30/7/18 :- Unit-II :-

Regular expressions & languages :-

Regular language :- which is accepted by F.A

Regular language is taken in the form of simple expressions (operator & operands) is called regular expression.

union :- (operation)

$L(M) = \{ab, cd, ba\}$

$L(N) = \{bc, bd, ed\}$

$L(M) \cup L(N) = \{ab, cd, ba, bc, bd, ed\}$

↳ $L(M) + L(N)$ } different path
/ (or)

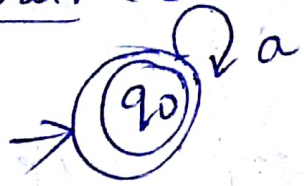
Concadination (denoted by (.)) → along the path

$L(MN) = \{abbc, abbd, abed, cdbc, cdbd, cded, babc, babd, baed\}$

$$L(M) \cdot L(N) = L(M \cdot N) = L(MN)$$

closure :- (star (or) kleene)

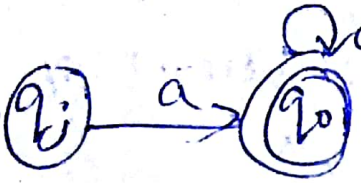
a^* =



$a^* = \{\in, a, aa, aaa \ldots\}$

positive :- subset of star

a^+ =



Regular Expression

The language accepted by F.A or Regular language & these languages are easily described by simple expressions called as Regular expressions

→ The class of languages accepted by F.A is precisely the class of language described by Regular expressions.

→ The Regular expressions can define exactly the various forms of the same languages that automata describe

The operations of R.E :-

1) union : The union of two languages L & M denoted by L∪M if L={001, 10, 111}, M={ϵ, 001}

L(L∪M) = {ϵ, 10, 001, 111} ... +1

2) concatination : The concatination of languages denoted by LM

L={001, 10, 111} M={ϵ, 001}

LM={001, 10, 111, 001001, 10001, 111001}

3) The closure (star (or) kleene) of a language L is denoted by L^*.

Note :-

*

Buildir

The

using

langua

⇒ UNIO

Finite

⇒ let

and

recu

1) An

2) ϕ

3) ϵ

epsil

4) For

the

5) If

then

$L^* = L^0 \cup L^1 \cup L^2 \cdots$ positive closure

Note:-

* $\phi^* = \{\phi^0, \phi^1, \phi^2 \ldots\} = \epsilon$

$\phi^0 = \epsilon$

Building R.E :-

The algebra of Regular Expression (R.E) follows, using constants & variables that denote languages, and operators for the 3 operations

⇒ UNION, DOT, STAR closure

Finite Automata & R.E :-

⇒ let 'Σ' be an alphabet the R.E over 'Σ' and the sets that they denote are defined recursively as follows. (Rules)

1) Any symbol from Σ, ϵ & ϕ are also R.E

2) ϕ is a R.E and denotes the emptyset

3) ϵ is a R.E and denotes the set {set of epsilon ϵ} '$\{\epsilon\}$'

4) For each 'a' in Σ 'a' is a R.E & denotes the set '$\{a\}$'.

5) If r & s are R.E denoting the language R&S then $(R+s)$, $(r.s)$, (r^*) are R.E that

denotes the sets (R∪S), RS, R^* respectively

6) The R.E denoted by applying Rule 1 to 5 ones or more than ones is also a R.E

Ques:-

1) If $\Sigma = \{a, b\}$ then 'a' is a regular expression from Rule 1 & Rule 4

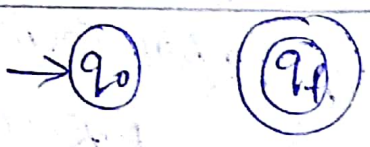
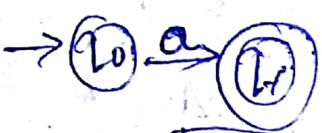
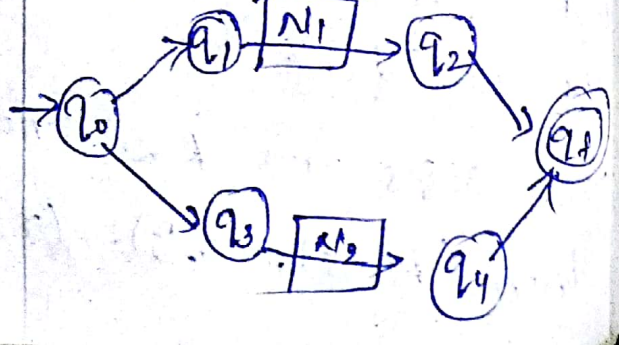
2) 'b' is a R.E from Rule 1 & 4

