(Q.5) (i) $L_1 = \{a^n b^m : n \geq 4, m \leq 3\}$

$L_2 = \{a^n b^m : n < 4, m \leq 3\}$

$L_3 = \{a^n b^m : n \geq 1, m \geq 1, nm \geq 3\}$

(Q6) $\Sigma = (0,1)$

i) All strings containing even no. of 0's.

ii) All strings having atleast two occurances of substring "00".

Regular expression to nfa :-

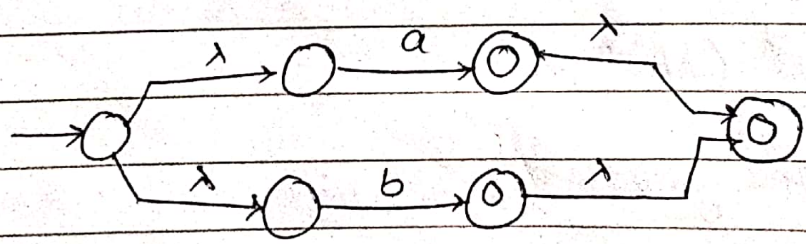
operators :- +, •, *

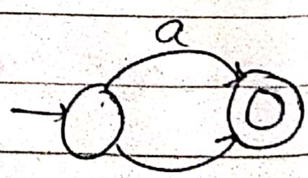
Thompson's rule:-

It states that we must have one initial and one final state.

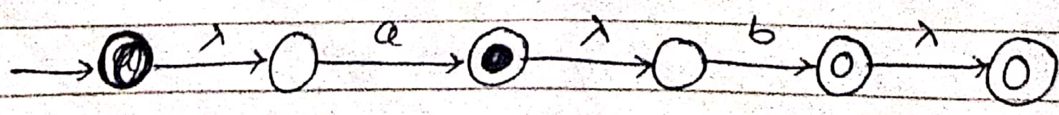
Eg:

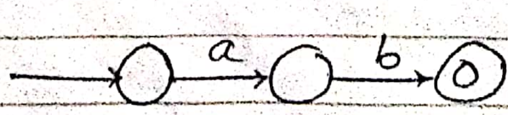
a+b.



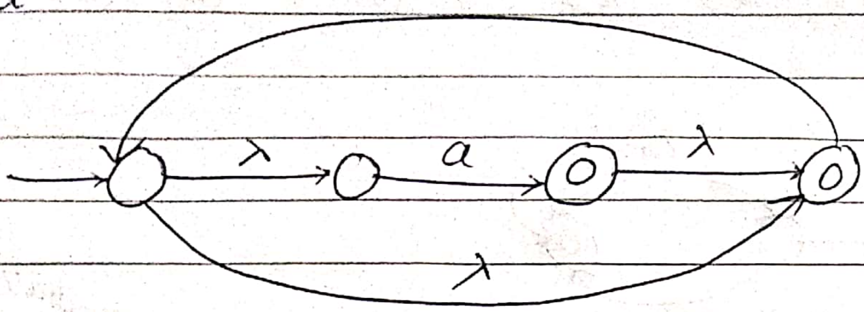


a•b

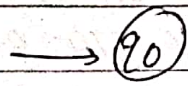




a^*

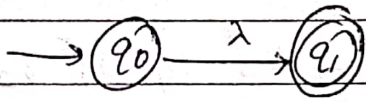


ϕ



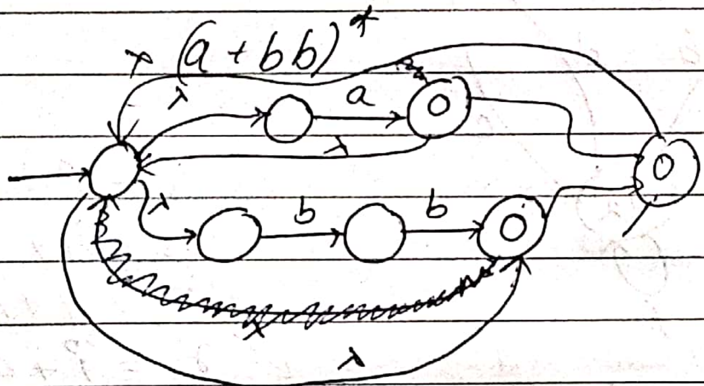
$S \to A$.

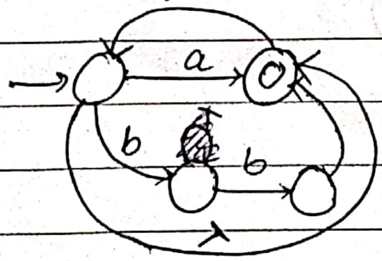
λ



$S \to \lambda$.

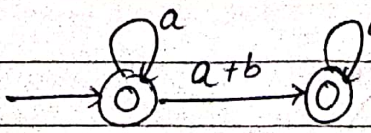
According to thompson's ~~φ~~ rule complete this nfa conversion





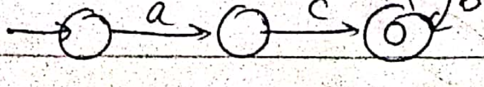
Q1 $((a+bb)^* \cdot (ba^* + \lambda))$

Q2



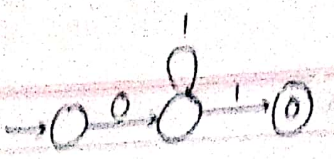
$= a^* \cup (a^* \cdot (a+b) c^*)$

Q3

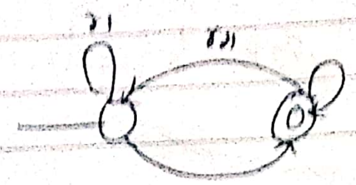


$= acb^*$

Scanned by CamScanner

(4) 

