash_m (xxyyzzq_4B) \vdash_m (xxyyzzBq_5)$

string "aabbcc" is accepted.

3. Design a TM for language L. The set of strings with an equal no. of 0's and 1's.

SOLUTION:

Assume that the i/p string may start with either 0 or 1, but it should have equal no. of 0's and 1's.

For eg 0101, 0110, 1001 ...

a. Change all 0's to x's and all 1's to y's, whether the i/p may be in any position till reaches the blank symbol.

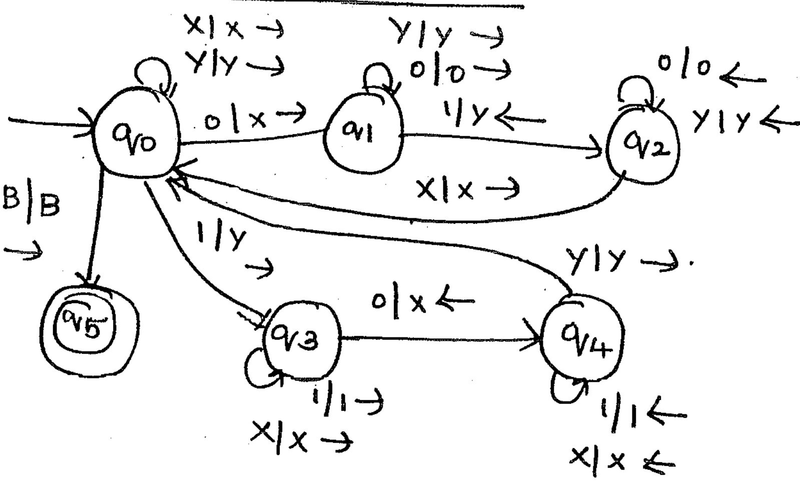
b. Initially, the TM is at state q_0. At q_0, if it finds the leftmost symbol as '0' change it to x and enters q_1

then moves right. If it finds 1 by skipping 0's y's at q1, change it to y and enters state q2. At state q2, the TM searches for the leftmost x by skipping 0's and y's and enters q0. Repeat the process till the TM finds blank symbol at q0

c. At q0, if it finds the leftmost symbol as 1, change it to y and enters state q3. At q3, if it finds 0's by skipping 1's and x's, change it to x and enters state q4 by moving left. At q4, it searches for the leftmost y. If it finds y at q4, the TM enters state q0. Repeat the process till it finds blank symbol.

d. For all other state changes, the input is rejected.

STEP 2: TRANSITION DIAGRAM:



REJECTING STATE.

(q3, 1) = (qreject, B, R).

(q1, B) = (qreject, B, R).

STEP 3: TABLE:

	0	1	X	Y	B
→ q0	(q1, X, R)	(q3, Y, R)	(q0, X, R)	(q0, Y, R)	(q5, B, R)
q1	(q1, 0, R)	(q2, Y, L)	—	(q1, Y, R)	-
q2	(q2, 0, L)	—	(q0, X, R)	(q2, Y, L)	-
q3	(q4, X, L)	(q3, 1, R)	(q3, X, R)	—	-
q4	—	(q4, 1, L)	(q4, X, L)	(q0, Y, R)	-
q5	—	—	—	—	-

STEP 4: TM definition $M = (\{q_0, q_1, q_2, q_3, q_4, q_5\}, \{0,1\}, \{0,1,X,Y,B\}, \delta, q_0, B, \{q_5\})$

STEP 5: ID $w_1 = 1001$

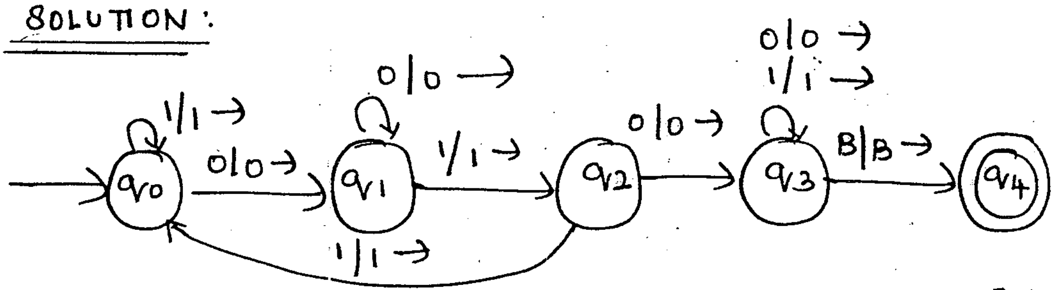
$\delta(q_0, 1001) \vdash_m (q_0 1001) \vdash_m (Y q_3 001) \vdash_m (q_4 Y X 01) \vdash_m (Y q_0 X 01)$
$\vdash_m (Y X q_6 01) \vdash_m (Y X X q_1 1) \vdash_m (Y X q_2 X Y 1) \vdash_m (Y X X q_0 Y) \vdash_m (Y X X Y q_0 B$
$\vdash_m (Y X X Y B q_5) \Rightarrow$ string is accepted.

$w_2 = 0100$

$\delta(q_0, 0100) \vdash_m (q_0 0100) \vdash_m (q_2 X Y 00) \vdash_m (X q_0 Y 00) \vdash_m$
$(X Y q_0 00) \vdash_m (X Y X q_1 0) \vdash_m (X Y X 0 q_1 B) \Rightarrow$ Rejected [No Transition]

4. Design a TM to accept the language L contains a substring "010"

SOLUTION:



TM: $M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0,1\}, \{0,1,B\}, \delta, q_0, B, \{q_4\})$.

5. Design the TM to accept the language of palindromes over the alphabet $\{a,b\}$ or to accept the lang. $L = \{ww^R \mid w \in \{a,b\}\}$.

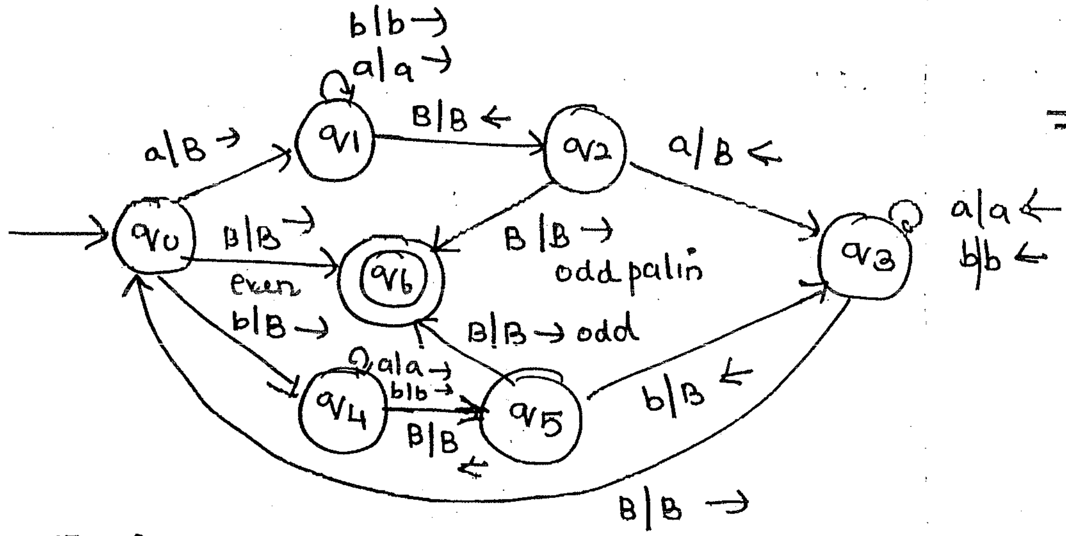
SOLUTION:

STEP 1: IDEA OF CREATION:

- The TM that we are designing now should accept the strings of palindromes such as ababa, abbbba .... The idea to design this TM, is that if the m/c reads 'a' on the left most symbol, replace 'a' to 'B' and move to right and changes last 'a' to B.