3) a+b → Rule 5

4) a^* → Rule 5

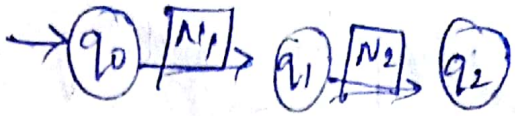
5) ab → Rule 5

6) ab + a^* → Rule 6

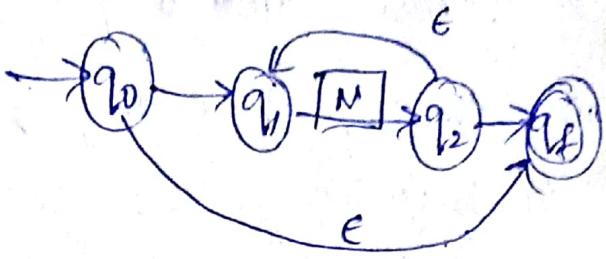
Regular expression	Regular set	Finite Automata
ϕ	$\{\}$	
ϵ	$\{\epsilon\}$	
a	$\{a\}$	
$R_1 + R_2$	$R_1 \cup R_2$	

N → Notations.

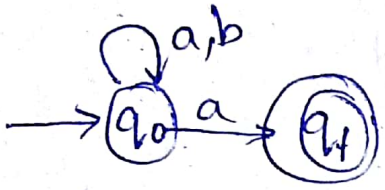
$L_1 L_2$ $R_1 R_2$



L^* R^*



⇒A language consists of all the words over an alphabet set $\Sigma = \{a, b\}$ ending with single 'a'

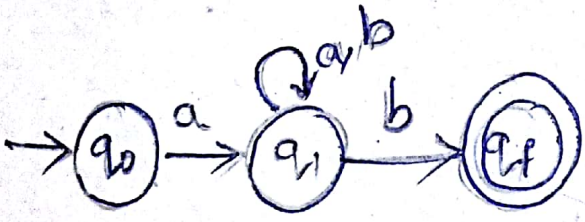


Regular Expression $RE = (a+b)^* \cdot a$

1/08/18 :-

⇒ Language consisting of all the strings over $\Sigma = \{a, b\}$ starting with 'a' & ending with 'b'

Sol:-



it should accept all the strings

R.E ⇒ $a(a+b)^*b$

2) Having a substring of aa

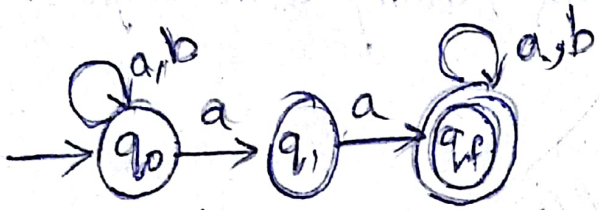
3) accepts all the strings except null string

4) starting & ending with 'a' & any no. of b in between

=

2) length of substring = 2

state = 3

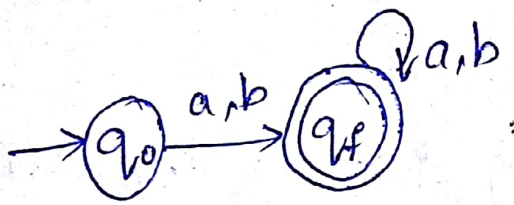


Regular expression :-

⇒ $(a+b)^* a a (a+b)^*$

---

3)



⇒ $(a+b)(a+b)^*$

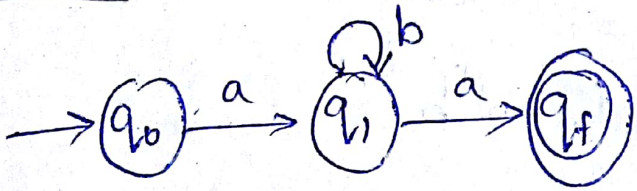
$= (a+b)^+$

RE :- $(a+b)^+$

Since null should not present it will be a positive closure

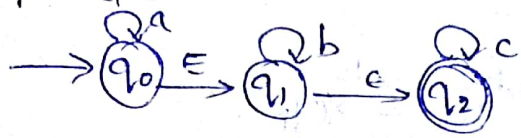
---

4)



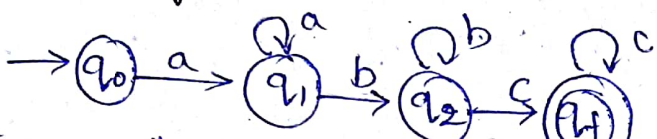
RE :- ab^*a

5) Language consisting of all the strings over $\Sigma = \{a, b, c\}$ any no of a's, followed by any no of b's & followed by any no of c's.



