

UNIT - 4

Normal forms for Context free Grammars:

Eliminating useless symbols, Eliminating ϵ -production
Chomsky normal form, Greibach Normal form.

Normal forms

→ Every context free grammar that doesn't generate the empty string can be transformed into an equivalent one in Chomsky normal form or Greibach normal form. "Equivalent" here means that the two grammars generate the same language.

— A context free grammar $G = (V, \Sigma, R, S)$ is in normal form if and only if R contains no rules of the form.

1. $A \rightarrow \epsilon$, for any $A \in V$ or

2. $A \rightarrow B$, for any $A, B \in V$

Here the procedure for converting a grammar G into a grammar G' such that G' is in normal form and $L(G') = L(G) - \{\epsilon\}$ through

out the procedure, A and B are the arbitrary elements of V and u and v are arbitrary strings in $(V \cup \Sigma)^*$

* When the production in the grammar satisfy certain restrictions then that grammar is said to be in a "Normal form"

Minimization of CFG

Various languages can efficiently be represented by CFG. But all the grammar are not always optimized that means the grammar may consist of some extra symbols. (not terminal)

→ Having extra symbols ^{will} unnecessarily increase the length of grammar. Simplification of grammar means reduction of grammar by removing useless symbols.

The properties of reduced grammar are

1. Each variable (i.e. non terminal) and each terminal of G appears in the derivation of some word in L .
2. There shouldn't be any production as $x \rightarrow y$ where x and y are non terminals
3. If ϵ is not in the language 'i' then there need not to be the production $x \rightarrow \epsilon$

→ A Grammar can be reduced by

- ① Removal of useless symbols
2. Elimination of ϵ production
3. Removal of unit production.

Removal of useless symbols

→ A symbol can be useless if it doesn't appear on the right hand side of the production rule and doesn't take part in the derivation of any string. That symbol is known as useless symbol.

→ Variable can be useless if it doesn't take part in the derivation of any string, then that variable is also known as useless variable.

Example:—

$$T \rightarrow aaB \mid abA \mid aat$$

$$A \rightarrow aA$$

$$B \rightarrow ab \mid b$$

$$C \rightarrow ad$$

⇒

$$\boxed{\begin{array}{l} T \rightarrow aaB \mid aat \\ B \rightarrow ab \mid b \end{array}}$$

1. The variable 'c' will never occur in the derivation of any string and hence $C \rightarrow ad$ is useless. So we can eliminate it and the other productions are written in such a way that variable C can never reach from the starting variable 'T'.
2. $A \rightarrow aA$ is also useless because there is no way to terminate it. If it never terminates then it can never produce a string. Hence this production can never

take part in any derivation. To remove this useless production $A \rightarrow aA$, we will first find all the variables which will never lead to a terminal string such as variable 'A'. Then we will remove all the productions in which the variable ~~A~~ occurs.

Elimination of ϵ productions.

The productions of type $S \rightarrow \epsilon$ are called ϵ productions. These type of productions can only be removed from those grammars that do not generate ϵ .

Step 1: First find out all nullable non-terminal variable which derives ϵ .

Step 2: For each production $A \rightarrow a$ construct all productions $A \rightarrow x$ where x is obtained from a by removing one or more non-terminal from step 1.

Step 3: Now combine the result of step 2 with the original production and remove ϵ -productions.

Example:-

Remove the ϵ production from the following CFG by preserving the meaning of it

$$S \rightarrow XY, \quad X \rightarrow 0X | \epsilon, \quad Y \rightarrow 1Y | \epsilon$$

Sol

Now while removing ϵ production, we are deleting the rule $x \rightarrow \epsilon$ and $y \rightarrow \epsilon$. To preserve the meaning of CFG we are actually placing ϵ at the right-hand side whenever x and y have appeared

Let us consider

$$S \rightarrow XY$$

if the first x at R.H.S is ϵ then

$$S \rightarrow Y$$

if the ^{last} ~~second~~ x in R.H.S is ϵ then

$$S \rightarrow XY$$

if $y = \epsilon$ then $S \rightarrow X$

if both x and y are ϵ then $S \rightarrow \epsilon$

if both x are replaced by ϵ then $S \rightarrow Y$

Now

$$S \rightarrow YX | XY | XX | Y | X$$

Now let us consider

$$X \rightarrow 0X$$

if we place ϵ at right hand side for x then

$$x \rightarrow \epsilon$$

So we can write $x \rightarrow \epsilon x \mid \epsilon$

Similarly Consider $y \rightarrow \epsilon y \mid \epsilon$ can be written as.

$$y \rightarrow \epsilon y \mid \epsilon$$

Collective now we can write the productions

as

$$S \rightarrow xy \mid yx \mid xx \mid x \mid y \quad x \rightarrow \epsilon x \mid \epsilon \quad y \rightarrow \epsilon y \mid \epsilon$$

Removing unit productions.

The unit productions are the productions in which one non-terminal gives another non-terminal. Use the following steps to remove unit productions.

Step 1: To remove $x \rightarrow y$, add production $x \rightarrow a$ to the grammar rule whenever $y \rightarrow a$ occurs in the grammar

Step 2: Now delete $x \rightarrow y$ from the grammar

Step 3: Repeat Step 1 and Step 2 until all unit productions are removed

$$S \rightarrow 0A \mid 1B \mid C$$

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

$$B \rightarrow 1 \mid A$$

$$C \rightarrow 01$$

Sol $S \rightarrow C$ is a unit production. But while removing $S \rightarrow C$ we have to consider what C gives, so we can add a rule to S .

$$S \rightarrow 0A \mid 1B \mid 01 \quad \because C \rightarrow 01$$

Similarly $B \rightarrow A$ is also a unit production so we can modify it as

$$B \rightarrow 1 \mid 0S \mid 00$$

Thus we can CFG without unit productions

as
$$S \rightarrow 0A \mid 1B \mid 01$$

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

$$B \rightarrow 1 \mid 0S \mid 00$$

$$C \rightarrow 01$$

Chomsky's Normal Form (CNF)

In Chomsky Normal Form if all production rules satisfy one of the following conditions

- start symbol generating ϵ . For eg. $A \rightarrow \epsilon$
- A non-terminal generating two non-terminals
for example $S \rightarrow AB$
- A non-terminal generating a terminal. For

example $A \rightarrow a$.

~~$A \rightarrow a$~~ ~~$A \rightarrow BC$~~

A CFG is Chomsky NF if the productions are in following forms.

$$A \rightarrow a$$

$$A \rightarrow BC$$

where A, B and C are non-terminal and a is a terminal.

Steps to convert a given CFG to CNF

- Step 1. If the start S occurs on some right side create a new start symbol ' S' ' and a new production $S' \rightarrow S$
- Step 2. Remove null productions.
- Step 3. Remove unit productions.
- Step 4. Replace each production $A \rightarrow B_1 \dots B_n$ where $n > 2$ with $A \rightarrow B_1 C$ when $C \rightarrow B_2 \dots B_n$. Repeat this step for all productions having two or more symbols on the right side.
- Step 5. If the right side of any production is in the form $A \rightarrow aB$ ~~where~~ where ' a ' is a terminal and A , ~~and~~ B are non terminals then the production is replaced by $A \rightarrow XB$ and $X \rightarrow a$. Repeat this step for every production which is of the form $A \rightarrow aB$

Convert the following CFG to CNF:

$P: S \rightarrow ASA | AB, A \rightarrow B | S | \epsilon, B \rightarrow b | \epsilon$

1. Since 'S' appears in R.H.S we add a new state S' and $S' \rightarrow S$ is added to the production.

$P: S' \rightarrow S, S \rightarrow ASA | AB, A \rightarrow B | S | \epsilon, B \rightarrow b | \epsilon$

2. Remove the Null productions: $B \rightarrow \epsilon$ and $A \rightarrow \epsilon$.

After removing $B \rightarrow \epsilon$:

$P: S' \rightarrow S, S \rightarrow ASA | AB | a | AS | SA, A \rightarrow B | S, B \rightarrow b$
 $B \rightarrow \epsilon$ is leading to $A \rightarrow \epsilon$ so remove $A \rightarrow \epsilon$

After removing $A \rightarrow \epsilon$:

$P: S' \rightarrow S, S \rightarrow ASA | AB | a | AS | SA, A \rightarrow B | S, B \rightarrow b$

3. Remove the unit productions: $S \rightarrow S, S' \rightarrow S, A \rightarrow B$ and $A \rightarrow S$.

After removing $S' \rightarrow S$:

$P: S' \rightarrow S, S \rightarrow ASA | AB | a | AS | SA, A \rightarrow B | S, B \rightarrow b$

After removing $S' \rightarrow S$:

$P: S' \rightarrow ASA | AB | a | AS | SA \because S \rightarrow ASA | AB | a | AS | SA$

$S \rightarrow ASA | AB | a | AS | SA$

$$A \rightarrow B|s, B \rightarrow b.$$

After removing $A \rightarrow B$

$$P: s' \rightarrow ASA | ab | a | As | SA$$

$$s \rightarrow ASA | ab | a | As | SA$$

$$A \rightarrow b|s, B \rightarrow b$$

After removing $A \rightarrow s$:

$$P: s' \rightarrow ASA | ab | a | As | SA$$

$$s \rightarrow ASA | ab | a | As | SA$$

$$A \rightarrow b | ASA | ab | a | As | SA$$

$$B \rightarrow b$$

4 find out the productions that has more than 2 variables in R.H.S

$$s' \rightarrow ASA, s \rightarrow ASA, A \rightarrow ASA$$

After removing these we get.

$$s' \rightarrow Ax | ab | a | As | SA$$

$$s \rightarrow Ax | ab | a | As | SA$$

$$A \rightarrow b | Ax | ab | a | As | SA$$

$$B \rightarrow b$$

$$X \rightarrow SA.$$

SA is replaced with X

Now change the production

$$s' \rightarrow aB, s \rightarrow aB \text{ and } A \rightarrow aB$$

finally we get

$$s' \rightarrow Ax | yB | a | As | SA$$

$$s \rightarrow Ax | yB | a | As | SA$$

$$A \rightarrow b | Ax | yB | a | As | SA$$

$$B \rightarrow b$$

$$x \rightarrow SA$$

$$y \rightarrow a$$

which required CNF for the given CFG

R.H.S
not a terminal
not allowed.