- Similarly if the m/c reads 'b' then it replaces b to B and moves to right by searching B and last b and replace b to B.
- So the overall idea is for each 'a' that is first 'a' on the left it matches the last 'a' on the right most side and for each b on the 1st time on the left, it matches last b on right si

STEP 3: TRANSITION DIAGRAM.



REJECTING STATE

$\delta(q_2, a) = (q_{reject}, a, L)$

$\delta(q_5, b) = (q_{reject}, b, L)$

TM for $L = \{ww^R \mid w \in (a,b)^*\}$

STEP 4: $M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \{a,b\}, \{a,b,B\}, \delta, q_0, B, \{q_6\})$.

6. Design the TM to compute the fn $F(w) = w\,c\,w^R$. where w is any string of a's & b's.

SOLUTION:

STEP 1: IDEA OF CREATION.

- The idea to create this TM is that to read the string w and to create wcw^R.

→ Here we initially read all the symbols in the string w upto 'B' and then moves on the left one position and symbol.

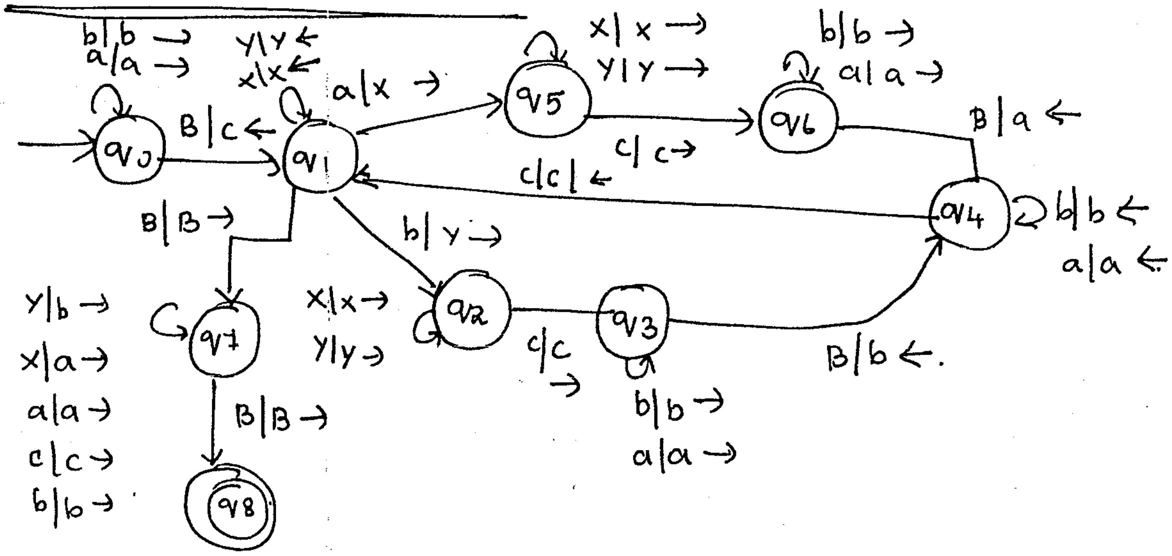
→ If the symbol is 'a', then we replace it by x and if the symbol is 'b', it is replaced by Y.

→ After replacing the symbol, we move to the right and replace B by 'a' or 'b' based on the symbol read before the B.

→ After processing all the strings w and we replace 'x' by 'a' and 'Y' by b.

→ After replacing the entire string symbol in 'w', we move to the right side until blank symbol.

STEP 2: Transition Diagram.



Rejecting state

$\delta(q_1, a) = (q_{reject}, a, L)$

$\delta(q_1, b) = (q_{reject}, b, L)$.

STEP 3 : TRANSITION TABLE

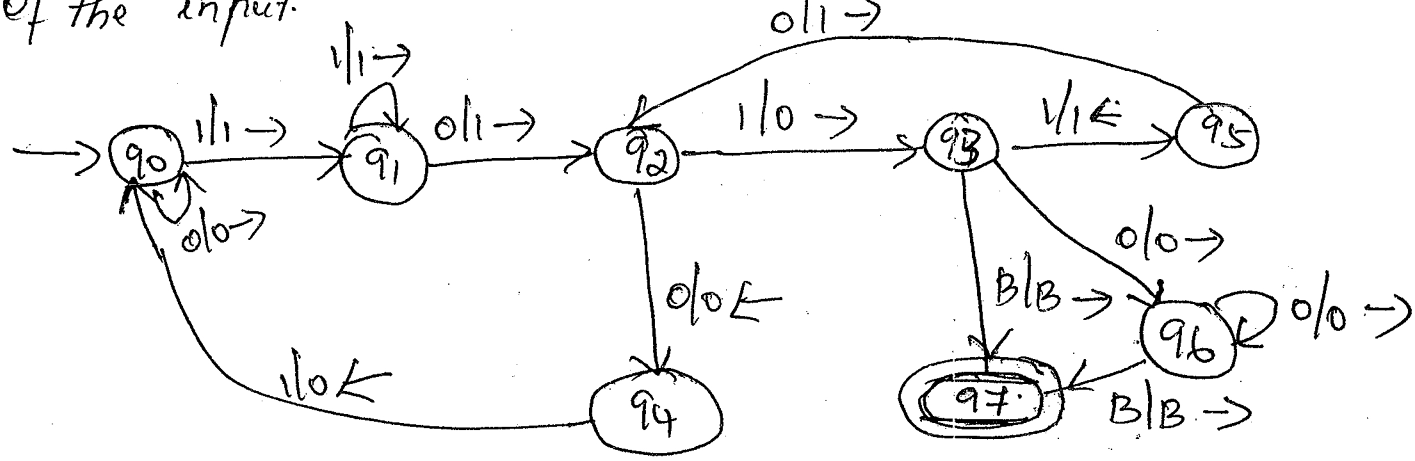
STEP 5: ID - any string.