01^*1 (or) 01^+

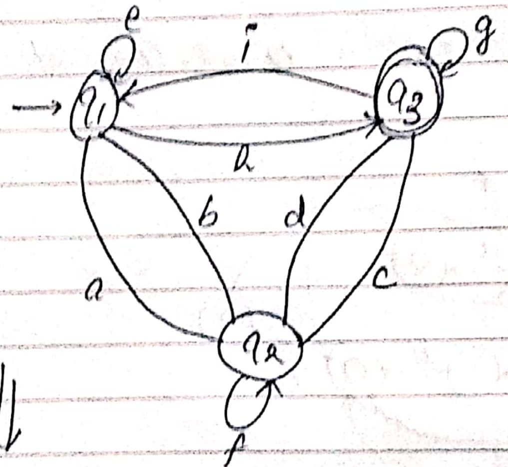


$V = 2$ $\quad E = 4.$

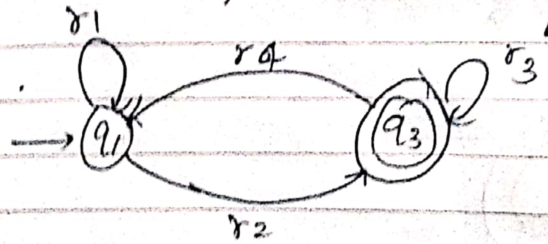
For a given V we will have V^2 Edges. Then it is called GTG graph. (gendralised transition graph.)

$$r_1^* r_2 + r_1^* r_2 r_3^* + r_1^* r_2 r_3^* r_4 \ldots$$

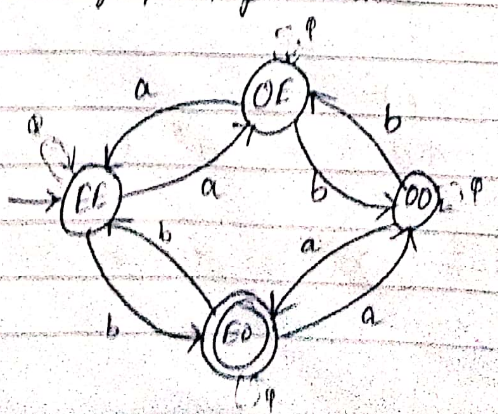
$$= r_1^* r_2 \left[r_3^* + r_3^* r_4 r_1^* r_2 \right]^*.$$



$$r_1 = e + af^*b$$
$$r_2 = h + af^*c$$
$$r_3 = g + df^*c$$
$$r_4 = i + df^*b$$

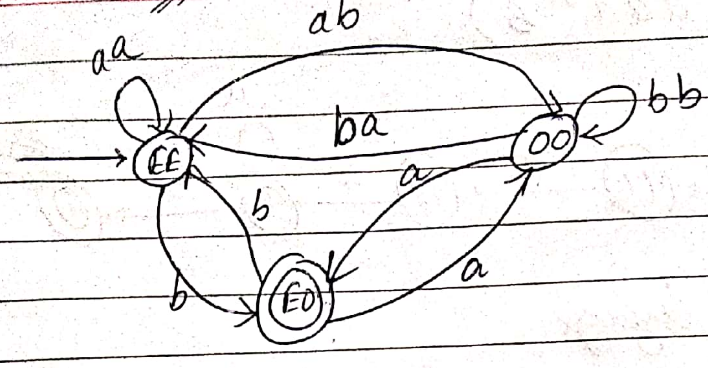


(5) Given a graph, convert it into complete gtg graph.

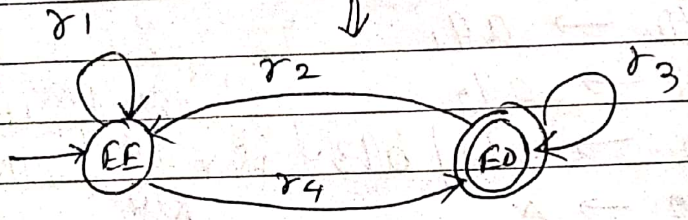


$\Rightarrow a\phi^* a$

An



$\Downarrow$



$r_1 = (aa + ab(bb)^*ba)^*$

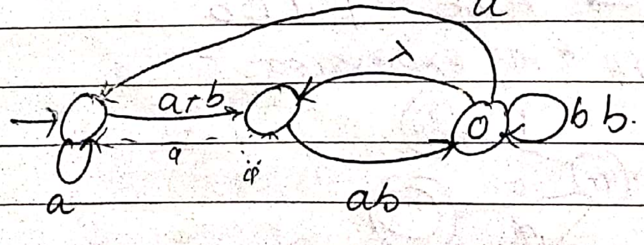
~~r2 = (ab(bb)*a)*~~

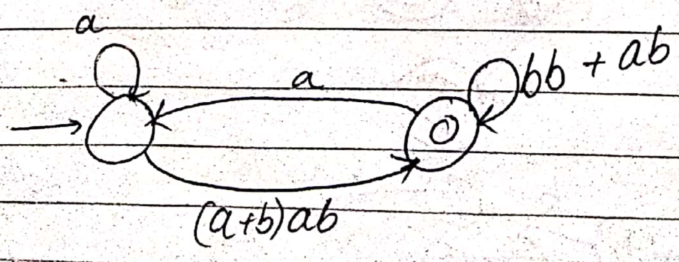
$r_3 = a(bb)^*a$

$r_2 = b + ab(bb)^*a$

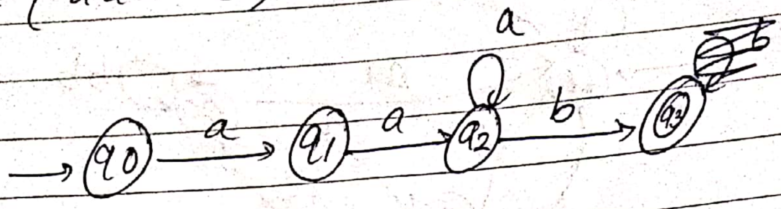
$r_4 = b + a(bb)^*ba$.