Replace a with y

Greibach Normal form

A CFG is in GNF if the production are in the following forms

$$A \rightarrow b$$

$$A \rightarrow bc_1c_2 \dots c_n$$

where $A, c_1, \dots, c_n$ are Non-terminals and b is a terminal

Steps to convert a given CFG to GNF

Step 1: Check if the given CFG has any unit productions or Null productions and remove if there are any

Step 2:- Check whether the CFG is already in CNF and convert it to CNF if it is not

Step 3:- Change the names of the non-terminal symbols into A_i in ascending order of i

Consider CFG

$$S \rightarrow CA|BB$$

$$B \rightarrow b|SB$$

$$C \rightarrow b$$

$$A \rightarrow a$$

sol : no unit or null productions. and it is already in CNF

Replace the non terminals into some A_i

So S is replaced with A_1

C " " " A_2

D " " " A_3

B " " " A_4

order has to be maintained

Now we get

$$A_1 \rightarrow A_2 A_3 \mid A_4 A_4$$

$$A_4 \rightarrow b \mid A_1 A_4$$

$$A_2 \rightarrow b, A_3 \rightarrow a$$

Step 4: ~~After~~ Alter the rules so that the non-terminals are in ascending order, such that if the production is of the form $A_i \rightarrow A_j X$, then $i < j$ and should never be $i \geq j$

$$A_1 \rightarrow A_2 A_3 \mid A_4 A_4$$

$$i=1, j=2 \text{ or } 4$$

So the production is correct.

$$A_2 \rightarrow b, A_3 \rightarrow a \text{ both are correct.}$$

$$A_4 \rightarrow b$$

$$\text{Consider } A_4 \rightarrow A_1 A_4$$

$i=4; j=1$ which doesn't follow rule.

So A_1 can be replaced by $A_2 A_3 + A_4 A_4$

$$A_4 \rightarrow b \mid A_1 A_4$$

$$A_4 \rightarrow b \mid A_2 A_3 A_4 \mid A_4 A_4 A_4$$

$i=4, j=2 \text{ or } 4$ which doesn't follow rule.

$$A_1 \rightarrow A_2 A_3 \mid A_4 A_4$$

$$A_4 \rightarrow b \mid A_1 A_4$$

$$A_2 \rightarrow b, A_3 \rightarrow a.$$

So replace A_2 with b in the above product
 $A_4 \rightarrow b \mid b A_3 A_4 \mid A_4 A_4 A_4$

$$A_4 \rightarrow b \mid b A_3 A_4 \mid A_4 A_4 A_4$$

$i = j = 4$
 leads to left recursion

Steps: Remove left recursion.

Introduce a new variable. to remove the left recursion

$$A_4 \rightarrow b \mid b A_3 A_4 \mid \underline{A_4 A_4 A_4}$$

problematic variable.

$$Z \rightarrow A_4 A_4 Z \mid A_4 A_4 \quad (\text{new production})$$

Substitute Z into A_4 production leaving $(A_4 A_4 A_4)$

$$A_4 \rightarrow b \mid b A_3 A_4 \mid \underline{b Z} \mid \underline{b A_3 A_4 Z}$$

once combination with b .

Combination with 2nd production.

CNF.
 $A \rightarrow a$
 $A \rightarrow BA$

Now the grammar is

$$A_1 \rightarrow \textcircled{A_2} A_3 \mid \textcircled{A_4} A_4$$

$$A_4 \rightarrow b \mid b A_3 A_4 \mid b Z \mid b A_3 A_4 Z$$

$$Z \rightarrow \textcircled{A_4} A_4 \mid \textcircled{A_4} A_4 Z$$

$$A_2 \rightarrow b$$

$$A_3 \rightarrow a$$

CFGNF
 $A \rightarrow b$
 $A \rightarrow aB, B_2, \dots$
 $A \rightarrow a(V^*)$

according CNF

$A \rightarrow aB$

$$A_1 \rightarrow b A_3 \mid b A_4 \mid b A_3 A_4 A_4 \mid b Z A_4 \mid b A_3 A_4 Z A_4$$

$$A_4 \rightarrow b \mid b A_3 A_4 \mid b Z \mid b A_3 A_4 Z$$

$$Z \rightarrow b A_4 \mid b A_3 A_4 A_4 \mid b Z A_4 \mid b A_3 A_4 Z A_4 \mid b A_4 Z \mid b A_3 A_4 A_4 Z$$

$$b Z A_4 Z \mid b A_3 A_4 Z A_4 Z ; A_2 \rightarrow b ; A_3 \rightarrow a$$

7-9

$$A_2 \rightarrow b$$

$$A_3 \rightarrow a$$

Cases of $i \& j$

if $i < j$ - case (1)

if $A_i \rightarrow A_j B$ & $i < j$
leave the production as it is

Case (2) - if $i > j$

if $A_i \rightarrow A_j \beta$ & $i > j$

Replace A_j with its production bodies, such that

$$A_j \rightarrow \alpha_1 | \alpha_2 | \dots | \alpha_n$$

$$A_i \rightarrow \alpha_1 \beta | \alpha_2 \beta | \dots | \alpha_n \beta$$

Case (3) $\rightarrow i = j$

if $A_i \rightarrow A_j \beta | \alpha$ & $i = j$ (eliminate left recursion
& ϵ -productions)

Introduce new variable B .

$$B \rightarrow \beta | \beta B$$

$$A_i \rightarrow \alpha | \alpha \beta$$

3. Remove unit productions.

$$S' \rightarrow S, S \rightarrow S, A \rightarrow B, A \rightarrow S$$

Removing $S \rightarrow S$

$$S' \rightarrow S, S \rightarrow ASA|ab|a|AS|SA, A \rightarrow B|S, B \rightarrow b.$$

Removing $S' \rightarrow S$

$$S' \rightarrow ASA|ab|a|AS|SA$$

$$S \rightarrow ASA|ab|a|AS|SA$$

$$A \rightarrow B|S$$

$$B \rightarrow b.$$

Removing $A \rightarrow B$.

$$S' \rightarrow ASA|ab|a|AS|SA$$

$$S \rightarrow ASA|ab|a|AS|SA$$

$$A \rightarrow b|S \quad \because B \rightarrow b.$$

$$B \rightarrow b.$$

Removing $A \rightarrow S$.

$$S' \rightarrow \underline{ASA}|ab|a|AS|SA$$

$$S \rightarrow \underline{ASA}|ab|a|AS|SA$$

$$A \rightarrow b|\underline{ASA}|ab|a|AS|SA.$$

$$\text{A} \rightarrow b \quad B \rightarrow b.$$

4 (9.1)

$$P: S \rightarrow ASA | aB$$

$$A \rightarrow B | S | \epsilon$$

$$B \rightarrow b | \epsilon$$

preparation

1. S is appearing on R.H.S so new production

$$S' \rightarrow S, S \rightarrow ASA | aB, A \rightarrow B | S | \epsilon, B \rightarrow b | \epsilon$$

2. Remove null production for $A \rightarrow \epsilon$ & $B \rightarrow \epsilon$

$$\begin{array}{l} S' \rightarrow S \\ S \rightarrow SA | AS | S | aB \\ A \rightarrow B | S \end{array}$$

$B \rightarrow b$
 $S' \rightarrow S$
 $S \rightarrow ASA | aB | a$
 $A \rightarrow B | S | \epsilon$
 CFG $\rightarrow$ CNF

$$\begin{array}{l} S' \rightarrow S \\ S \rightarrow ASA | aB \\ A \rightarrow S \\ B \rightarrow b \end{array}$$

$$S' \rightarrow S, S \rightarrow ASA | aB | a$$

$$A \rightarrow B | S | \epsilon$$

$$B \rightarrow b$$

Now remove $A \rightarrow \epsilon$

$$S' \rightarrow S, S \rightarrow ASA | aB | a | SA | AS | S$$

$$A \rightarrow B | S$$

$$B \rightarrow b$$

Extra CNF

More than two variables in R.H.S.

$$S' \rightarrow ASA$$

$$S \rightarrow ASA$$

$$A \rightarrow ASA$$

After removing these we get.

$$S' \rightarrow AX | \cancel{B} | \cancel{A} | AS | SA$$

$$S \rightarrow AX | \cancel{B} | a | AS | SA$$

$$A \rightarrow b | AX | \cancel{B} | a | AS | SA$$

$$B \rightarrow b. \quad Y \rightarrow a.$$

$$X \rightarrow SA$$

CFG to GNF

$$S \rightarrow CA | BB$$

$$B \rightarrow b | SB$$

$$C \rightarrow b$$

$$A \rightarrow a$$

} Replace

S with A_1

C " A_2

A " A_3

B " A_4

$$A_1 \rightarrow A_2 A_3 | A_4 A_4$$

$$A_4 \rightarrow b | A_1 A_4 X$$

$$A_2 \rightarrow b.$$

$$A_3 \rightarrow a.$$

$$A_4 \rightarrow A_1 A_4 | b.$$

$$A_4 \rightarrow b | A_2 A_3 A_4 | A_4 A_4 A_4$$

$A_i \rightarrow A_j X$ never be
the $i < j$ and $i \geq j$

4 - (9.2)

Replace A_2 with other.

$$A \rightarrow b$$

$$A \rightarrow bC_1C_2 \dots C_n.$$

$$A_4 \rightarrow b \mid b A_3 A_4 \mid \underbrace{A_4 A_4 A_4}_{\text{leads left recursion.}}$$

Step 5: Remove left Recursion

$$A_4 \rightarrow b \mid b A_3 A_4 \mid A_4 A_4 A_4$$

→ creating new variable Z

no combination of Z .

$$Z \rightarrow A_4 A_4 \mid A_4 A_4 Z$$

Z combination with $A_4 A_4$

$$A_4 \rightarrow b \mid b A_3 A_4 \mid b Z \mid b A_3 A_4 Z$$

$$A_1 \rightarrow A_2 A_3 \mid A_4 A_4$$

$$A_4 \rightarrow b \mid b A_3 A_4 \mid b Z \mid b A_3 A_4 Z$$

$$Z \rightarrow A_4 A_4 \mid A_4 A_4 Z$$

$$A_2 \rightarrow b$$

$$A_3 \rightarrow a$$

$$A_1 \rightarrow b A_3 \mid b A_4 \mid b A_3 A_4 A_4 \mid b Z A_4 \mid b A_4 A_4 Z A_4$$

$$A_4 \rightarrow b \mid b A_3 A_4 \mid b Z \mid b A_3 A_4 Z$$

$$Z \rightarrow b A_4 \mid b A_3 A_4 A_4 \mid b Z A_4 \mid b A_3 A_4 Z A_4$$

$$\rightarrow b A_4 Z \mid b A_3 A_4 A_4 Z \mid b Z A_4 Z \mid b A_3 A_4 Z A_4 Z$$

$$A_2 \rightarrow b, A_3 \rightarrow a$$

Intermediate Code Generation

It is used to translate the source code into the machine code. It lies between high level lang and machine.