STEP 4: TM Definition

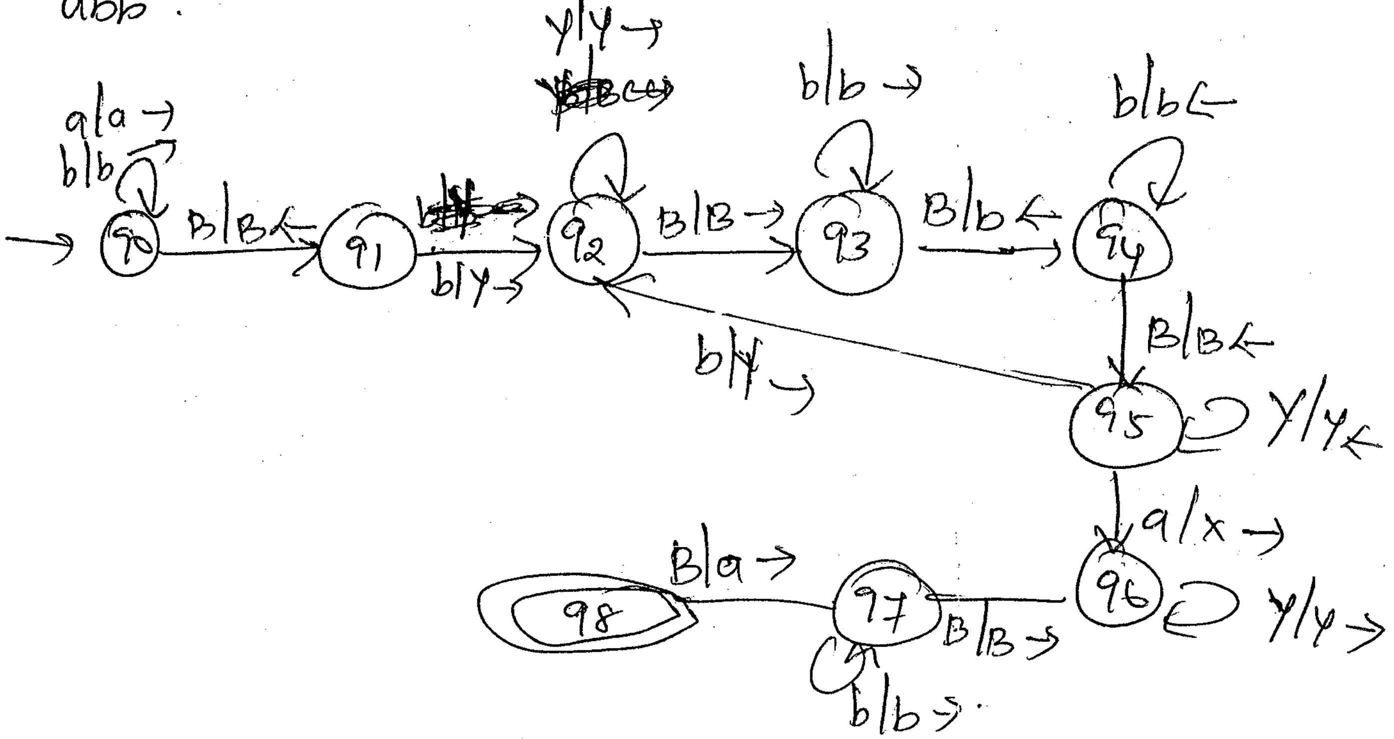
$M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}, \{a, b, c\}, \{a, b, c, B\}, \delta, q_0, B, \{q_8\})$

Design a TM which recognizes the input language having a substring as 101 and replaces every occurrence of 101 by 110.

Soln The TM has to be constructed considering 101 as a substring and leaving 110 substring after complete scan of the input.

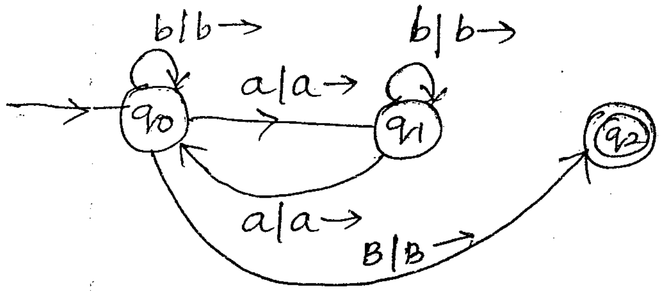


Design a TM which reverse the given string "abb".



Qn: Design the TM to accept the set of all strings over alphabet $\{a, b\}$ with even number of a's.

Solution:

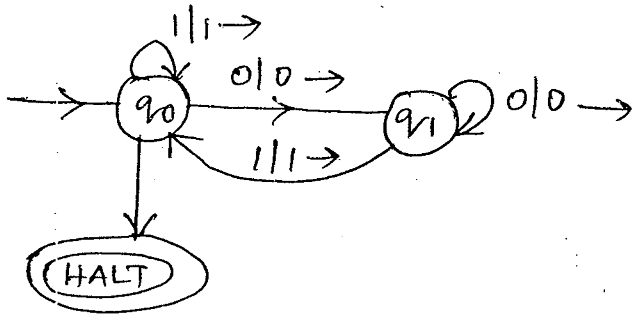


TM $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, b, B\}, \delta, q_0, B, \{q_2\})$

Qn: Design a TM that accepts the language of odd integers written in binary.

Soln:

Logic: The binary string that ends with 1 is always an odd integer. Hence the TM will be



Techniques for Turing Machine construction

* Here let us see some of the programming techniques that are used to construct an efficient TM that functions as powerful as a conventional computer.

* The different techniques that are used to design a TM are as follows,

(1) storage in finite control

(2) Multiple tracks or multi head TM

(3) Multiple tape (or) multi tape TM

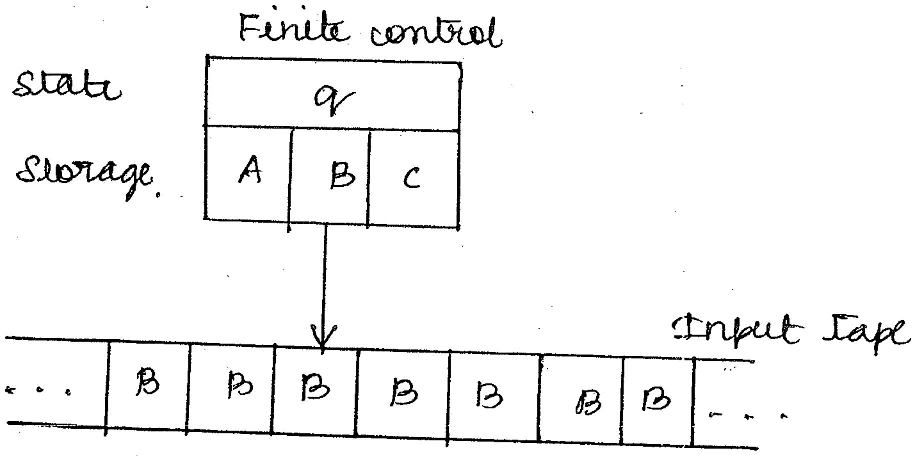
(4) Subroutines

Storage in finite control :

* In TM, generally the finite control contains the FA with the state transitions.

* And the finite control in TM represents the set of states.

* But here in the storage of finite control, we store the data along with the state, so here we use the finite control to hold finite amount of data and it is shown below,



[TM with storage in finite control]

* This type of TM makes the state to remember and to have a memory for the symbol scanned in the input. From the above TM, the state is q and this state q contains A, B, C as the symbol in storage

with q.

* This type of TM can be designed to store in the state with any data from the i/p.

* Each state contains the 'B' blank symbol as its storage initially.

* This type of TM is used to store any symbol in the input and to check whether the stored symbol appears in the input.