$RE = a^* b^* c^*$

6) Having atleast one 'a' followed by atleast one 'b' followed by atleast one 'c'



$RE = a a^* b b^* c c^*$

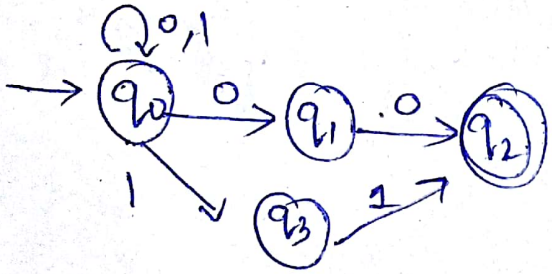
$= a^+ b^+ c^+$

$a \quad a^* \Rightarrow$ concadinate

$\{a\}\{\epsilon, a, aa, aaa \ldots\}$

$= \{a, aa, aaa\} = a^+$

7) accepts all the string end with either 00 or 11

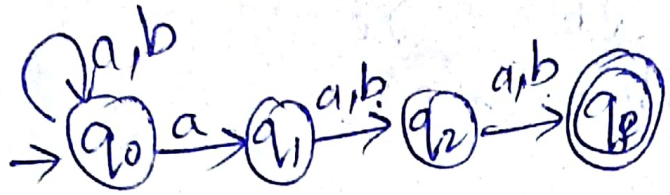


$= (0+1)^* (00+11)$

/

8) Third character from right is 'a'

(a) --



RE:-

$(a+b)^* a (a+b)(a+b)$

5/8/18:- Identities of Regular language:-

If R is a R.E then L(R) is a language repre-
sented for that R.E

1) $\phi + R = R + \phi = R$

2) $\phi R = R\phi = \phi$

3) $\epsilon R = R\epsilon = R$

4) $\epsilon^* = \epsilon$

$\phi^* = \epsilon \rightarrow \{\phi^0, \phi^1 \ldots\}$

$= \{\epsilon, \phi^1 \ldots\}$

5) $R + R = R$

6) $RR^+ = R^+R = R^+$

7) $(R^*)^+ = R^*$

8) $R^+R^+ = R^+$

9) $R(P+Q) = RP + RQ$; $[(P+Q)R = PR + QR]$

Convesion of DFA to R.E :-

Theorem If $L = L(A)$ for some DFA 'A' then there is a R.E $L = L(R)$

Proof :- Let us suppose 'A' has some 'n' no of states i.e $\{1, 2 \dots n\}$

Then Rename the states with first positive 'n' no's (no '0' included)

construct a collection of R.E that describe progressively border set of paths in 'A'

Let $R_{ij}^{(k)}$ as a name of a RE whose language is the set of strings 'w' such that 'w'

i = initial state

j = final state

k = how many no of states in FA intermediate state

is the label of paths from state i to state j in 'A' and that path has no intermediate node whose no is greater than 'k'

Note :- The beginning & end points of the path are not intermediate nodes

1) To construct expressions R.E $R_{ij}^{(k)}$ use following inductive defination

Starting at K=0 and finally reaching k=n

Basis :-

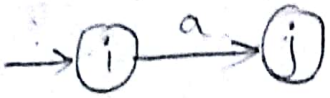
1) K=0 (since all states are numbered 1 or above the restriction on path is the path must have no intermediate states at all..