Example 2: Convert CFG to CNF

$S \rightarrow ABa$

$A \rightarrow aab$

$B \rightarrow Ac.$

CNF
 $A \rightarrow BC$
 $A \rightarrow a$

Sol let $\left. \begin{array}{l} x \rightarrow a \\ y \rightarrow b \\ z \rightarrow c \end{array} \right\}$

$S \rightarrow ABx$

$A \rightarrow xxb$

$B \rightarrow xc. \checkmark$

$x \rightarrow a. \checkmark$

$y \rightarrow b. \checkmark$

$z \rightarrow c. \checkmark$

$P \rightarrow AB \}$

$Q \rightarrow xx \}$

$S \rightarrow Px$

$A \rightarrow Qb$

$B \rightarrow xc$

$x \rightarrow a$

$y \rightarrow b$

$z \rightarrow c.$

$P \rightarrow AB$

$Q \rightarrow xx$

Example 3:

$S \rightarrow bA|aB$

$A \rightarrow bAA|as|a$

$B \rightarrow aBB|bs|b.$

let $x \rightarrow a. \checkmark$

$y \rightarrow b. \checkmark$

$S \rightarrow yA|xB. \checkmark$

$A \rightarrow yAA|xs|a. \checkmark$

$B \rightarrow xBB|ys|b.$

$P \rightarrow AA \quad Q \rightarrow BB.$

$S \rightarrow yA|xB$

$A \rightarrow yP|xs|a$

$B \rightarrow xQ|ys|b.$

$P \rightarrow AA$

$Q \rightarrow BB$

$x \rightarrow a$

$y \rightarrow b.$

Extra CNF
 $4 \rightarrow 9.3$

GNF Example 2

GNF

$$\begin{aligned} A &\rightarrow a \\ A &\rightarrow av^* \end{aligned}$$

①

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow aA | bB | b \\ B &\rightarrow b \end{aligned}$$

$$\begin{aligned} S &\rightarrow aAB | bBB | bB \\ A &\rightarrow aA | bB | b \\ B &\rightarrow b \end{aligned}$$

②

$$\begin{aligned} S &\rightarrow CA | BB \\ B &\rightarrow b | SB \\ C &\rightarrow b \\ A &\rightarrow a \end{aligned}$$

$$\Rightarrow S \rightarrow bA | \cancel{bB} | \cancel{BBB} \dots \because C \rightarrow b$$

$$B \rightarrow b | SB$$

$$C \rightarrow b$$

$$A \rightarrow a$$

$$\Rightarrow S \rightarrow bA | BB$$

$$B \rightarrow b | bAB | BBB \dots S \rightarrow bA | BB$$

$$C \rightarrow b$$

$$A \rightarrow a$$

left recursion.

$$\begin{aligned} A &\rightarrow A\alpha | \beta \\ A &\rightarrow \beta A' \\ A' &\rightarrow \epsilon | \alpha A' \end{aligned} \Rightarrow \beta \alpha^*$$

$$\begin{aligned} B &\rightarrow \underline{b} | \underline{bAB} | \underline{BBB} \\ A &\quad \beta_1 \quad \beta_2 \quad A \alpha \end{aligned}$$

$$B \rightarrow bB' | bABB'$$

$$B' \rightarrow \epsilon | BBB'$$

Removing null production.

$$B \rightarrow bB' | bABB' | b | bAB$$

$$B' \rightarrow BBB' | BB$$

$$S \rightarrow bA | BB$$

$$B \rightarrow b | bAB | bB' | bABB'$$

$$B' \rightarrow BBB' | BB$$

$$C \rightarrow b$$

$$A \rightarrow a$$

Combining new var
all
B' with b | bAB

$$S \rightarrow bA | bB | bABB | bBB' | bABB'$$

$$B \rightarrow b | bAB | bB' | bABB'$$

$$B' \rightarrow bBB' | bABB' | bB'BB'$$

$$bABB'BB' | bB | bABB |$$

$$bB'B | bABB'B$$

$$C \rightarrow b$$

$$A \rightarrow a$$

Extra GNF
Example.

Removing useless symbols.
(non terminals)

$$T \rightarrow aaB | abA | T$$

$$A \rightarrow aA \quad B \rightarrow ab | b \quad C \rightarrow ad$$

Sol $T \rightarrow aaB | abA | T$

Remove $C \rightarrow ad$ ($\because$ since it is useless product)

$$A \rightarrow aA$$

$$\rightarrow aaA$$

$$\rightarrow aaaA \quad \therefore \text{Remove } A \rightarrow aA$$

$$\begin{aligned} T &\rightarrow aaB | T \\ B &\rightarrow ab | b \end{aligned}$$

Example 2: $S \rightarrow AB | a$

$$A \rightarrow b$$

Remove $A \rightarrow b$.

$$B \rightarrow \text{AB} \quad B \text{ is not used.}$$

$\therefore$ $S \rightarrow a$ only CFG required.

Removing Null productions (or)

ϵ - productions.

Example: $S \rightarrow \cancel{a} x y x$, $a \rightarrow \epsilon$, $y \rightarrow \epsilon$

$$x y x \rightarrow y x$$

$$x y x \rightarrow x y$$

$$x y x \rightarrow y$$

$$x y x \rightarrow x x$$

$\downarrow$
 ϵ

$$(a) y x \rightarrow x$$

$$x y \rightarrow x$$

(e)

$$\therefore S \rightarrow x y x | y x | x y | y$$

$$x \rightarrow \epsilon$$

$$\therefore x \rightarrow \epsilon$$

$$y \rightarrow \epsilon$$

$$y \rightarrow \epsilon$$

$$\therefore S \rightarrow x y x | x x | x | y x | x y | y$$

Removing unit productions.

Generally

$$L.H.S \rightarrow R.H.S$$

non terminal

Terminal & non

$$\left. \begin{array}{l} L.H.S \rightarrow R.H.S \\ \text{Single non-terminal} \quad \text{Single non-terminal} \end{array} \right\} \text{unit production}$$

Example: $S \rightarrow \cancel{A} | B | C$

$$A \rightarrow \epsilon | \epsilon$$

$$B \rightarrow A$$

$$C \rightarrow \epsilon$$

$$S \rightarrow C$$

$$S \rightarrow \epsilon$$

$$S \rightarrow \cancel{A} | B | \epsilon$$

$$B \rightarrow A$$

$$B \rightarrow \epsilon | \epsilon$$

$$B \rightarrow \epsilon$$

show that $L = \{a^N b^N c^N \mid N \geq 0\}$ is not context free.

- Sol - Assume that L is context free
- L must have a pumping length (say p)
 - Now we take a string s such that $s = a^p b^p c^p$.
 - We divide string into $uvxyz$

Let $p=4$ so $s = a^4 b^4 c^4$.

Case 1: v and y each contains only one type of symbol

$$\begin{array}{ccccccc} aaaa & bbbb & cccc \\ | & | & | & | & | & | & | \\ u & v & & x & & y & z \end{array}$$

$$uv^k xy^k z \in L$$

Let $k=2$.

$$uv^2 xy^2 z \Rightarrow aaaaaabbbbbc ccc$$

$$\Rightarrow a^6 b^4 c^5 \notin L \quad (\because \text{condition was failed})$$

Case 2: Either v or y has more than one kind of symbols.

$$\begin{array}{ccccccc} aaaa & bbbb & cccc \\ | & | & | & | & | & | & | \\ u & v & x & y & & & z \end{array}$$

$$uv^k xy^k z \in L \Rightarrow \text{Let } k=2 \Rightarrow uv^2 xy^2 z.$$

$$\underbrace{aa}_u \underbrace{aabb}_{2} \underbrace{aabb}_{4} \underbrace{b}_{5} \underbrace{cccc}_y$$
 should be of $a^n b^n c^n$

so the string doesn't belong to language L is not context free language.

Pumping lemma for Context free languages:-

statement of pumping lemma, Applications

- pumping lemma for CFL is used to prove that a language is not a CFL

Assume L is CFL,

Then there is a pumping length n , such that any string $w \in L$ of length $\geq n$ can be written as follows

$$|w| \geq n$$

we can break w into 5 strings $w = uvxyz$, such as the ones given below

Note: select k such that the resulting string is not in L

$$- |vxy| \leq n$$

$$- |vy| \neq \epsilon$$

3 conditions

- For all $k \geq 0$, the string $uv^kxy^kz \in L$

The steps to prove that the language is not a context free by using pumping lemma are explained below (we prove using contradiction) $L \notin CFL$

- Assume that L is context free

- The pumping length is n

- All strings longer than n can be pumped

$$|w| \geq n$$

- Divide w into $UVXYZ$
- show that $UV^kXY^kZ \notin L$ for some k
- Then, consider the ways that w can be divided into $UVXYZ$.
- show that none of these can satisfy all the 3 pumping conditions at same time.
- w cannot be pumped (Contradiction)

Example: find out whether $L = \{x^n y^n z^n \mid n \geq 1\}$ is Context free or not

Sol Let L is Context free

$\Rightarrow$ now we can take a string such that $s = x^n y^n z^n$
we divide s into 5 parts $UVXYZ$.

Case (i), if $n=4$ so $s = x^4 y^4 z^4$

Let $V \& Y$ Each contains one type of symbols.

$\begin{array}{ccccccc} x & x & x & x & y & y & y & y & z & z & z & z \\ \downarrow & \downarrow & & & \downarrow & & & & \downarrow & \downarrow & & \\ u & v & & & x & & & & y & z & & \end{array}$ } dividing string into 5 parts ($UVXYZ$)

Applying formula UV^kXY^kZ (let $k=2$)

$3 \neq 4$
 $|VXY| \neq 0$

UV^2XY^2Z

$u = x$

$v = xx$

$x = xxyyzz$

$y = z$

$z = zz$

$|VXY| < n \Rightarrow 9 < 4$

$$x \text{ } x x x x \text{ } x \text{ } x y y y y z \text{ } z z \text{ } z z$$

$$x^6 y^4 z^5$$

no. of $x \neq y \neq z$.