(a)





$L = (aaa^* b)$



Writing language:-

$q_0 \to a q_1$
$q_1 \to a q_2$
$q_2 \to a q_2 / b q_3$
$q_3 \to \lambda$

} In all these cases the variables are in right side these are called RIGHT LINEAR PRODUCTION

If variable accumulated towards left then it is left linear grammar.

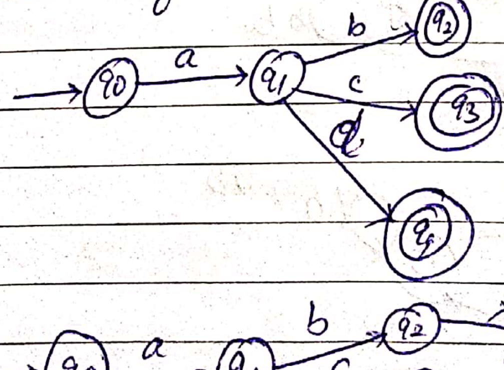
eg:- $S \to$ ~~[illegible]~~ Sa / a

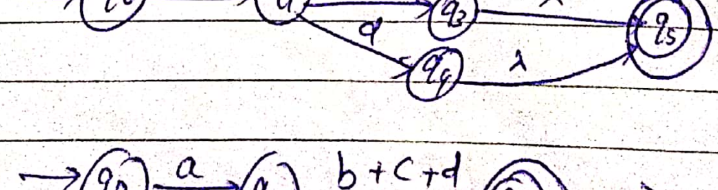
If the grammar is not left or right linear it is called linear grammar.

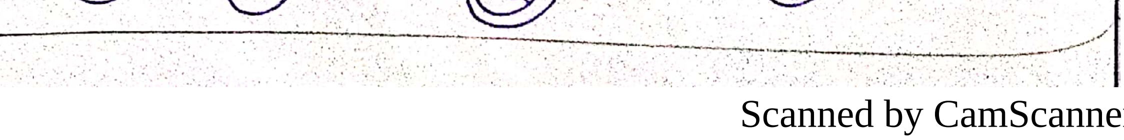
eg $S \to AaB / aBb$.

Conversion of NFA to reg. expression

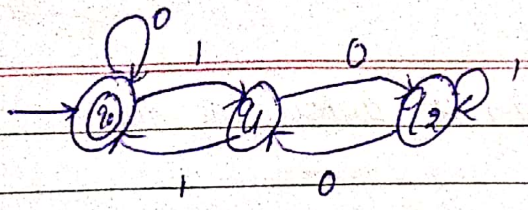
①

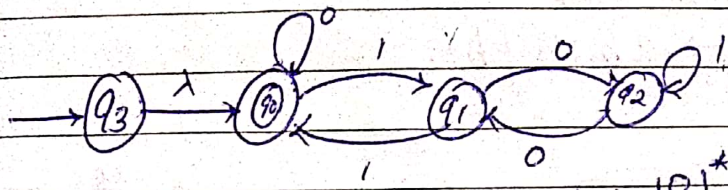


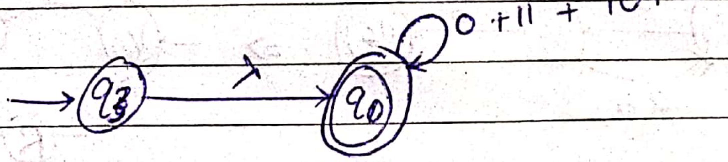


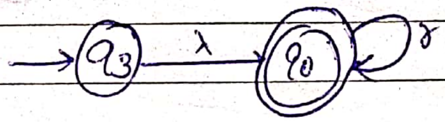


(2)







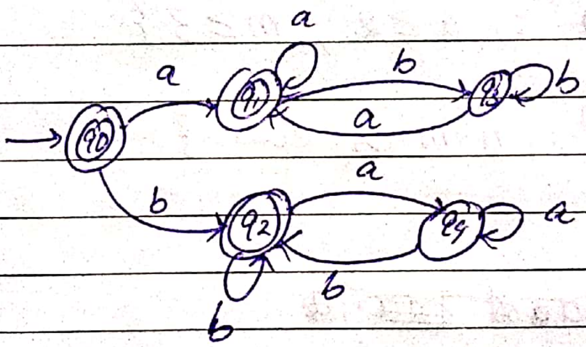


~~$r = 0+11+101^*01$~~

$r = 0 + 1[01^*0]^*1$

(3)

Hw



Given a grammar how to convert into automata.

For right linear :-

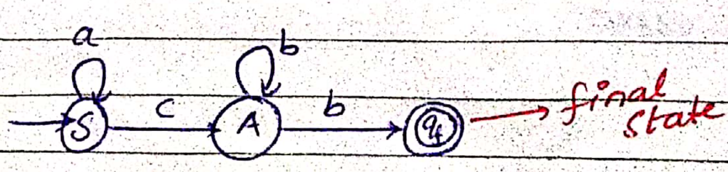
(1)

$S \to aS$

$S \to cA$

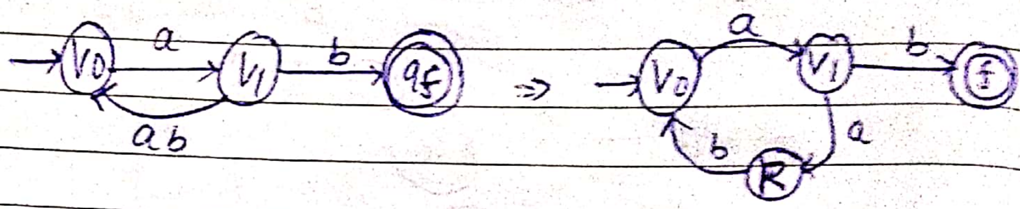
$A \to bA$

$A \to b$



Construct the finite automata for

$V_0 \to a V_1$
$V_1 \to ab V_0 \mid b$.