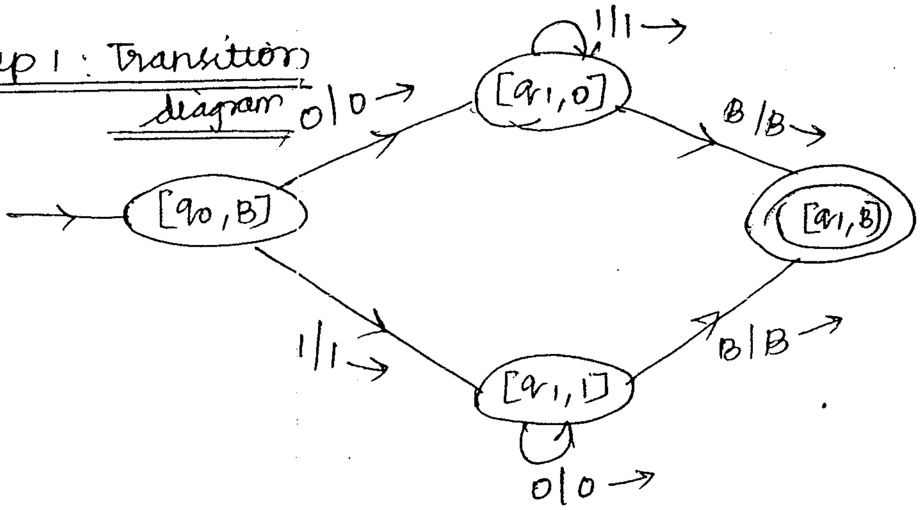
PROBLEMS [For storage in finite control]

Qn: Design a TM to accept the string $01^* + 10^*$.

Soln:

To design a TM, that it should accept the strings such as 01111, 10000.... etc so the string should have the first symbol as '0' or '1' and it should not appear else where in the input.

Step 1: Transition diagram



Step 2: Transition table

state	0	1	B
$\rightarrow [q_0, B]$	$([q_1, 0], 0, R)$	$([q_1, 1], 1, R)$	–
$[q_1, 0]$	–	$([q_1, 0], 1, R)$	$([q_1, B], B, R)$
$[q_1, 1]$	$([q_1, 1], 0, R)$	–	$([q_1, B], B, R)$
$* [q_1, B]$	–	–	–

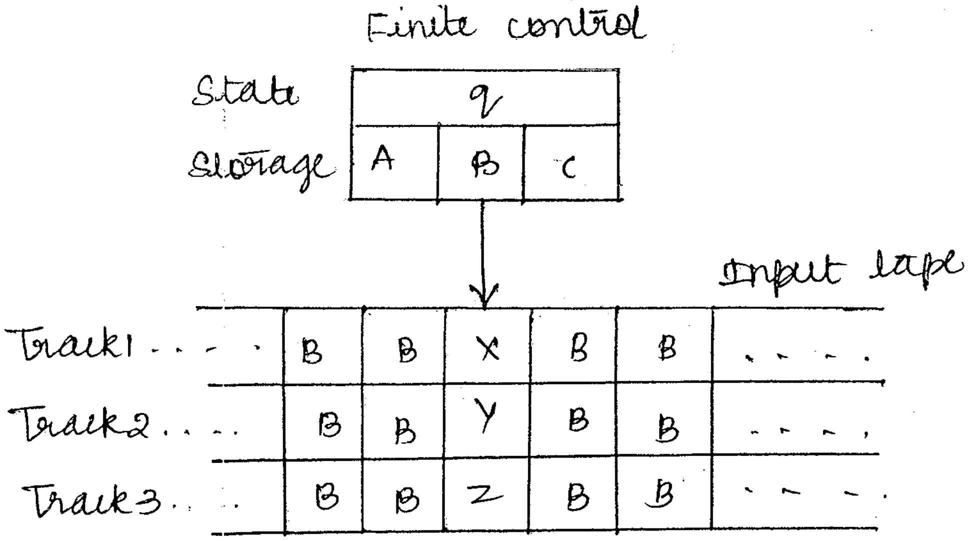
Step 3 : TM definition

$$\text{TM } M = (\{[q_0, B], [q_1, 0], [q_1, 1], [q_1, B]\}, \{0, 1\}, \{0, 1, B\}, \delta, [q_0, B], B, \{[q_1, B]\})$$

Multiple Tracks or Multi Head Turing Machine:

Now we are going to extend this TM to include multiple tracks in the input tape.

- In this TM, where the finite control contains the state and its storage and the input tape contains multiple tracks.
- Each track in the i/p tape contains one symbol.
- The tape alphabet of TM consists of tuples with one component in each track and the number of components in the tuple depends on the number of tracks of the input tape.



- Here, the cell scanned by the tape head contains the symbol [X, Y, Z].
- The multiple tracks of TM is used to find whether the number is odd / even.
- The multiple tracks can be used to check whether the number is prime.

Example: Design a TM using multiple tracks to check whether the given input number is prime or not.

Soln:

* Store the i/p symbol in the 1st track of i/p tape.

* Store the number 2 is binary in the 2nd tracks of i/p tape.

* Copy the i/p in the 3rd Track also.

* All the symbols in the three tracks of the TM are in binary form.

* Now subtract the 2nd Track from third Track until we get '0' or any remainder.

* If the remainder is zero, then the number is not prime, since the prime number is one which is divided by 1 and itself.

* If the remainder is non-zero value, then the 2nd track value is incremented by 1 and again subtraction procedure is continued.

* If the value of the 2nd & 1st track is equal, then the number is prime number. Let us take an i/p value 5 and it is stored as,

Track 1	1	0	1	B	...
Track 2	B	1	0	B	...
Track 3	1	0	1	B	...

Divide the value 2 in 2nd track from value in 3rd track

1	0	1	B	...
B	1	0	B	...
1	0	1	B	...

$\longrightarrow$

1	0	1	B	...
B	1	0	B	...
0	1	1	B	...

$\longrightarrow$

1	0	1	B	...
B	1	0	B	...
B	0	1	B	...

The remainder is 1, so increment the value of 2nd track by 1.

1	0	1	B	...
B	1	1	B	...
1	0	1	B	...

$\longrightarrow$

1	0	1	B	...
B	1	1	B	...
0	1	0	B	...

The remainder is 2, so increment the value of 2nd track

1	0	1	B	...
1	0	0	B	...
1	0	1	B	...

→

1	0	1	B	...
1	0	0	B	...
0	0	1	B	...

The remainder is 1, so increment value of 2nd track by 1.

1	0	1	B	...
1	0	1	B	...
1	0	1	B	...

Now the value of first & second track is equal, so the number 5 is a prime number.

Example 2: Input string = '7'.

1	1	1	B	...
B	1	0	B	...
1	1	1	B	...

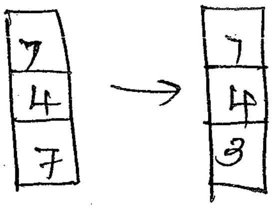
Subtracting 2 from 7, we get.

$$\begin{array}{|c|}\hline 7\\\hline 2\\\hline 7\\\hline\end{array} \rightarrow \begin{array}{|c|}\hline 7\\\hline 2\\\hline 5\\\hline\end{array} \rightarrow \begin{array}{|c|}\hline 7\\\hline 2\\\hline 3\\\hline\end{array} \rightarrow \begin{array}{|c|}\hline 7\\\hline 2\\\hline 1\\\hline\end{array}$$

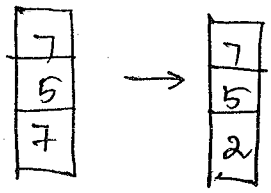
The remainder is 1, so increment the value of 2nd track by 1

$$\begin{array}{|c|}\hline 7\\\hline 3\\\hline 7\\\hline\end{array} \rightarrow \begin{array}{|c|}\hline 7\\\hline 3\\\hline 4\\\hline\end{array} \rightarrow \begin{array}{|c|}\hline 7\\\hline 3\\\hline 1\\\hline\end{array}$$

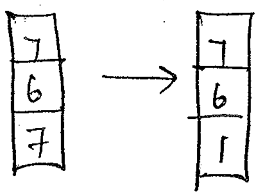
The remainder is 1, so increment the value of 2nd track by 1.