So the resultant is not satisfying the condition

$$x^6 y^4 z^5 \notin L$$

Hence given language is not belongs to CFL.

Concatenation $\rightarrow$

Let L_1 and L_2 be two CFL's accepted by the

CFG's $G_1 = (V_1, \Sigma, P_1, s_1)$ and $G_2 = (V_2, \Sigma, P_2, s_2)$

respectively. Then $L_1 \cdot L_2$ will be accepted by the

CFG $G = (V, \Sigma, P, s)$ where

~~$V = V_1 \cup V_2 \cup \{s\}$ and~~

~~$P = P_1 \cup P_2 \cup \{s \rightarrow s_1 s_2\}$.~~

Example: Let $L_1 = \{a^n b^n \mid n > 0\}$ Grammar G_1

$P_1: s_1 \rightarrow a s_1 b \mid \epsilon$

$L_2 = \{c^m d^m \mid m > 0\}$

Grammar $G_2: P_2: s_2 \rightarrow c s_2 d \mid \epsilon$

Then Concatenation of L_1 & L_2 is L_3 languages

and grammar G_3 is

$L_1 \cdot L_2 = \{ \{a^n b^n\} \cdot \{c^m d^m\} \mid m, n > 0 \}$

$L_3 = \{ abcd, aabbccdd, \dots \}$

$P_3: s \rightarrow s_1 s_2$

$s_1 \rightarrow a s_1 b \mid ab; s_2 \rightarrow c s_2 d \mid cd.$

$VTPS = \{ \{s, s_1, s_2\}, \{c, d, a, b\} P, \{s\} \}$

$\therefore$ CFL is closed under concatenation

Kleen closure/star

If L is a CFL then L^* is also CFL

$$\text{ex: } L = \{a^n b^n \mid n > 0\}$$

$$G = P \quad S \rightarrow a S b \mid \epsilon$$

$$L^* = \{a^n b^n\}^*$$

$$G = S_1 \rightarrow S S_1 \mid \epsilon$$

$$S \rightarrow a S b \mid \epsilon$$

$\therefore$ CFL is closed under Kleen star

Intersection

If L_1 & L_2 are two CFL's then $L_1 \cap L_2$ is not CFL

$$L_1 = \{a^n b^n \mid n > 0\} \quad L_2 = \{b^m c^m \mid m > 0\}$$

$$L_3 = L_1 \cap L_2 = \{a^n b^n c^m \mid n > 0\}$$

$\therefore L_3$ is not CFL because L_3 language contains three variables comparison so CFL is not closed under intersection

Complementation: If the language L_1 is in CFL then the complement of that language is not CFL

Intersection with regular language.

If L_1 is regular language and L_2 is CFL
then the intersection of L_1 & L_2 is a CFL

$$L_3 = L_1 \cap L_2 = \text{is a CFL}$$

$\therefore$ CFL is closed under intersection with
Regular language.

Union : Let L_1 and L_2 be the two CFL

Then $L_1 \cup L_2$ is also a CFL

Ex: $L_1 = \{a^n b^n, n > 0\}$ $G_1: P: s_1 \rightarrow a s_1 b \mid \epsilon$

$L_2 = \{b^m c^m \mid m > 0\}$ grammar $G_2 = P: s_2 \rightarrow b s_2 c \mid \epsilon$

Then the language L_1 and G_1 gives the no. of
a's are equal to no. of b's and the pattern
is a's are followed by b symbols.

→ The language L_2 and Grammar G_2 it gives the
no. of b's are equal to no. of c's and the
pattern is b's must be followed by c's

$$L_1 \cup L_2 = L_3 = \{ \{a^n b^n\} \cup \{b^m c^m\}, m, n > 0 \}$$

$$L_3 = \{ab, aabb, \dots\}$$

$$L_3 = G_3 = s \rightarrow a s b \mid b s c \mid \epsilon$$

$$(V, T, P, S) = (\{s\}, \{a, b, c\}, P, S) \quad P: s \rightarrow a s b \mid b s c \mid \epsilon$$

Decision properties :-

To check whether CFL is empty or non empty, finite or infinite.

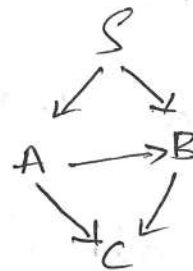
Test whether the following grammar is finite or not:

$$S \rightarrow AB$$

$$A \rightarrow BC/a$$

$$B \rightarrow CC/b$$

$$C \rightarrow a.$$



Consider only non-terminal to draw
Here no cycles are formed hence the
given grammar is finite

$$\text{Rank of } S \rightarrow 2$$

$$B \rightarrow 1$$

$$A \rightarrow 2$$

$$C \rightarrow 0$$

Testing Emptiness of CFL

Ex: $S \rightarrow XY$
 $X \rightarrow AX$
 $X \rightarrow AA$
 $X \rightarrow \Lambda$

$$A \rightarrow a$$
$$X \rightarrow BY$$
$$Y \rightarrow BB$$
$$B \rightarrow b.$$

Replace $B \rightarrow b$ & $A \rightarrow b$.

$S \rightarrow xy$

~~$S \rightarrow ax$~~

$x \rightarrow aa$

$y \rightarrow by$

$y \rightarrow bb$.

now replace $y \rightarrow bb$ & $x \rightarrow aa$.

$S \rightarrow aabb$. (non empty).

Membership

CYK algorithm : (membership problem).
is to find whether particular string is
accepted by grammar or not.

CYK — C — Cocke
— Y — Younger
— K — Kasami

Given CFG G & a string w

$G = (V, T, P, S)$

finite variables $\downarrow$ finite terminals

production start symbol

$V \neq \emptyset$ are assumed to be disjoint

It helps us to determine whether w is in $L(G)$ or not

→ CYK works only when the grammar is in CNF

Example: Show CYK with the following

$w = \text{bababa}$

$S \rightarrow AB \mid BC$

$A \rightarrow BA \mid a$

$B \rightarrow cc \mid b$

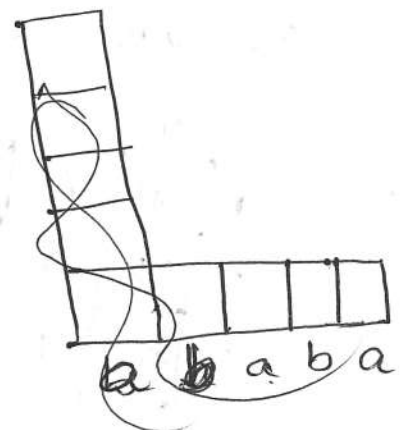
$C \rightarrow AB$

~~length is 5~~

$a \text{ b a b a}$

$S \rightarrow AB$
 $\rightarrow AB$
 $\rightarrow acc$
 $\rightarrow aAB$
 $\rightarrow a$

$\therefore A \rightarrow a$
 $\therefore B \rightarrow cc$
 $\therefore C \rightarrow AB$



Convert the grammar in CNF apply CKK.

Sol: $S \rightarrow AB$

$A \rightarrow BA \mid SA \mid b$

$B \rightarrow BB \mid BS \mid a$.

Let $w = ab$.

	a	b
a	B	A
b	A	

✗

$w = ba$

	b	a
b	A	B
a	S	

✓

	a	b	a
a	B	A	B
b	A	S	
a	B, S		

Example 2 Check whether a string 'abbb' is valid member of following

CFG

$S \rightarrow AB$; $A \rightarrow BB \mid a$ $B \rightarrow AB \mid b$

$S \rightarrow AB$
 $AA \times$
 $AB \times$
 $SB \times$
 SA
 A, B
 $AA \times$
 $AB \times$

Sol

	4	3	2	1
1	S, B	A	S, B	A
2	S, B	A	B	
3	A	B		
4	B			

abbb
1 2 3 4

Step 1) Consider $(1,1) \rightarrow a$ is leading A.
 " $(2,2) \rightarrow b$ is " B
 " $(3,3) \rightarrow b$ " " "
 " $(4,4) \rightarrow b$ " " B

Step 2:- Consider $(1, 2) \rightarrow$ $\overset{12}{(11)(22)}$
 $A \cdot B \rightarrow$ leading with ϵ

Consider $(2, 3) \rightarrow 23$
 $(22)(33).$
 $B \cdot B \rightarrow$ leading by $A.$

Consider $(3, 4) \rightarrow 34$
 $(33)(44).$
 $B \cdot B \rightarrow$ leading by A

Step 3:- Consider $(1, 3) \rightarrow 13$ can be consider as
 123
 $(11)(23)$
 $A \cdot A \rightarrow \emptyset$
 $(12) \cdot (33)$
 $(\epsilon, B) \cdot B$
 $\epsilon B, BB \rightarrow$ leading by $A.$
 $\emptyset A$

Consider $(2, 4) \rightarrow 24$ can be considered as.
 234
 $(22)(34).$
 $B \cdot A \rightarrow \emptyset$
 $(23) \cdot 44$
 $A \cdot B \rightarrow \epsilon, B.$

4 - (16)

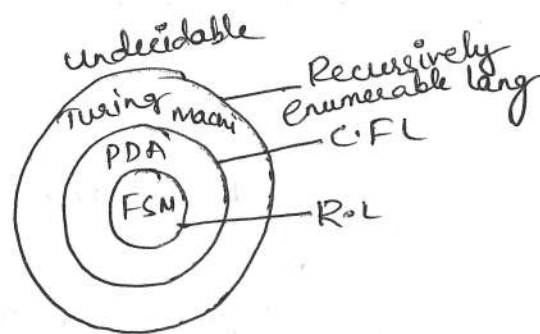
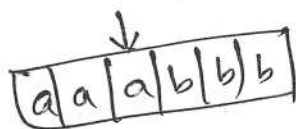
Step 4: Consider $(1, 4) \Rightarrow 1, 4$ can be considered as

$1\ 2\ 3\ 4$
 $(11)\ (24)$
 $A \cdot (S, B)$
 $A \cdot S, A \cdot B$
 $\times\ S, B$
 $(1, 2)\ (3, 4)$
 $(S, B) \cdot (A)$
 SA, BA
 $\times\ \times$
 $(13)\ (44)$
 $A \cdot B$
 S, B

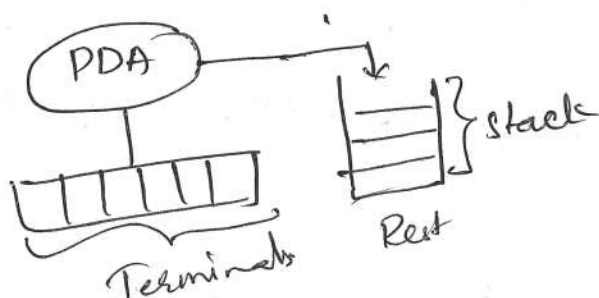
if get S in 14 i.e the length of
 string 4 (abbb)
 then we will tell that w is a member
 of CFL

Turing Machines

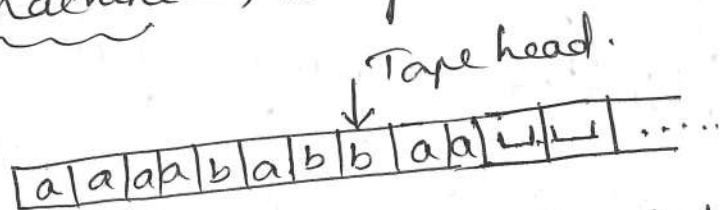
FSM: The i/p string (data structure)



PDA $\rightarrow$ The i/p string & A stack.