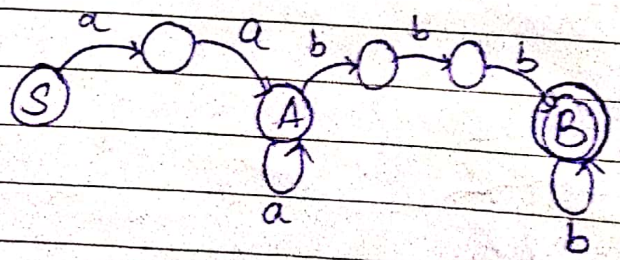


Right linear

① $L = \{ a^n b^m : n \geq 2 , m \geq 3 \}$

② $L = \{ a^n b^m : n+m \text{ is even} \}$

~~$S \to aaaa$ [illegible]~~
$S \to aaA$.
$A \to bbbB \mid aA$
$B \to bB \mid \lambda$



③ n+m is even.

Closure Properties :-

Given two languages L_1 and L_2. If we are able to ~~operate~~ generate regular expression for L_1 & L_2.

then.

$$\begin{array}{cc} L_1 & L_2 \\ \downarrow & \downarrow \\ RE_1 (r_1) & RE_2 (r_2) \end{array}$$

Union Property .

$$r_1 + r_2 \rightarrow L_1 \cup L_2$$

Star ~~Concatenation~~ property :-

$$L_1 \leftarrow r_1 = (ab^*)$$

then

$$L_1^* \leftarrow (r_1)^* = (ab^*)^*$$

Concatenation property :-

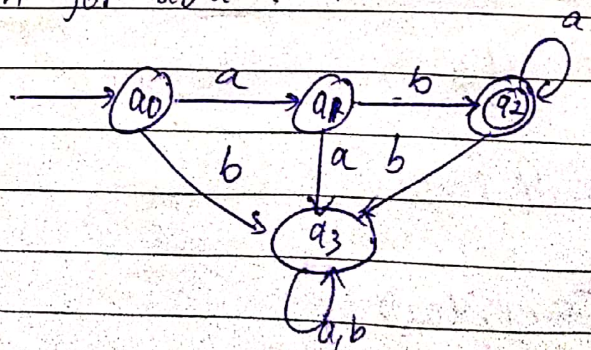
$$L_1 . L_2 \rightarrow r_1 . r_2 .$$

Complement property :-

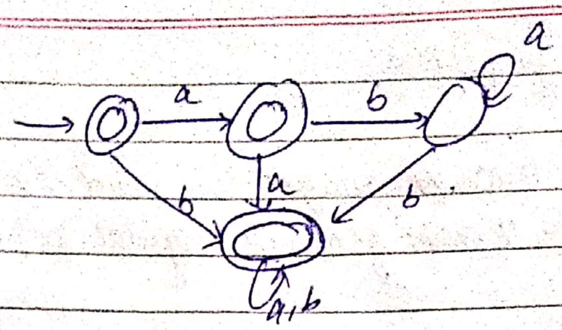
$\bar{L}$

$L \leftarrow$ Reg. Lang .

DFA for $aba^* - L_1$



L_1 =



for DFA

~~to~~ $L_1 = \{Q, \Sigma, \delta, q_0, F\}$

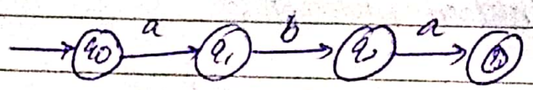
$\overline{L_1} = \{Q, \Sigma, \delta, q_0, Q-F\}$.

Reversal operator:-

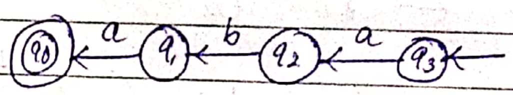
$L_1 = aba$

$L_1^R = ?$

L_1 :



L_1^R =



Intersection operators:-

$L_1 \cap L_2$

$M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ $\Sigma = \{a, b\}$.