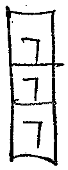
Remainder is 3, so increment value of 2^{nd} track by 1



Remainder is 2, so increment the value of 2^{nd} track by 1

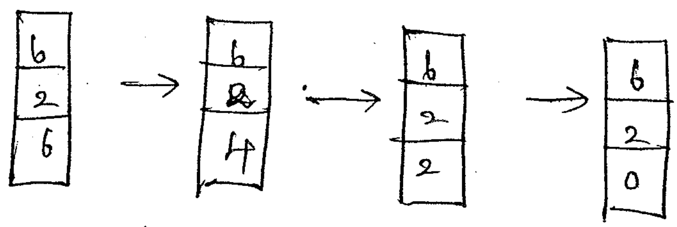


Remainder is 1, so increment value of 2^{nd} track by 1.



Now the value of 1^{st} & 2^{nd} track is equal, so the no. 7 is a prime number.

<u>Example 3:</u> I/P string = 6



on dividing 2 from 6, we get 4 thus by subtracting 4 by 2, we get 2 and again by subtracting 2 by 2 we get 0, since the remainder is 0, the number 6 is not a prime number.

PROBLEMS :

Qn : Build a multitrack turing machine for checking whether given number is prime or not ?

Soln : Here we can build a two track TM. We will consider the input $\Sigma = \{0, 1\}$ i.e a binary input string. Let n be the number to be checked.

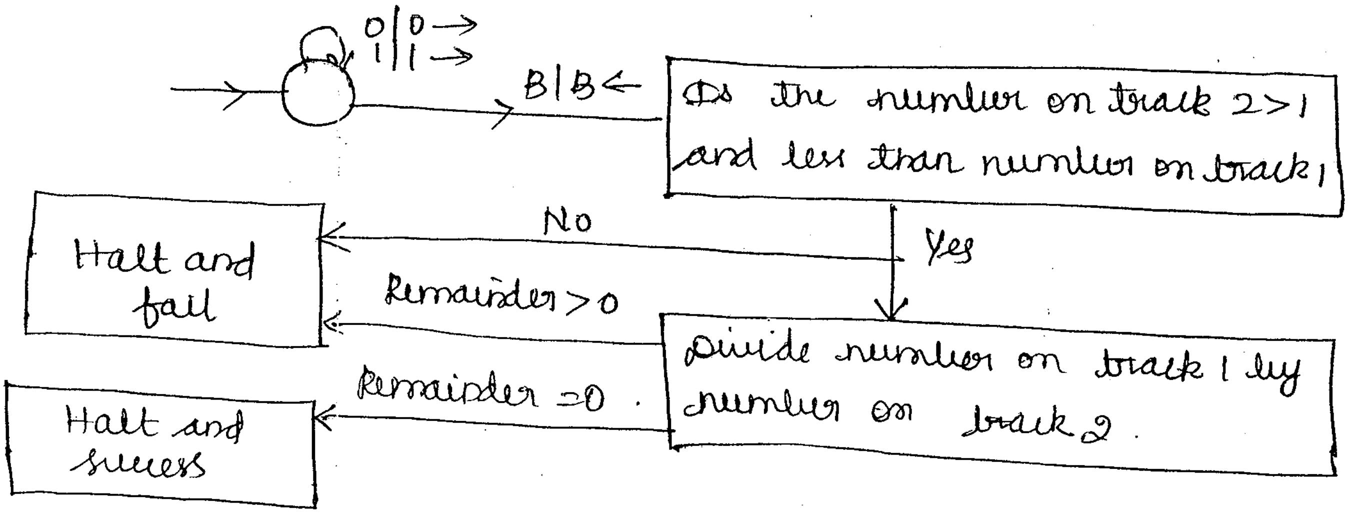
(1) We will guess a number m, where $1 < m < n$.

(2) Divide n by m.

(3) If there is 0 remainder then it halts and succeed.

(4) Otherwise it halts & fails.

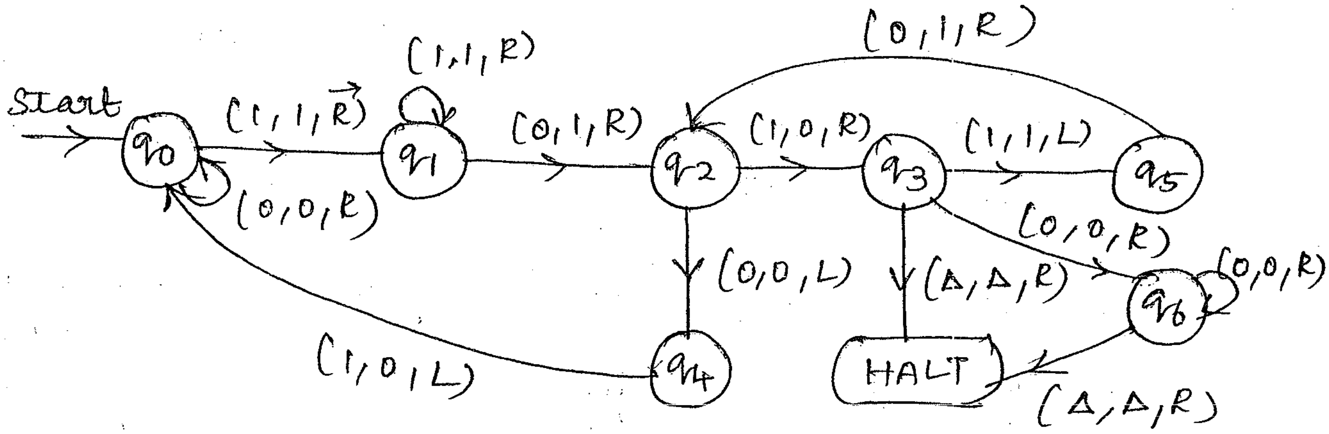
It can be modelled as,



Qn : Design a TM which recognise the i/p language having a substring as 101 & replaces every occurence of 101 by 110.

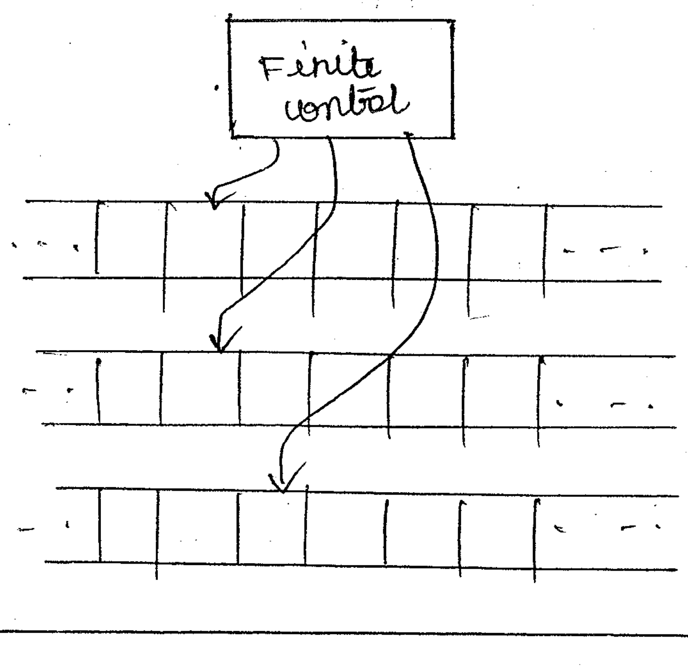
Soln : Replacement of any symbol by some another one means after reading of that specific symbol we should print the replacement symbol.