Turing machine $\rightarrow$ A Tape.



- $\rightarrow$ A tape is a sequence of infinite symbols.
- $\rightarrow$ tape head is positioned on current control
- tape head can move either one step forward or step back depending on the requirement

Tape alphabets $\Sigma = \{0, 1, a, b, x, z_0\}$

The blank symbol ($\sqcup$) is a special symbol ($\sqcup \notin \Sigma$) 4- (17)

The blank is a special symbol used to fill the infinite tape.

Turing Machine :-

A Turing machine is an accepting device which accepts the language (recursively enumerable set) generated by type 0 grammars. It was invented by Alan Turing in 1936.

There are various features of the Turing machine.

1. It has an external memory which remembers arbitrary long sequence of input.
2. It has unlimited memory capability.
3. The model has a facility by which the input at left or right on the tape can be read easily.
4. The machine can produce a certain output based on its input. Sometimes it may be required that the same input has to be

used to generate the output. So in this machine, the distinction between input and output has been removed. Thus a common set of alphabets can be used for the Turing machine.

9. Formal Description $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$
A Turing machine can be defined as a collection of 7 components.

Q : the finite set of states

Σ : the " " " i/p symbols.

Γ/T : " Tape symbol (non empty)

q_0 : the initial state

F : a set of final states (Accept & Reject)

b : a blank symbol used as a endmarker for input

δ : a transition or mapping function.

$$Q \times \Sigma \rightarrow T \times (R \cup L) \times Q$$

Thus the production rule of Turing machine will be written as

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

Basic model of Turing Machine.

1. The I/P tape is having an infinite no. of cells each cell containing one i/p symbol. The empty tape is filled by blank characters

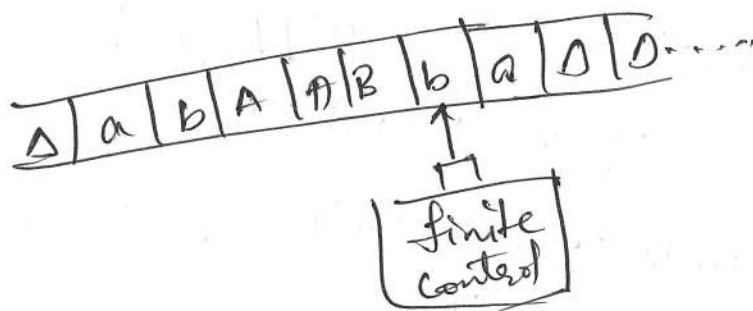
a	b	c	Δ	Δ	Δ
---	---	---	---	---	---

 ... I/P tape.

2. The finite control & the tape head which is responsible for reading the current i/p symbol. The tape head can move to left to right

3. A finite set state through which m/c has to undergo

4. Finite set of symbols called external symbols which are used in building the logic of TM



- A language L and Σ is said to be Recursively enumerable if there exists a Turing machine that accepts it.

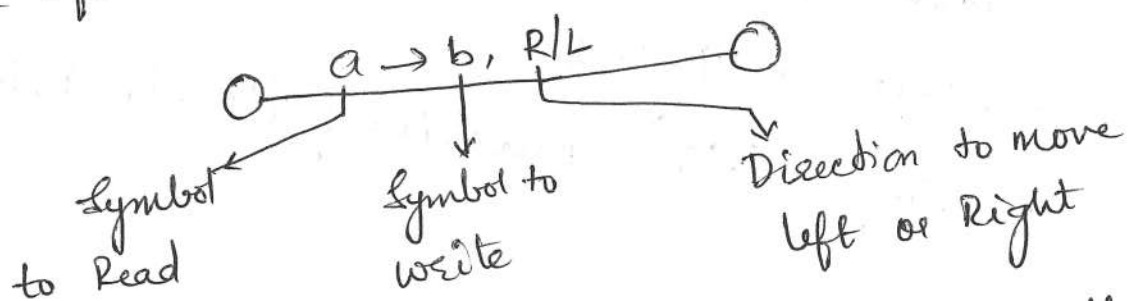
Rules of operation - 1

At each step of the computation

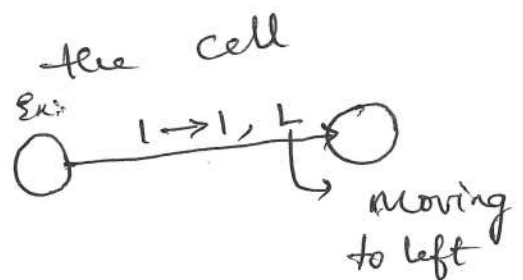
- Read the current symbol
- update (i.e. write) the same cell

- move exactly one cell either left or right

If we are at the left end of the tape and trying to move left, then do not move stay at the left end.



If you don't want to update just write the same symbol



Rules of operation - 2.

→ Control is with a sort of FSM

→ Initial state

→ Final states (there are two final states)

1. The accept state (string is accepted)
2. " Reject " (" " rejected)

when done computation can either

1. Halt and Accept
2. Halt " Reject
3. Loop (the machine fails to halt)

Instantaneous Description.

- An ID's of a Turing machine M is a string $\alpha q_i \gamma$ where q_i is the present state of M , the entire input string is split as α & γ , the first symbol of γ is the current symbol a under the R/W head and γ has all the subsequent symbols of the input string, and the string α is the substring of the i/p string formed by all the symbols to the left of a .

$$\begin{array}{c} \overline{\alpha | a | b | b} \\ \quad \uparrow \\ \quad \alpha \quad \text{head} \end{array}$$

$\underbrace{a|a|b|b}_x$
 $\uparrow$
 head

Example : Design a TM that accepts
 $L = \{0^n 1^n 2^n \mid n > 0\}$.

$$L = \{0^n 1^n 2^n \mid n \geq 0\}.$$

$W = 001122\#$

Sol ID of a Turing M.

q_0 001122 #

1 x 9,01122#

1
- x09, 1122 #

$\vdash x0yq_2122\#$

$\vdash x0y1q_222\#$

$\vdash x0yq_3122\#$

$\vdash x0yq_3y122\#$

$\vdash x\cancel{0}yq_30y122\#$

$\vdash q_3x0y122\#$

Decision properties of CFL

1. Emptiness: Given CFG G there is an algorithm for deciding if $L(G) = \emptyset$

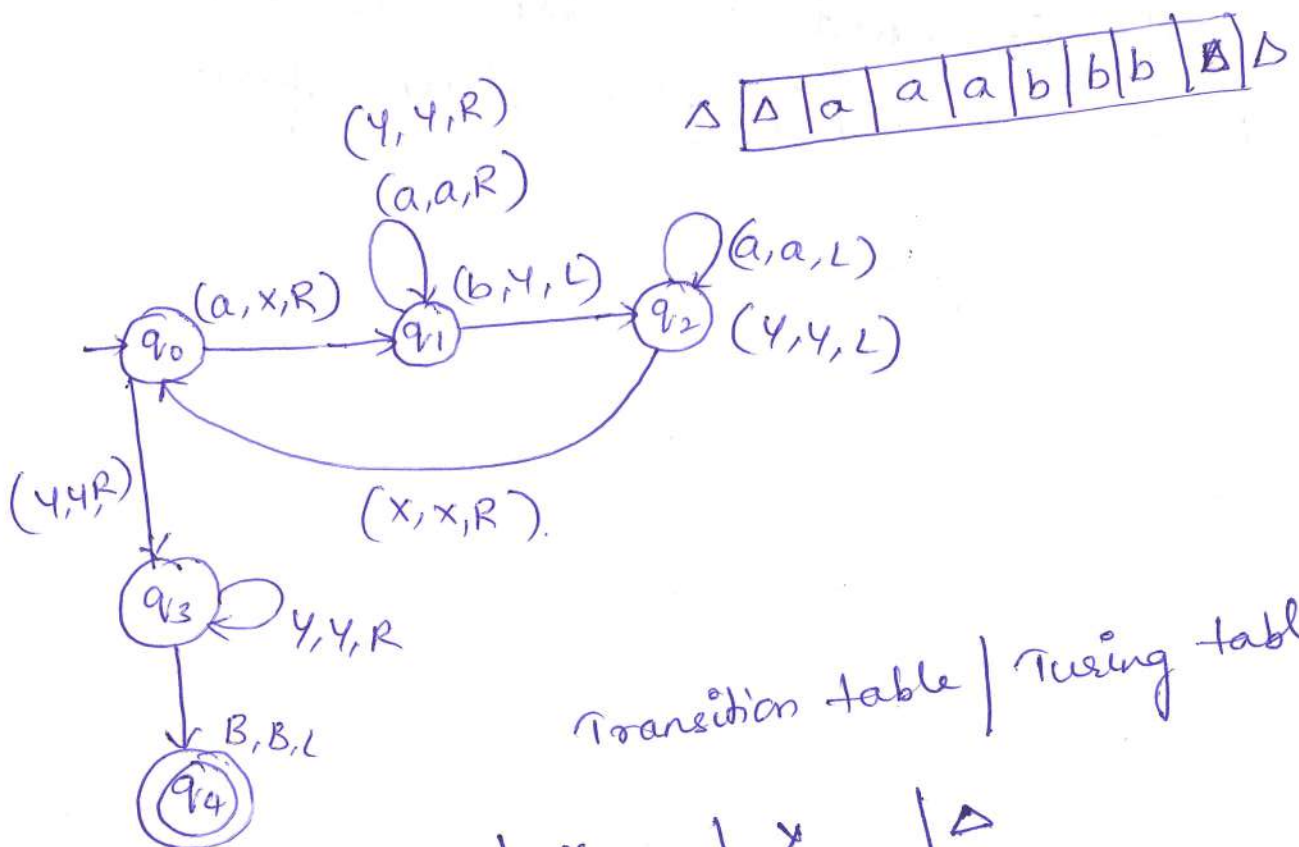
Proof: use the algo for eliminating useless variables.