$M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$

$\delta((q_i, q_j), a) = (q_k, q_l)$

$\downarrow$

$(q_i, a) = q_k$

$(q_j, a) = q_l$

Initial state $\rightarrow (q_{01}, q_{02})$

Final state $\rightarrow \{F_1 \text{ (or) } F_2\}$

Right quotiond operator :-

$L_1/L_2 = \{ \underbrace{x : xy \in L_1}\ \text{for some } y \in L_2 \}$

$w \in L_1$

$\downarrow$

xy

$\hookrightarrow L_2.$

$L_1 = \{ carrot \}$

$L_2 = \{ t, ot \}$

$L_1/L_2 = \{ carro, carr \}$

$L_1 = \{ xab, yab \}$ $L_2 = \{ b, ab \}$

$L_1/L_2 = \{ xa, ya, x, y \}$

$L_1 = \{ \lambda, a, ab, aba, abab, \dots \}$

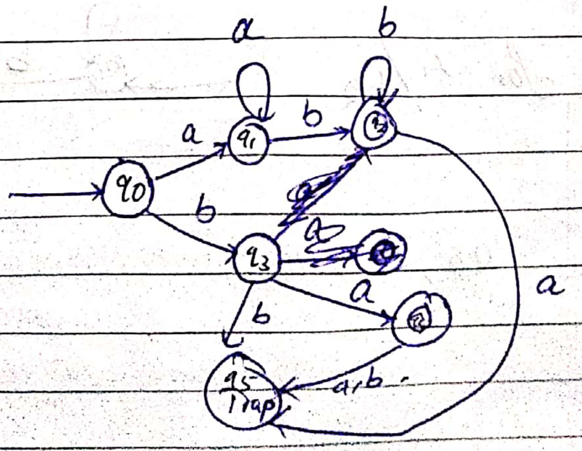
$L_2 = \{ b, bb, bbb, \dots \}$

$L_1/L_2 = \{ a, aba, \dots \}$

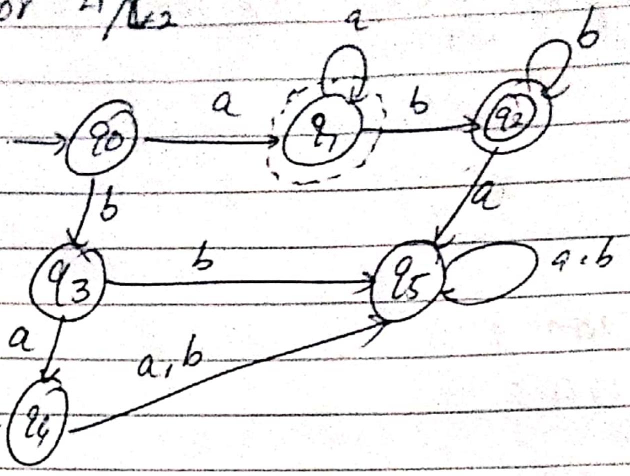
$L_1 = \{ a^n b^m : n \geq 1, m > 0 \} \cup \{ ba \}$

$L_2 = \{ b^m, m \geq 1 \} \Rightarrow b^+$ or bb^*

DFA for L_1.



DFA for L_1/L_2



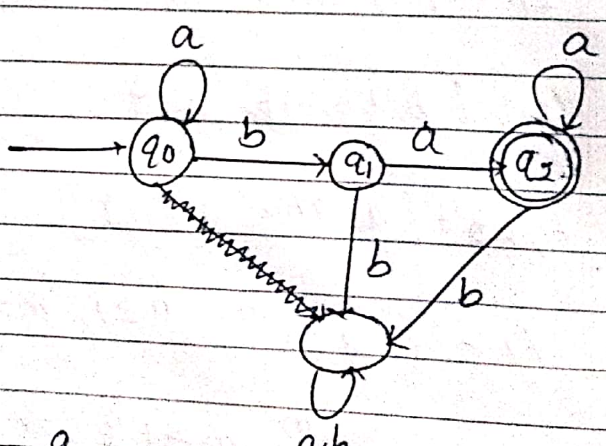
It becomes nonfinal

(*) $L_1 = \{a^* b a a^*\}$

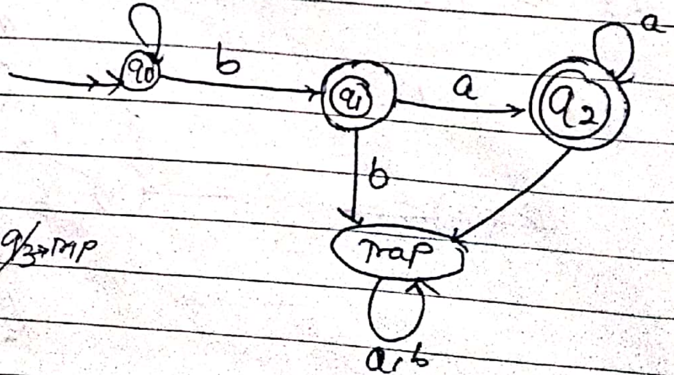
$L_2 = \{a b^*\}$

$L_1/L_2 = ?$

Dfa for L_1



for L_1/L_2



$q_0, q_1, q_2, q_3 \rightarrow$ TMP

Date
Page

Homomorphic operator.

$$w = aba$$

$$h(a) = cb \quad h(b) = da.$$

$$h(w) = cbdacb$$

$L_1 \to r_1 = (a+b) \cdot ab^*$

$h(r_1) = ?$

$$h(r_1) = (cb+da) \cdot (cb(da)^*)$$

① How can you say the given two languages are same?

② How can you say the given language is empty?

PUMPING LEMMA.

$L = \{a^n b^n, n \geq 0\}$ → has equal no of a and b.

Verify whether the given language is regular.

① → $w \in L$, $|w| \geq m$

Eg: $|a^m b^m| \geq m \geq 0$

② → Selecting $a^m b^m$

doing decomposition.

$$\underbrace{a^{m-k}}_{x} \underbrace{a^k}_{y} \underbrace{b^m}_{z}.$$

$$|xy| \leq m, \quad |y| \geq 1$$

$$w_i = x(y)^i z.$$

$w_0 = a^{m-k} (a^k)^0 b^m \notin L$

since it has different no. of a and b.

specific case.

① $|a^3 b^3| \geq 3$

② $\underbrace{a^2}_{x} \underbrace{a}_{y} \underbrace{b^3}_{z}$

$w_1 = a^m b^m \in L.$

$w_2 = a^{m-k} (a^k)^2 b^m$

$= a^{m+k} b^m \notin L.$

If any one of the w does not belong to L then it is not a regular language.

(2) $L = \{a^n b^n c^n : n \geq 0\}$

1) $|w| \geq m \Rightarrow w = a^m b^m c^m$

2) $a^m b^m c^m$

$\underbrace{a^{m-k}}_{x} \quad \underbrace{a^k}_{y} \quad \underbrace{b^m c^m}_{z}.$

3) $w_0 = a^{m-k} (a^k)^0 b^m c^m$

$w_0 = a^{m-k} b^m c^m \notin L$

$\Rightarrow$ Hence given language is not regular

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

(4) $L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$

(5) $L = \{a^n b^k c^{n+k}, n \geq 0, k \geq 0\}$

(6) $L = \{a^n : n$ is a proper square$\}$

③ $L = \{ww^R : w \in \{a,b\}^*\}$

1) $w = aaaa$.

2) $\underset{x}{a}$ $\underset{y}{aa}$ $\underset{z}{a}$.

3) $w_i = x(y)^i z$

wrong decomposition.

2) $\underset{x}{aa}$ $\underset{y}{a}$ $\underset{z}{a}$.

③ $w_i = x(y)^i z$.

$w_0 = aa(a)^0 a$

$w_0 = aaa \notin L$.