* In this case 101 has to be replaced by 110. Then TM has to be constructed considering 101 as a substring and leaving 110 substring after complete scan of the input.



Multitape Turing Machine:

The multitape TM has a finite control state and some finite number of tapes. Each tape in the multiple (or) multitape TM is divided into cells and each cell can hold any symbol & the multitape TM is shown below,



The multitape has the following,

(1) The i/p which is the finite sequence of i/p symbols and is placed on the 1st tape.

(2) All the other cells of all the tapes hold the blank symbols.

(3) The finite control is in the initial state.

(4) The head of the first tape is at the left end of the input.

(5) All the other tape head will be at some arbitrary cell.

Since the tapes other than 1st tape are completely blanks, there is no need to see where the head is placed initially and all the cells of those tapes look the same. A move of the multitape TM depends on the following.

(1) State of the finite control

(2) Symbol scanned by each tape head.

In a single move, the multitape TM does the following,

(1) The finite control enters a new state.

(2) On each tape, a new tape symbol is written on the cell scanned.

(3) Each of the tape head makes a move, which can be either left, right or stationary.

(4) The heads move independently, so different heads may move in different directions and some heads may

not at all move.

Checking off symbols:

The TM can be extended by using checking off symbols. This method is used by the TM for the languages that contains the repeated string, and to compare the length of the two substrings.

The examples languages are,

$$L = \{ wcw \mid w = \{a, b\}^* \}$$

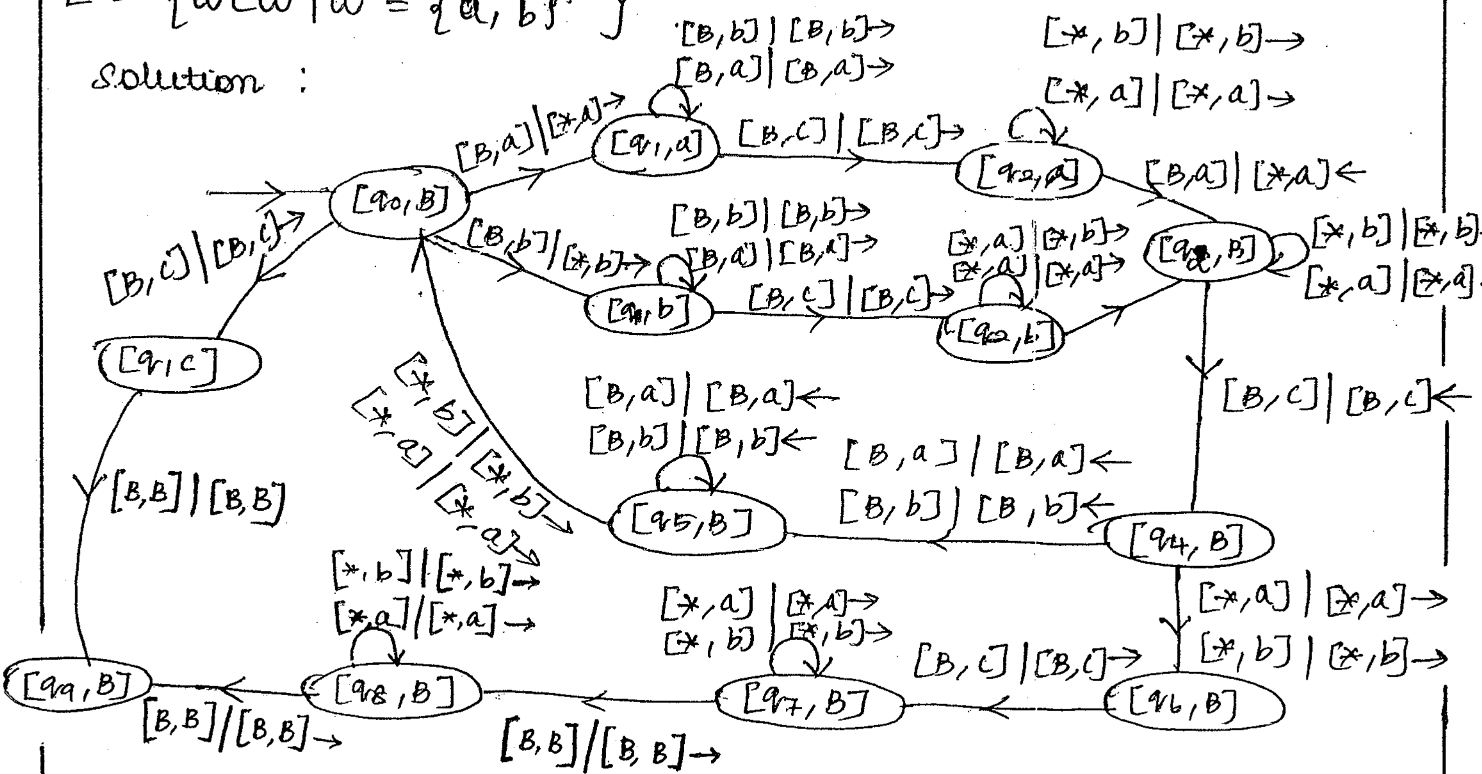
$$L = \{ ww \mid w = \{0, 1\}^* \}$$

$$L = \{ ww^R \mid w = \{0, 1\}^* \}$$

PROBLEM:

1) Design a turing machine to recognize the language $L = \{wcw \mid w = \{a, b\}^* \}$

Solution:



Subroutines :

* There are some problems, in which some tasks need to be performed repeatedly and it can be done by subroutines. The subroutines are also called as function. The subroutine in the turing machine is a set of state that specifically performs some tasks.

→ The set of states in the subroutine has one start state and another state namely the return state.

→ The return state of the subroutine doesnot have moves and it pass the control to other set of states of the turing machine that calls the subroutine.

→ The subroutine is called whenever there is a transition to its initial state.

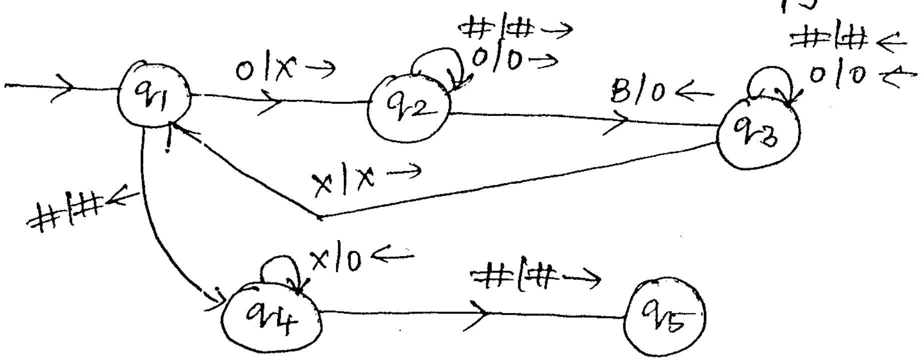
→ The calls are made to the start state of different copies of the subroutine and each copy returns to a different state.

→ The subroutines of the TM perform some task simultaneously.