2. If the start variable is useless return true ($L(G) = \emptyset$), else return false.

10 Design T.M for $\{a^n b^n \mid n \geq 1\}$.

Sol Consider aabb.

Let a be substituted by x
 " b " " " y

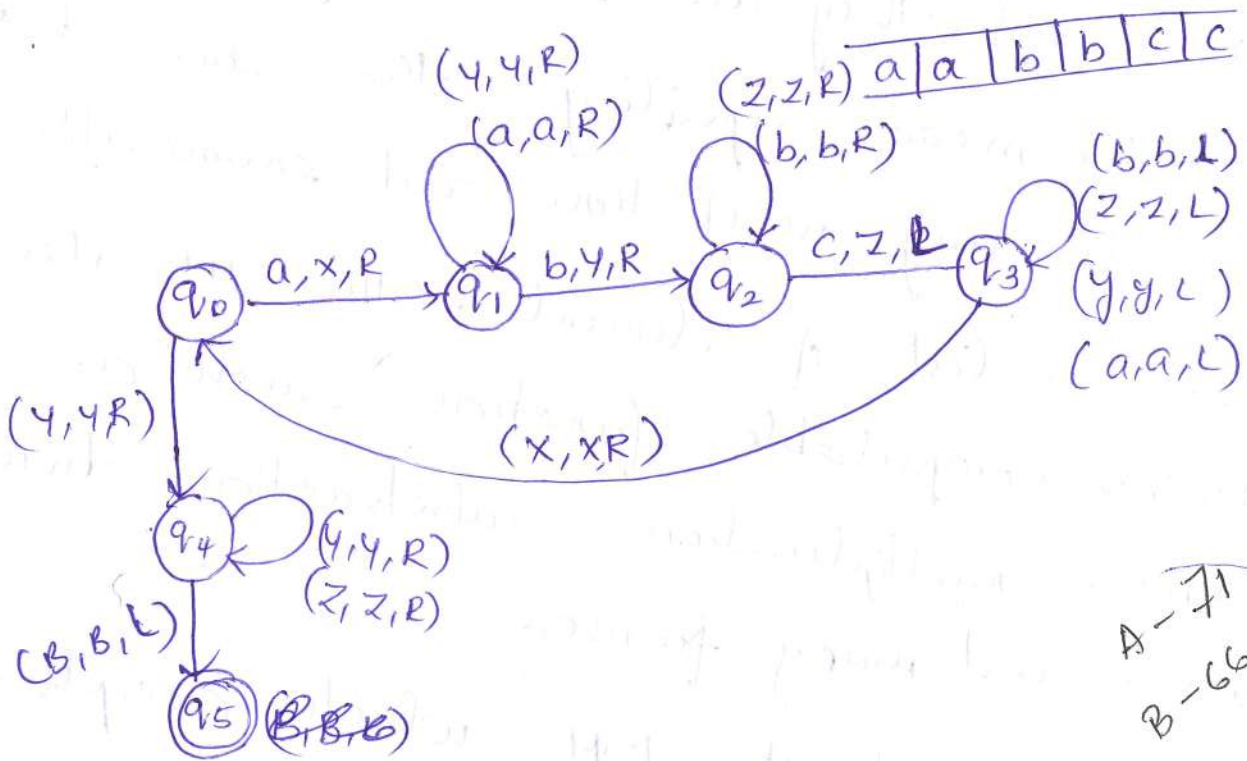


Transition table / Turing table.

	a	b	x	y	Δ
q_0	(q_1, x, R)	—	—	(q_3, y, R)	—
q_1	(q_1, a, R)	(q_2, y, L)	—	(q_1, y, R)	—
q_2	(q_2, a, L)	—	(q_0, x, R)	(q_2, y, L)	—
q_3	—	—	—	(q_3, y, R)	(q_4, B, L)
q_4	—	—	—	—	—

Design T.M for $\{a^n b^n c^n \mid n \geq 1\}$

aa bb cc.



A-71
B-66

States	a	b	c	x	y	z	Δ
q_0	(q_1, xR)	-	-	-	(q_4, yR)	-	-
q_1	(q_1, aR)	(q_2, yR)	-	-	(q_1, yR)	-	-
q_2	-	(q_2, bR)	(q_3, zL)	-	-	(q_2, zR)	-
q_3	(q_3, a, L)	(q_3, b, L)	-	(q_0, xR)	(q_3, yL)	(q_3, zL)	-
q_4	-	-	-	-	(q_4, yR)	(q_4, zR)	(q_5, B, L)
q_5	-	-	-	-	-	-	-

Language of Turing machine:-

The Turing machine accepts all the language even though they are recursively enumerable

Recursive means repeating the same set of rules for any no. of times and enumerable means a list of elements. The TM also accepts computable functions such as addition, multiplication, subtraction, division power and many more.

Example: Construct T.M which accepts the language of 'aba' over $\Sigma = \{a, b\}$

Sol Let us assume input tape as

a	b	a	Δ
---	---	---	----------

The tape head will read out the sequence upto the Δ characters. If the tape head is readout 'aba' string then TM will halt after reading Δ

Initially state is q_0 and head points to 'a' as:

a | b | a | Δ
 $\uparrow$

Then we ~~will~~ move $\delta(q_0, a) = \delta(q_1, A, R)$ which means it will go to state q_1 , replaced 'a' by 'A' and head will move to right as

A | b | a | Δ
 $\uparrow$

The move will be $\delta(q_1, b) = \delta(q_2, B, R)$ which means. it will go to state q_2 , replaced 'b' by B and head will move to right as

A | B | a | Δ
 $\uparrow$

Then move will be $\delta(q_2, a) = \delta(q_3, A, R)$ which means. It will go to state q_3 , replace 'a' with 'A' and head will move to right

as A | B | A | Δ
 $\uparrow$

The move $\delta(q_3, \Delta) = (q_4, \Delta, S)$ which means it will go to state q_4 which is the halt state

and ~~that~~ it is an accept state of T.M

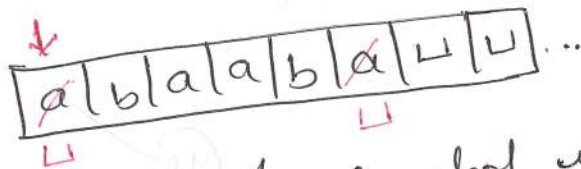
Transition Turing table

state	a	b	Δ
q_0	(q_1, A, R)	-	-
q_1	-	(q_2, B, R)	-
q_2	(q_3, A, R)	-	-
q_3	-	-	(q_4, Δ, S)
q_4	-	-	-