④ $L = \{w \in \Sigma^* : n_a(w) < n_b(w)\}$

1) $z = aaabbbbb$

2) ~~$\underset{x}{aaa}$ $\underset{y}{b}$ $\underset{z}{bbbb}$.~~

~~$w_i = x(y)^i z$~~

~~$w_0 = aaab \in L$~~

~~$w_2 = aaabbb \notin L$.~~

2) ~~aabbb~~.

$\underbrace{a}_{x}\ \underbrace{a}_{y}\ \underbrace{bbb}_{z}$.

$w_i = x(y)^j z$.

$w_0 = abbb \in L$

$w_1 = aabbb \in L$

$w_2 = aaabbb \notin L$.

5) $L = \{a^n b^k c^{n+k}, n \geq 0, k \geq 0\}$

① $w = \underbrace{a}_{x}\ \underbrace{b}_{y}\ \underbrace{cc}_{z}$.
②

$\underbrace{a}_{x}\ \underbrace{bc}_{y}\ \underbrace{c}_{z}$.

③ $w_i = x(y)^i z$

$w_0 = a(b)^0 cc$
$= acc \notin L$.

6) a^n . n is a perfect square.
let $n = 4$.

$L = aaaa$

① $\underbrace{a}_{x}\ \underbrace{aa}_{y}\ \underbrace{a}_{z}$.
②

$|xy| = 3 \leq 4$

$w_0 = x(y)^i z$.

$|y| = 2 \geq 1$

$= xz$
$= aa \notin L$.

FORMAT OF WRITING:-

1) $m = 4$
$w = abc^2$
$|w| \geq m$.

2) decompose:- $|xy| \leq m / |y| \geq 1$

3) $w_i = x(y)^i z$.

Context free grammar:

1) ① $S \rightarrow AB$
② $A \rightarrow aaA$
③ $A \rightarrow \lambda$
④ $B \rightarrow Bb$
⑤ $B \rightarrow \lambda$

Derive aab

$S \xrightarrow{1} \underset{l\ r}{AB} \xrightarrow{2} aa\underset{l\ (or)\ r}{AB} \xrightarrow{3} aaB \xrightarrow{4} aa\cancel{B}Bb \xrightarrow{5} aab$

$S \xrightarrow{1} AB \xrightarrow{4} ABb \xrightarrow{5} Ab \xrightarrow{2} aaAb \xrightarrow{3} aab$

Sentencial form

If there are multiple ways to derive a string in given grammar then it's called AMBIGUOUS GRAMMAR

left-most derivation ⇒ pattern because left most variable are substituted first.

Right-most derivation.

~~STRING~~:-

Give left and Right most derivation for:-

1) $S \rightarrow aAB$
2) $A \rightarrow bBb$ 'abbbb'
3) $B \rightarrow A/\lambda$

Left most

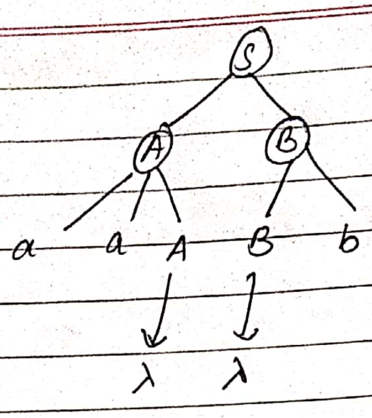
$S \xrightarrow{1} aAB \xrightarrow{2} abBbB \xrightarrow{2} \cancel{abbBbbB} \xrightarrow{3} \cancel{abbbbB}$

$\rightarrow abAbB \xrightarrow{2} abbBbbB \rightarrow abbbbB \xrightarrow{3} abbbb$

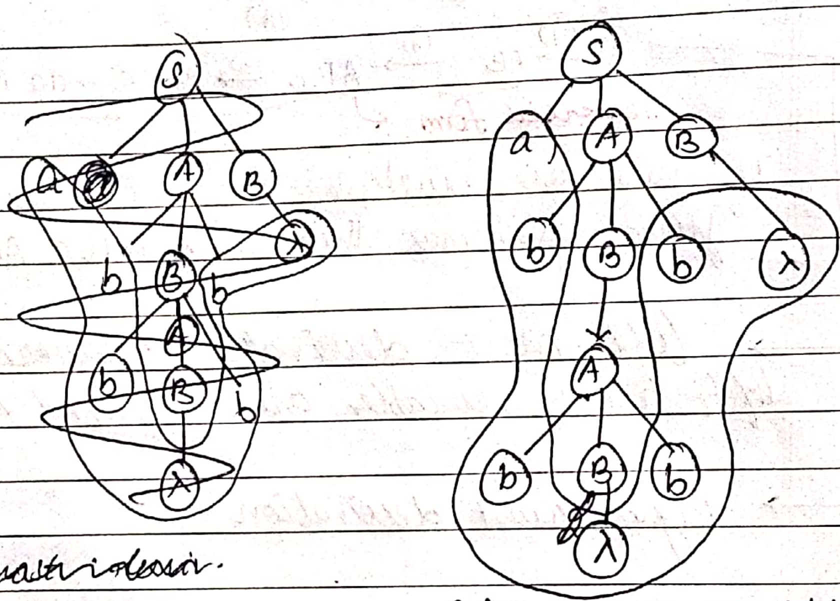
Right most:-

$S \xrightarrow{1} aAB \xrightarrow{3} aA \xrightarrow{2}$

$S \xrightarrow{1} aAB \xrightarrow{4} aAbBb \xrightarrow{3} aAbAb \xrightarrow{2} aAbbBbb \xrightarrow{3} aAbbbb \xrightarrow{3} abbbb$



Left most derivation tree:-



~~Right most deriv.~~

abbλb bλ = abbbb

① Draw the derivation tree for aabbaa.

in ① $S \to aSa \mid bSb \mid \lambda$.