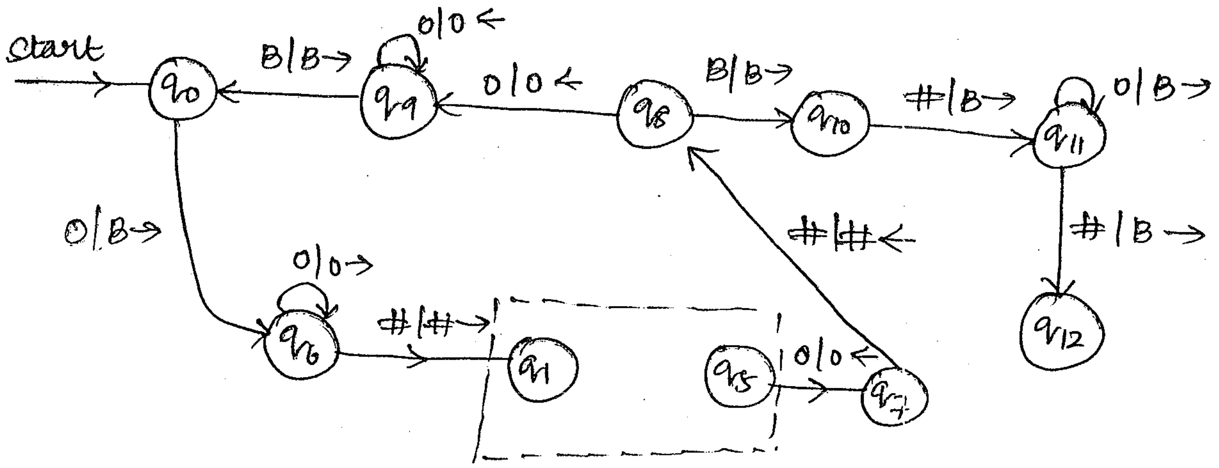
PROBLEM :

Design a TM to perform the multiplication function $f(m,n) = m * n$ using subroutine.

Soln: Transition diagram for subroutine copy.



Transition diagram for main program.



The complete multiplication program uses the subroutine copy.

Transition table for subroutine copy program.

State	0	#	X	B
$\to q_1$	(q_2, X, R)	$(q_4, \#, L)$	—	—
q_2	$(q_2, 0, R)$	$(q_2, \#, R)$	—	$(q_3, 0, L)$
q_3	$(q_3, 0, L)$	$(q_3, \#, L)$	(q_1, X, R)	—
q_4	(—	$(q_5, \#, R)$	$(q_4, 0, L)$	—

Transition table for main program.

State	0	#	B
$\to q_0$	(q_6, B, R)	—	—
q_5	$(q_7, 0, L)$	—	—
q_6	$(q_6, 0, R)$	$(q_1, \#, R)$	—
q_7	—	$(q_8, \#, L)$	—
q_8	$(q_9, 0, L)$	—	(q_{10}, B, R)
q_9	$(q_9, 0, L)$	—	(q_9, B, R)
q_{10}	—	(q_{11}, B, R)	(q_7, B, R)
q_{11}	(q_{11}, B, R)	(q_{12}, B, R)	—
* q_{12}	—	—	—

Non - Deterministic Turing Machine [NTM]

* Non determinism is a powerful feature of TM.

* These NTM machines are easy to design and are equivalent to deterministic TM.

* A NTM accepts a string, w if there exists a least one sequence of moves from the initial state to final state.

Definition :

A NTM is defined as,

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

where $Q \rightarrow$ set of states including initial, having rejecting states.

$\Sigma \rightarrow$ finite set of input alphabets.

$\Gamma \rightarrow$ finite set of tape symbols.

$\delta \rightarrow$ Transition function defined by

$$\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{L, R, N\})$$

where $P \rightarrow$ powerset.

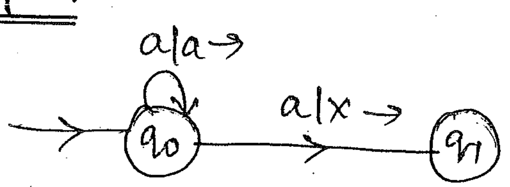
$q_0 \rightarrow$ initial state

$B \rightarrow$ blank symbol

$F \rightarrow$ set of final states $(F \subseteq Q)$

The transition function δ takes on the states tape symbols and head movement.

Example :



The above transition takes on two paths for the same input a. The transition of 'a' at q0 is defined as

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

THE HALTING PROBLEM :

* The Halting problem is the problem of finding if the program & machine halts or loop forever.
* The halting problem is undecidable over TM.

Description :

* Consider the TM M and a given string w, the problem is to determine whether M halts by either accepting (or) rejecting w or run forever.

Example :

```
while (1)
  {
   Printf ("Halting problem");
  }
```

* The above code goes to an infinite loop since the argument of while loop is true forever.
* Thus it doesn't halts.
* Hence Turing problem is the example for undecidabilit
* This concept of solving the halting problem being proved as undecidable was done by Turing in 1936.

* The undecidability can be proved by reduction techniques.

* Representation of the Halting set :

The halting set is represented as,

$$h(M, w) = \begin{cases} 1 & \text{if } M \text{ halts on input } w \\ 0 & \text{otherwise.} \end{cases}$$

where $M \rightarrow$ TM

$w \rightarrow$ I/p string

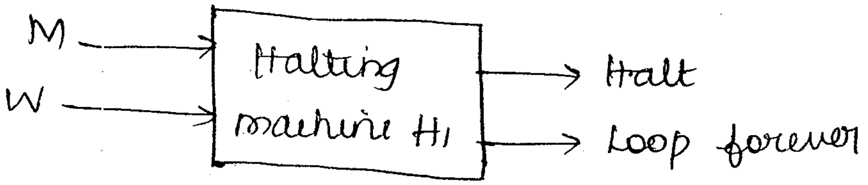
Theorem : Halting problem of TM is unsolvable / undecidable.

Proof :

* The theorem is proved by the method of proof by contradiction.

* Let us assume that TM is solvable/decidable

Construction of H_1

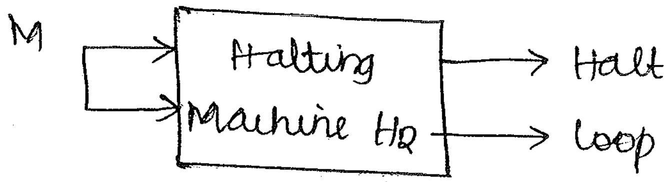


* Consider a string describing M and i/p string w for M.

* Let H1 generates "halt", if H_1 determines that the turing machine, M stops after accepting the i/p w.

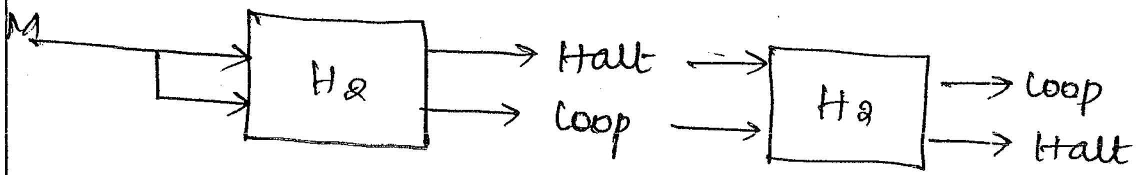
* Otherwise H1 loops forever when, M doesn't stop on processing w.

Construction of H_2.

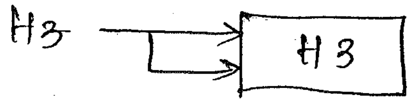


- H_2 is constructed with both the i/ps being M.
- H_2 determines M and halts if M halts otherwise loops forever.