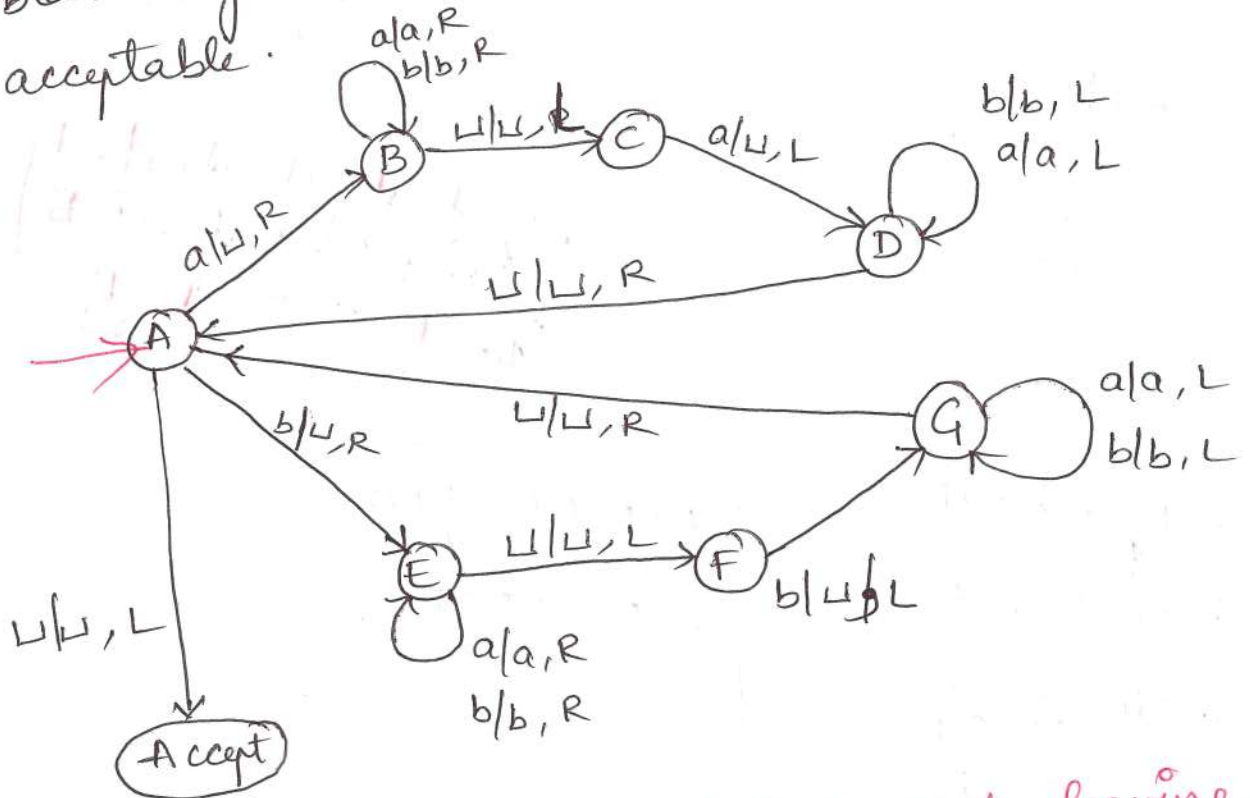
$\Delta/A/B/\Delta$
↑

Design TM that accepts even palindrome over the alphabet $\Sigma = \{a, b\}$

Sol Consider abaaba

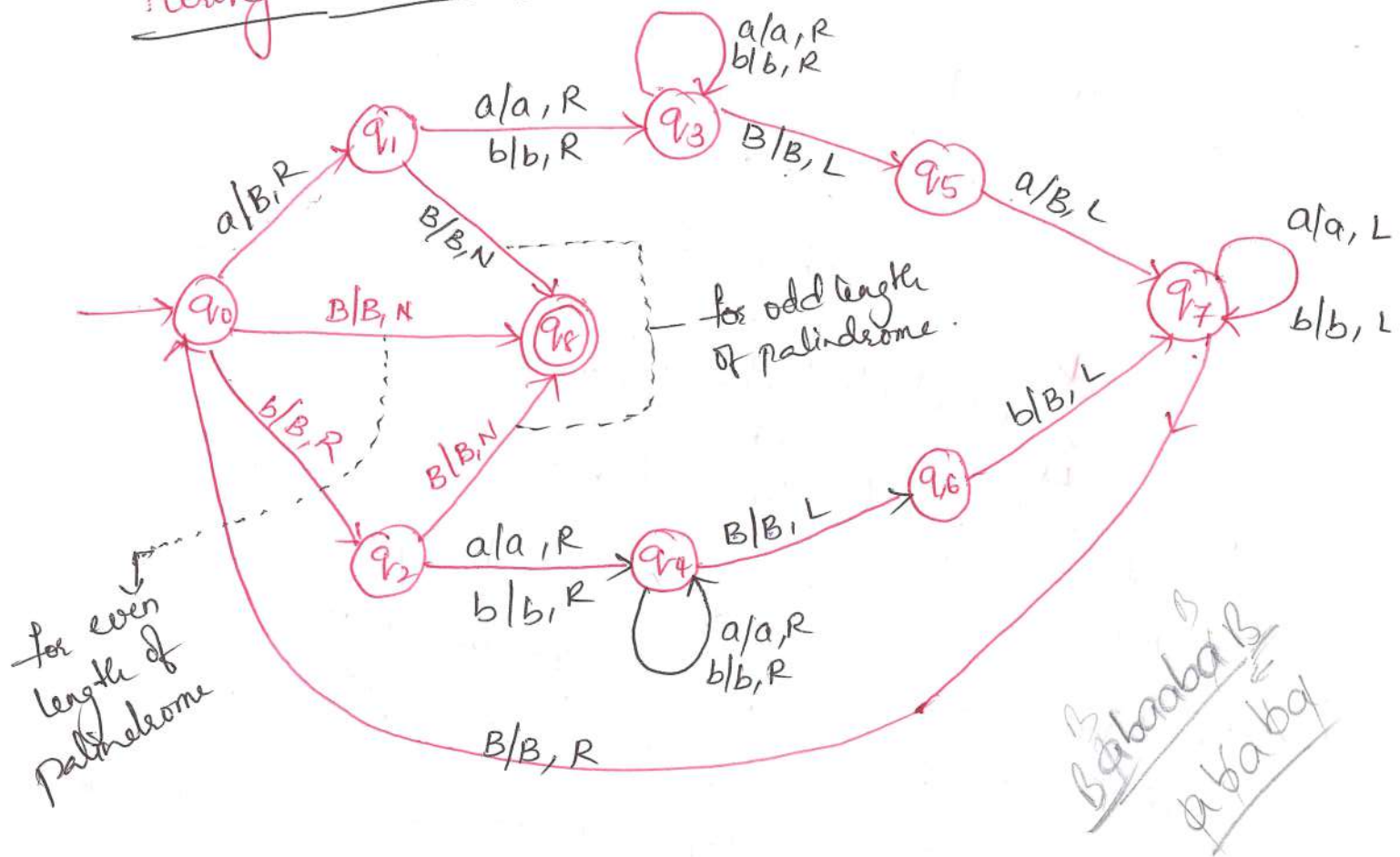


1. replace first symbol with 'L'
2. move to last symbol and replace it with 'L'
3. Again come to first symbol (How do we know it i.e if left most is 'L')
4. Repeat those steps until you see all are blank symbols then we say that string is acceptable.



if a particular condition or state is not having transition then it is reject state

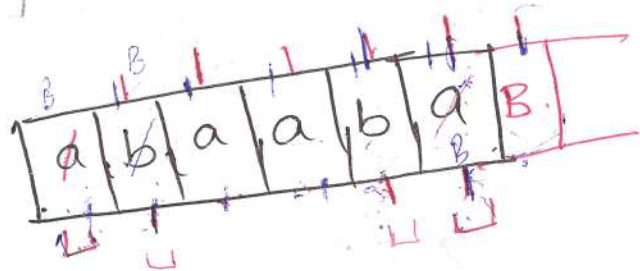
Turing machine for odd / even palindrome



Even length palindrome string is 'e'

odd " " " have either a or b i.e. '1' symbol

Consider $abaaba$



$\vdash BaabbaaB$
 $\uparrow$
 $\vdash BBabbaaB$
 $\uparrow$
 $\vdash BBabbaaB$
 $\uparrow \uparrow \uparrow \uparrow \uparrow$
 $\vdash BBabbaBB$
 $\uparrow \uparrow \uparrow \uparrow \uparrow$
 $\vdash BBBbbaBB$
 $\uparrow \uparrow \uparrow$
 $\vdash BBBbbBBB$
 $\uparrow \uparrow \uparrow$

$\vdash BBBBbBBB$
 $\uparrow$
 $\vdash BBBBbBBB$

Construct Turing machine to perform proper subtraction.

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

Lt $m=5$ $n=2$.

$m > n$

B	1	1	1	1	1	0	1	1	B
---	---	---	---	---	---	---	---	---	---

if we found zero on left then we can consider $m > n$.

$m = n$

B	1	1	0	1	1	B
---	---	---	---	---	---	---

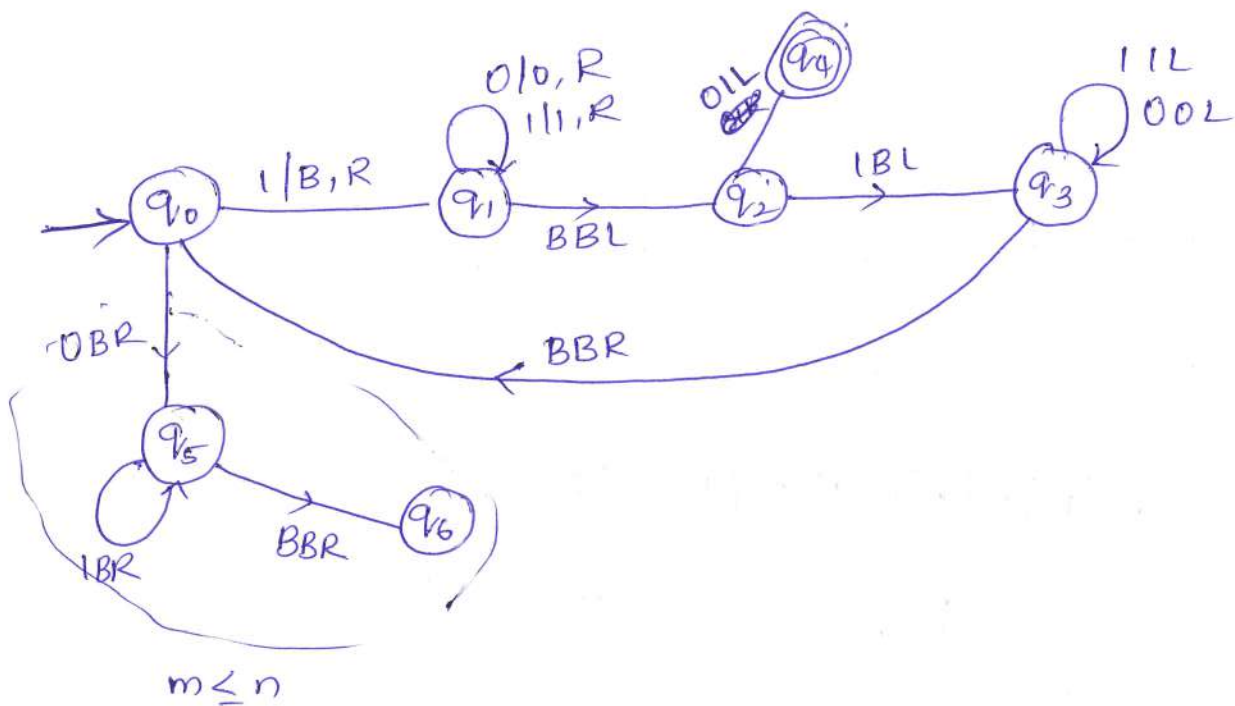
B B B B

if we find 0 on left side we consider $m = n$
 here 0 is also converted into blank.

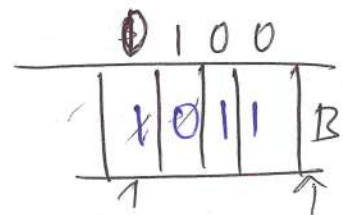
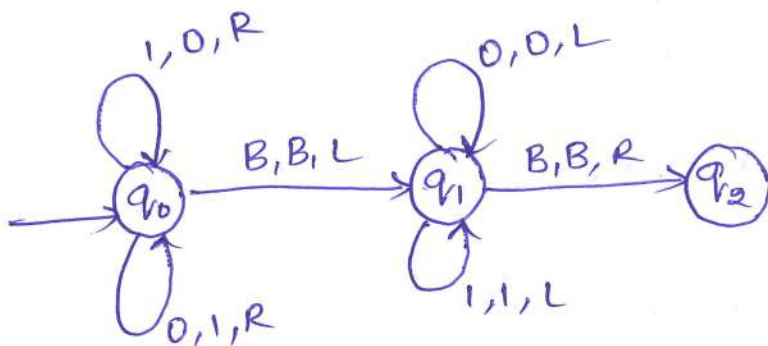
$m < n$

B	1	1	0	1	1	1	1	1	1	B
---	---	---	---	---	---	---	---	---	---	---

Convert all of them into blank.