② for "aabbbb"

$S \to AB \mid \lambda$

$A \to aB$

$B \to Sb$

③ $a+b*c$

① $E \to I$

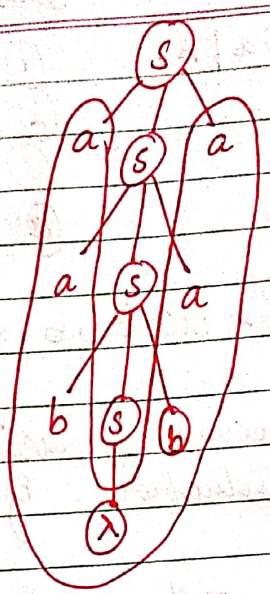
② $E \to E+E$

③ $E \to E*E$

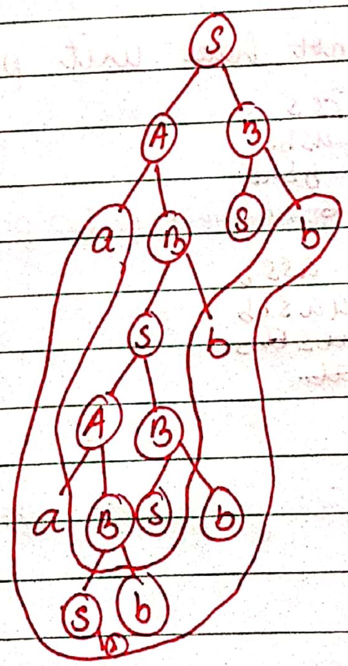
④ $E \to (E)$

⑤ $I \to a \mid b \mid c$.

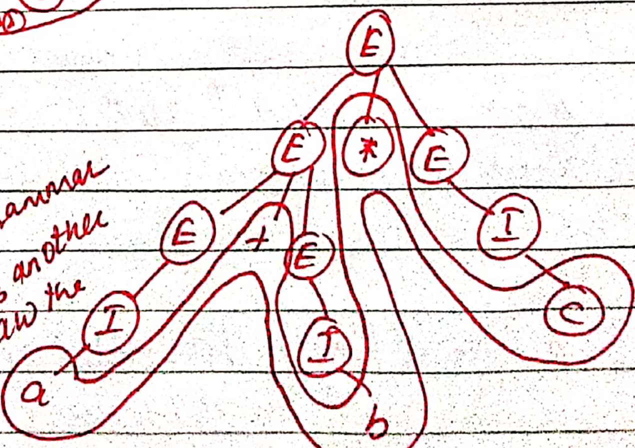
1



2 aabbbb



3



This is an ambiguous grammar because there is another possibility to draw the production.

(1) $S \rightarrow SS \mid aSb \mid bSa \mid \lambda$.

(1) $S \rightarrow SS$
(2) $S \rightarrow aSb$
(3) $S \rightarrow bSa$
(4) $S \rightarrow \lambda$.

$S \xrightarrow{2} aSb \xrightarrow{2} aaSbb \xrightarrow{4} aabb$.

If a variable derives a variable it is called unit production. Eg: $B \rightarrow A$.

For a computer to work there must be 2 restrictions.

(1) We must not have λ production (eg $S \rightarrow \lambda$)

(2) We must not have unit production (eg $A \rightarrow B$)

$\alpha \rightarrow \beta \rightarrow \gamma$

1) $S \rightarrow SS$ → SSS, $aSbS$, ~~$bSaS$~~, ~~S~~ (because it is λ production)
2) $S \rightarrow aSb$ → $aSSb$, $aaSbb$, ~~$abSab$~~, ~~ab~~
3) ~~$S \rightarrow bSa$~~
4) ~~$S \rightarrow \lambda$~~

Condition :- $|\gamma| > |\beta|$
↳ strictly greater than.

Substution Rule :-

$G \begin{cases} A \rightarrow a \mid aaA \mid abBc \\ B \rightarrow abbA \mid b. \end{cases}$

$\hat{G} = \{ A \rightarrow a \mid aaA \mid ababbAc \mid abbc.$

Identifiying USELESS production.

(1) $S \to aSb \mid \lambda \mid A$
$A \to aA$

Here $S \to A$ is useless because it will not produce a word because the variable A will be there in the production always.

(2) $S \to A$
$A \to aA \mid \lambda$.
$B \to bA$.

Here $B \to bA$ is useless because B is not reachable from S.

(3) $S \to aS \mid A \mid C$
$A \to a$
$B \to aa$
$C \to aCb$

$S \to C$ is useless because C never ends.
So $C \to aCb$ can also be removed.

Now, $B \to aa$ is useless because B is not reachable from S.

$$\hat{G} = \begin{cases} S \to aS \mid A \\ A \to a \end{cases}$$

(4) $S \rightarrow aS / AB$
$A \rightarrow bA$
$B \rightarrow AA$.

The entire grammar,
It does not give a useful production

So $\hat{G}$: $S \rightarrow \varphi$.

(5) $S \rightarrow a / aA / B / C$.
$A \rightarrow aB / \lambda$.
$B \rightarrow Aa$.
$C \rightarrow cCD$
$D \rightarrow ddd$.

Here ~~[illegible]~~ $S \rightarrow C$ is useless and because of it $C \rightarrow cCD$ and $D \rightarrow ddd$ become useless

$\hat{G}$ = $S \rightarrow a / aA / B$
$A \rightarrow aB / \lambda$
$B \rightarrow Aa$

Removing λ production .

(7) $S \rightarrow aS_1b$
$S_1 \rightarrow aS_1b / \lambda$.
$V_N \rightarrow$ Nullable set $\rightarrow$ it is a set of variable which yields λ production.

$V_N = \{S_1\}$

$\hat{G} = \begin{cases} S \rightarrow aS_1b \mid ab . \\ S_1 \rightarrow aS_1b \mid ab. \end{cases}$

Date
Page

$S \to ABaC$
$A \to BC$
$B \to b \mid \lambda$
$C \to D \mid \lambda$
$D \to d$.