Construction of H_3:



- Let H_3 be constructed from the outputs of H_2.
- If the outputs of H_2 are halt, the H_3 loops forever
- else if the o/p of H_2 is loop forever than H_3 halts
- Thus H_3 acts constructor to that of H_2.



- Let the output of H_3 be given as input to itself
- If the i/p is loop forever, then H_3 acts contradictory to it, hence halts.
- And if the i/p is halt, then H_3 loops by the construction
- Since the result is incorrect in both the cases, H_3 doesn't exist.
- Thus H_2 doesn't exist because of H_3.

- Similarly H_1, doesn't exist, because of H_2.
- Thus Halting problem is undecidable.

Partial solvability:

Problem Types,

There are basically three types of problems namely,

* Decidable / solvable / Recursive
* Undecidable / unsolvable
* Semidecidable / partial solvable / Recursively enumerable.

Decidable / solvable problems:

* A problem, P is said to be decidable if there exists a turing machine, TM that decides P.
* Thus P is said to be recursive.
* Consider a TM, M that halts with either "yes" or "no" after computing the input.

$$w \in \Sigma^* \longrightarrow \boxed{M} \begin{array}{l} \rightarrow \text{Yes (if } w \in L) \\ \rightarrow \text{No (if } w \notin L) \end{array}$$

* The machine finally terminates after processing.
* It is given by the function.

$$F_p(w) = \begin{cases} 1 & \text{if } p(w) \\ 0 & \text{if } \neg p(w) \end{cases}$$

* The machine that applies $F_p(w)$ is said to be turing computable.

Undecidable problem:

A problem, P is said to be undecidable, if there is a TM, TM that doesn't decide P.

Semidecidable / partial solvable / Recursively enumerable

* A problem, P is said to be semi-decidable, if P is recursively enumerable.

* A problem is RE if M terminates with "yes" if it accepts $w \in L$; and doesn't halt if $w \notin L$.

* Then the problem is said to be partial solvable (or) Turing acceptable.

$$w \in \Sigma^* \longrightarrow \boxed{M} \begin{cases} \rightarrow \text{yes (if } w \in L) \\ \rightarrow \text{loop forever (if } w \notin L) \end{cases}$$

* Partial solvability of machine is defined as

$$F_p(w) = \begin{cases} 1 & \text{if } P(w) \\ \text{undefined} & \text{if } \neg P(w). \end{cases}$$

Properties:

The semi-decidable / RE language are closed under

(1) union

(2) Intersection

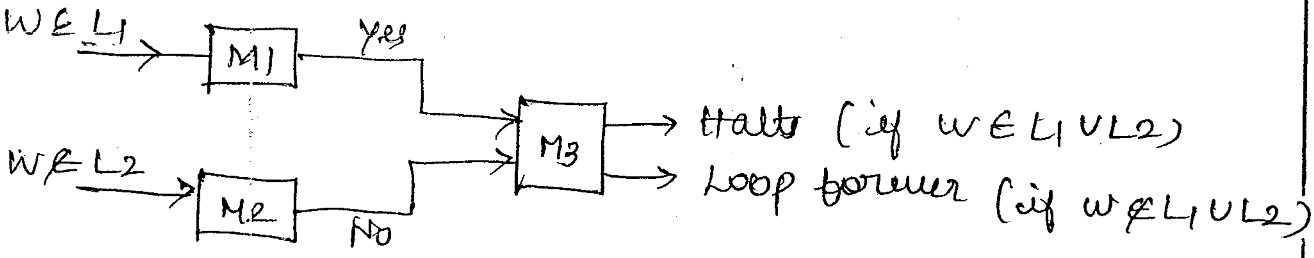
(3) But not under complementation.

Closure under union:

- Let L_1 & L_2 be two RE language.
- And consider M_1 that is a semi acceptor for L_1

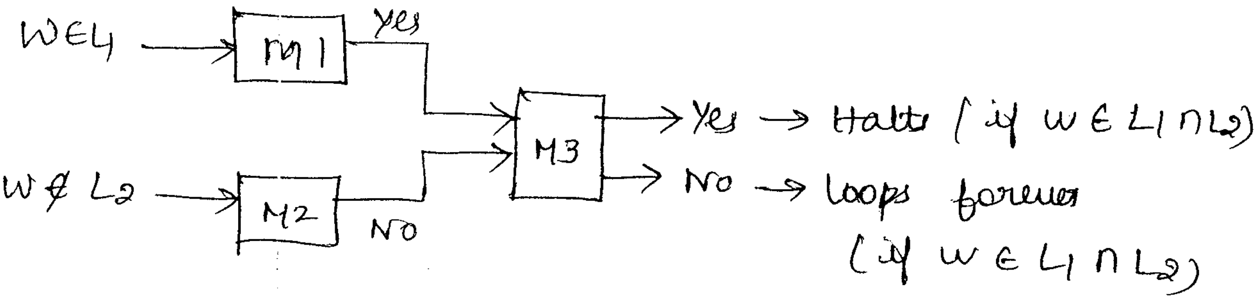
L_1 and M_2 be a TM for L_2.

- Let 'w' $\in L_1$ but not in L_2. Then $w \in L_1 \cup L_2$ and eventually M_3 halts if M_3 takes on both M_1 and M_2 and halts if any of them halts.
- If $w \notin L_1$ & $w \notin L_2$ then $w \notin L_1 \cup L_2$, which causes M_3 to loop forever.



Closed under Intersection :

- Let L_1 and L_2 be two RE languages accepted by M_1 and M_2 respectively.
- Let $w \in L_1$ and $w \in L_2$ be the input string.
- The TM, M_3 is constructed that takes on M_1 & M_2 and halts if both M_1 and M_2 halts.
- If $w \in L_1 \cap L_2$, M_3 halts with "yes".
- Else M_3 loop forever.



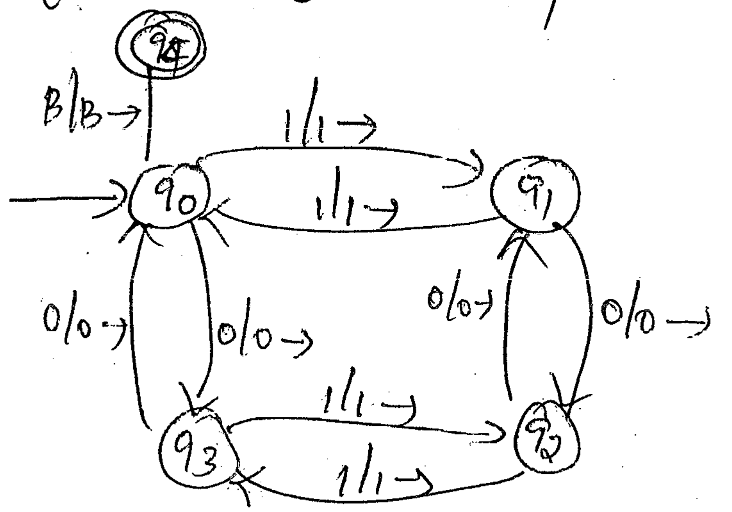
Note :

The next example of partial solvability is the halting problem ; acceptance problem.

CHOMSKY HIERARCHY OF LANGUAGES :

Refer UNIT - II.

Design a TM to accept the string with even number of 0's & 1's over the alphabet 0 & 1.



Design a TM with not more than three states that accepts the language $a(a+b)^*$. Assume $\Sigma = \{a, b\}$.

Soln Let Regular expression $= a(a+b)^*$ The corresponding TM will be