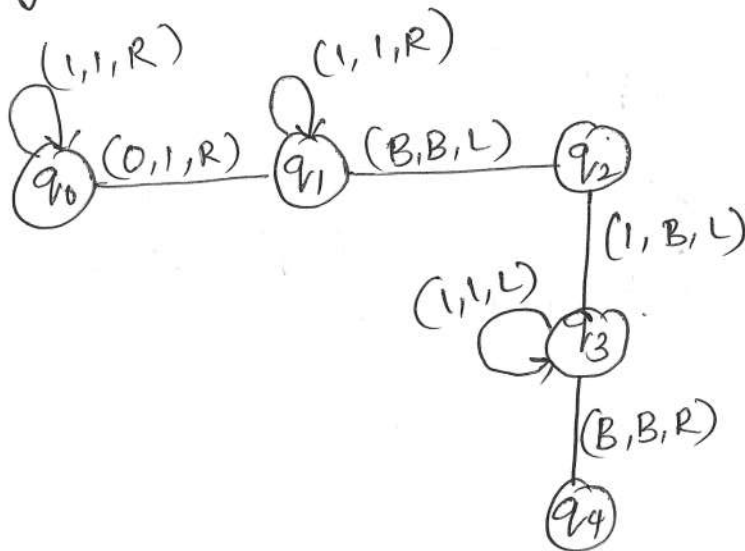


Turing machine for 1's complement. (transducent)



* Makes 1 to zero and zero to one
once you reach Blank then the string

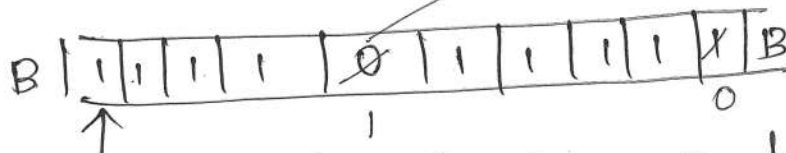
Turing machine for addition.



Consider $a = 4 \Rightarrow 1111$

$b = 5 \Rightarrow 11111$

Let the input tape be $11111+11111$ is an '+' symbol $4+5=9$
no. of 1's = 9

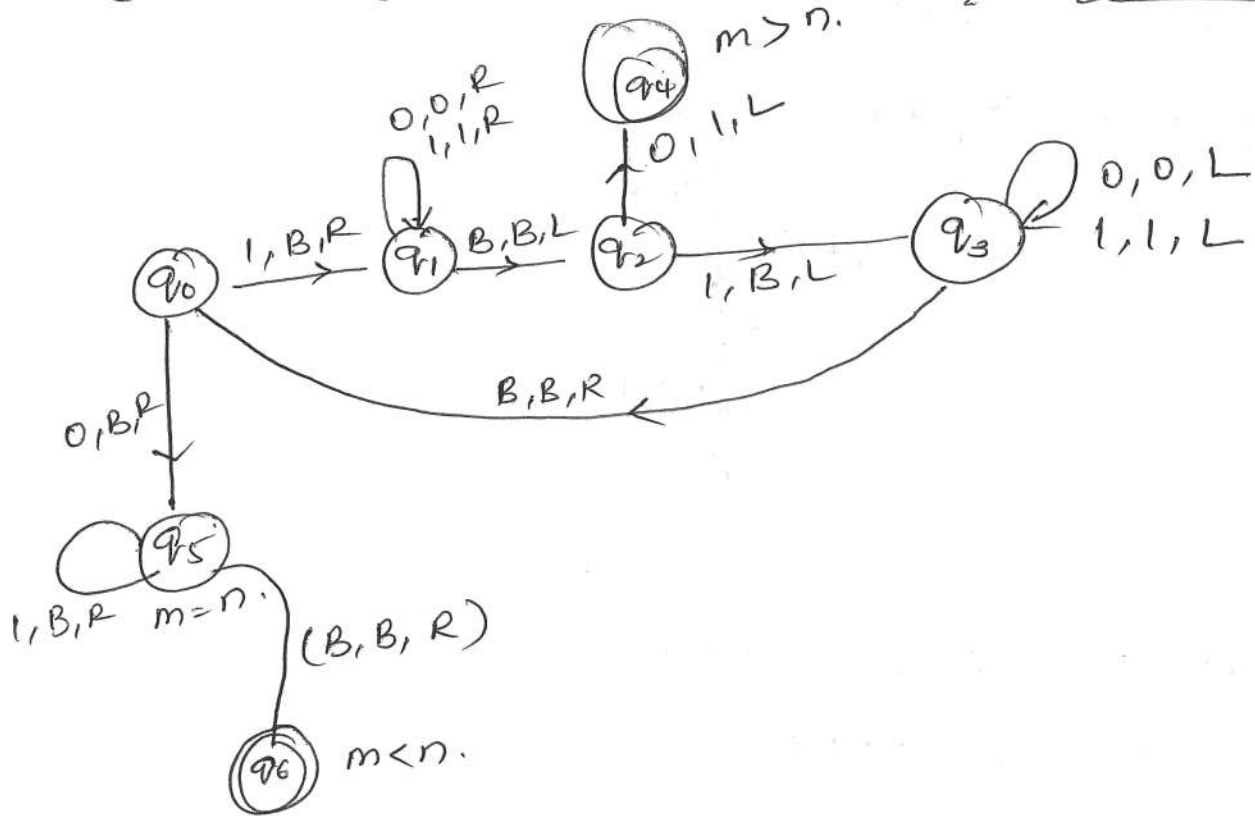


on the left side of zero simply move tape to right

→ Once the zero is occurred make it as '1' and move tape to right. Once you reach blank move your tape left and make '1' to zero. Then ~~get~~ ^{move} your tape to initial state.

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

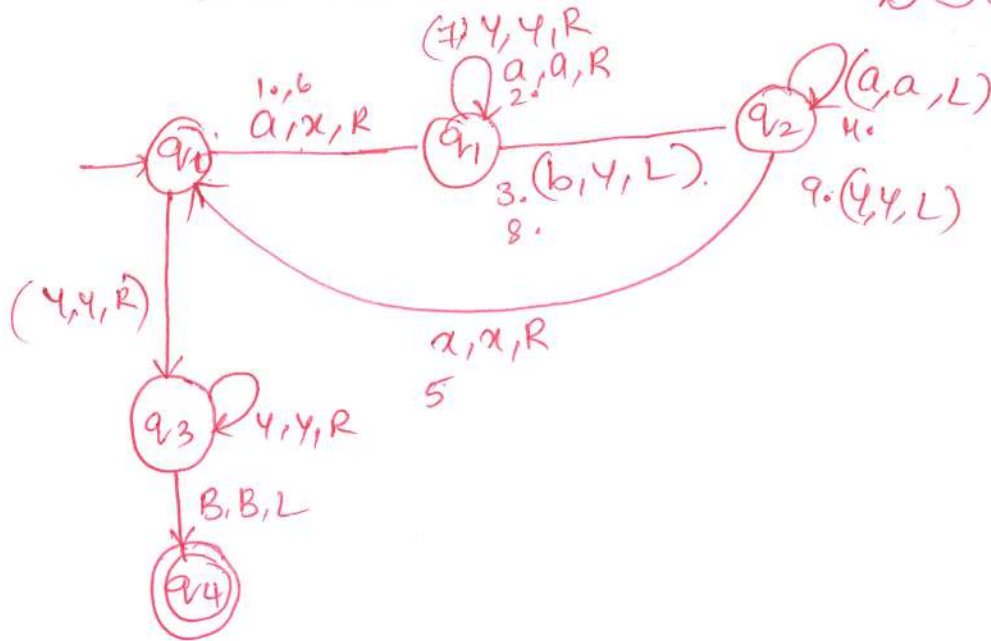
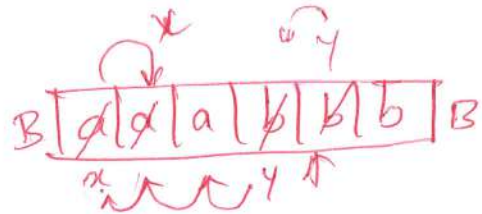
Subtraction



$$a^n b^n \quad |n \geq 1$$

$$a = x \quad b = y$$

$$a^3 b^3 \rightarrow a a a b b b$$

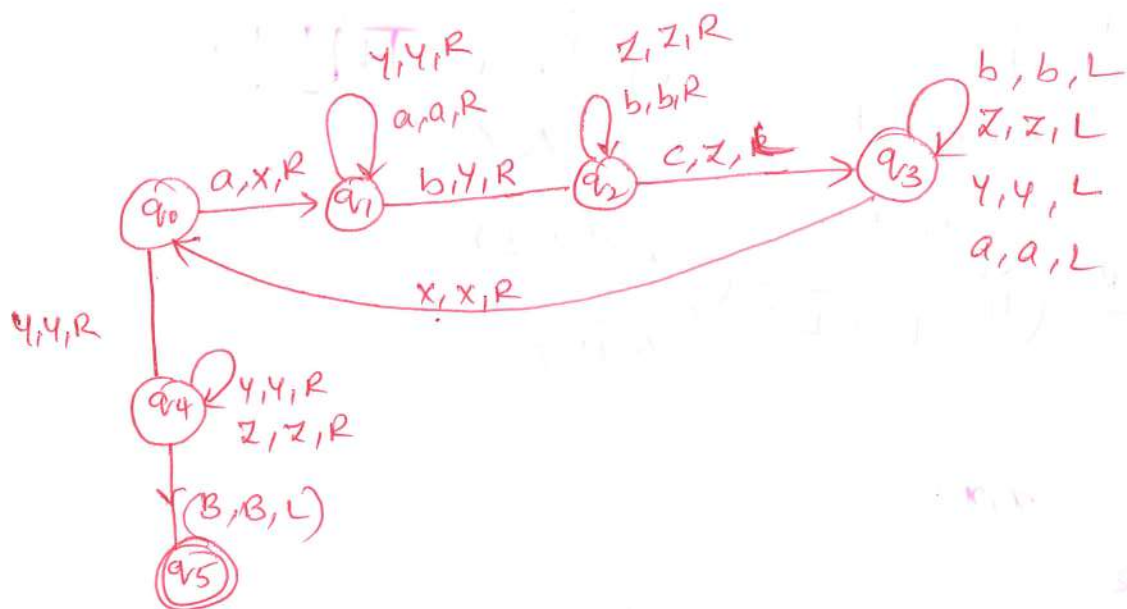


TM $\rightarrow$ Read, write is possible
 $\rightarrow$ ctrl can move in left or right

$$\delta = Q \times \Gamma^* \rightarrow Q \times \Gamma^* \times (L, R)$$

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

$$a^n b^n c^n \Rightarrow a^3 b^3 c^3 \rightarrow aaabbbccc$$



TM $\rightarrow$ Read, write, move head $\leftarrow$ HT
 TM can move in left or right
 (a, x, R) ... (y, y, L) ...
 (b, b, R) ... (z, z, L) ...