$V_N = \{A, B, C\}$

$\hat{G}$ =
$S \to ABaC \mid BaC \mid AaC \mid ABa \mid aC \mid Aa \mid Ba \mid a$.
$A \to BC \mid B \mid C$
$B \to b$
$C \to D$
$D \to d$

(Q) $S \to a \mid aA \mid B$
$A \to aB \mid \lambda$
$B \to Aa$.

$V_N = \{A\}$.

$S \to a \mid aA \mid B$
$A \to aB$
$B \to Aa \mid a$.

Limitation :
$G \Rightarrow S \to aSb \mid \lambda$ $= \{\lambda, ab, aabb, \ldots\}$
$\Downarrow$
$\hat{G} \Rightarrow S \to aSb \mid ab$ $= \{ab, aabb, \ldots\}$

Here we are not able to produce λ in $\hat{G}$.
For λ production,
Hence we have to come to the limitation that,
$\hat{G} = G - \{\lambda\}$

Removing Unit production

Case 1 :-

if $A \to A$.
Then remove A without any checking.

Case 2 :-

$A \to B$
$B \to y_1 / y_2 / y_3$

use substution Rule :-

$A \to y_1 / y_2 / y_3$.

Case 3 :

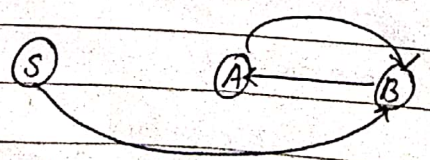
$A \to B$
$B \to A$

If you have transitivity rule draw dependency graph.

Eg:
1) $S \to Aa / B$
2) $B \to A / bb$
3) $A \to a / bc / B$

- Unit Productions are
1) $S \to B$
2) $B \to A$
3) $A \to B$.

dependency graph :-



$S \overset{*}{\Rightarrow} B$
$B \overset{*}{\Rightarrow} A$
$A \overset{*}{\Rightarrow} B$

$S \overset{*}{\Rightarrow} A$.

classmate

Date
Page

Now. Remove all the unit prodns.

$S \to Aa \mid bb \mid a \mid bc$
$B \to bb \mid a \mid bc$
$A \to a \mid bc \mid bb$.

Now B is not reachable from S.

Final G:-
$S \to Aa \mid bb \mid a \mid bc$
$A \to a \mid bc \mid bb$.

Rules :-
1) Remove λ production
2) Remove unit productions
3) Remove ~~that~~ useless productions

After the above 3 rules are followed then the resultant grammar will be in CNF (Chomsky Normal Form).

(2) $S \to aA \mid aBB$
$A \to aaA \mid \lambda$
$B \to bB \mid bbC$
$C \to B$

(1) Remove λ prod:

$V_N = \{A\}$

$\hat{G} =$ $S \to aA \mid aBB \mid a$
$A \to aaA \mid aa$
$B \to bB \mid bbC$
$C \to B$

Scanned by CamScanner

unit production   S   A $C \longrightarrow B$

$C \overset{*}{\Longrightarrow} B$

$S \rightarrow aA \,/\, aBB \,/\, a$

$A \rightarrow aaA \,/\, aa$

$B \rightarrow bB \,/\, bbC$

$C \rightarrow bB \,/\, bbC$.

~~Removing useless production~~

~~$S \rightarrow aA \,/\, aBB \,/\, a$~~

~~$B \rightarrow bB \,/\, bbB$~~

Removing useless production

B, C can be removed because it doesnot give terminal symbol.

$\{ L = a^{2n+1} ,\ n \geq 0 \}$. $- \ a(aa)^*$

$S \rightarrow aA \,/\, a$

$A \rightarrow aaA \,/\, aa$.

Q $S \rightarrow A \,/\, B$

$A \rightarrow \lambda$

$B \rightarrow aBb$

$B \rightarrow b$.

Simplify the given grammar.

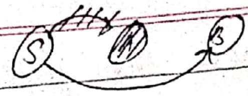
1) Removing λ prod.

$V_N = \{ A \}$.

$S \rightarrow B$

$B \rightarrow aBb$

$B \rightarrow b$



Rem. Unit Prod :-

~~$S \overset{*}{\Rightarrow} A$~~ $\qquad S \overset{*}{\Rightarrow} B$

~~$S \to aBb$~~

$S \to aBb \mid b$

$B \to aBb \mid b$

~~CNF :-~~

Q1 ① $S \to AS \mid a$

$A \to SA \mid b$

Q2 ② $S \to AS \mid AAS$

$A \to SA \mid aa$

CNF :-

Conds :- ① $S \to a$ (A var derives a terminal symbol)

② $S \to AB$ (A var derives 2 variables)

Q1 is in CNF

Q2 It is not in CNF.

$\Downarrow$

Changing it to CNF :

$S \to AS$

$S \to (AA)S$ → D_1 → another variable

$D_1 \to AA$.

$\Downarrow$

$S \to AS$	$A \to SA$
$S \to D_1 S$	$A \to aa$ → B_1
$D_1 \to AA$	$B_a \to a$

$\Downarrow$

$A \to SA$

$A \to B_1 B_1$

$B_1 \to a$

(Q) Convert the given grammar to CNF form. (10m)

$S \to AB \mid aB$
$A \to aab \mid \lambda$
$B \to bbA$.

(1) Remove λ prod.
$V_N = \{A\}$.

$G = S \to AB \mid aB \mid B$
$A \to aab$
$B \to bbA \mid bb$.

(2) Removing unit prod.

$\cancel{S} \to$

$S \overset{*}{\Rightarrow} B$

$S \to AB \mid aB \mid bbA \mid bb$
$A \to aab$
$B \to bbA \mid bb$

(3) Removing useless prod. (No useless prod).

$S \to AB$
$S \to aB$
$S \to bbA$
$S \to bb$