There are only 2 kinds of paths that means such condition

case 1) An arc from node i to node j

case 2) The path of length '0' that consists of only 1 node i

If $i \neq j$ then only case (1) is possible. we must examine the DFA and find those i/p symbols 'a' such that there is a transition from i to j on symbol a



step1 : if there is no such symbol 'a' then

$$R_{ij}^{(0)} = \phi$$

step2 if there is exactly one such symbol a.

step3 that

$i \neq$

If

arc

by

fix

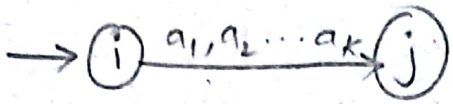
a) ϕ

b) a

c) ϵ +

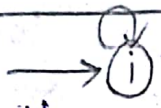
$R_{ij}^{(0)} = a$

Step3 if there are symbols $a_1, a_2, a_3 \cdots a_k$ that label are from i to j R_{ij}



$R_{ij}^{(0)} = a_1, a_2, a_3 \cdots a_k$

$i \neq j$

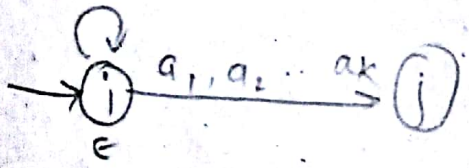


If $i=j$ then the legal paths of length = 0 and all loops from i to itself then the path of length = 0 is represented by 'ϵ'. So in the above cases the first case

a) $\phi + \epsilon$ (either ϕ or ϵ) $= \epsilon$ (state itself)

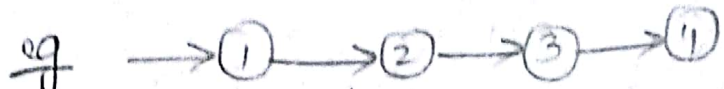
b) $a + \epsilon = a + \epsilon$ (because ϵ & a are proper path)

c) $\epsilon + a_1 + a_2 + \cdots a_k$



Induction :-

Suppose there is a path from state i to j that goes through no state higher than K

eg 

$R_{14}^{(2)}$ → intermediate K=4

2>4 X

There are 2 possible cases to consider.

Step1 The path doesnot go to K atall in this case the label of the path is in the language of $R_{ij}^{(k-1)}$

K=3 state going to get from i to j is 2 the correct one

$R_{ij}^{(k-1)} = (3-1) = 2$

Step2 The path goes to state K atleast one's then we can break the path into several pieces

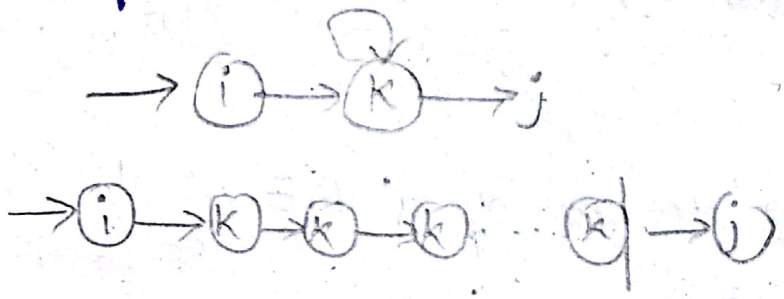
1> $i \rightarrow k$ } breaking into pieces

2> $k \rightarrow j$

First goes from $i \rightarrow k$

Second the last piece goes from $k \rightarrow j$ (not passing through k)

if the pieces in the middle go from K to itself



The set of labels for all paths of this type is represented by R.E

$$R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^{*} R_{kj}^{(k-1)}$$

↓ infinite no of types

when we combine the expressions for the path of 2 types of R.E

$$* \quad R_{ij}^{k} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^{*} R_{kj}^{(k-1)}$$

↓ either

Note :- If a path goes through state K only ones then there are no middle pieces it is having a path from i to k & K to j.

8/08/18 :-

Converting Regular Expression to Finite Automata

Theorem :-

Every language defined by a R.E is also defined by a F.A

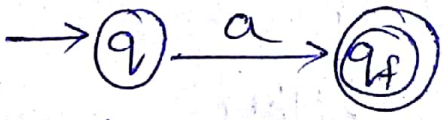
Suppose $L = L(R)$ for a R.E 'R' we show that $L = L(E)$ for some NFA with (ϵ)

NFA(ϵ) is 'E'

let 'r' be R.E over the alphabet set 'Σ'

Case (1) if $r = \phi$ → 

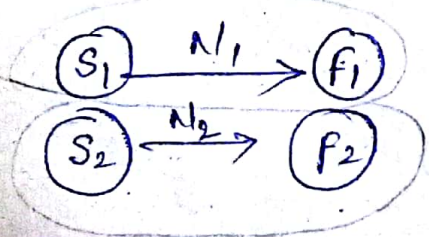
(ii) if $r = \epsilon$ → 

(iii) if $r = a$ → 

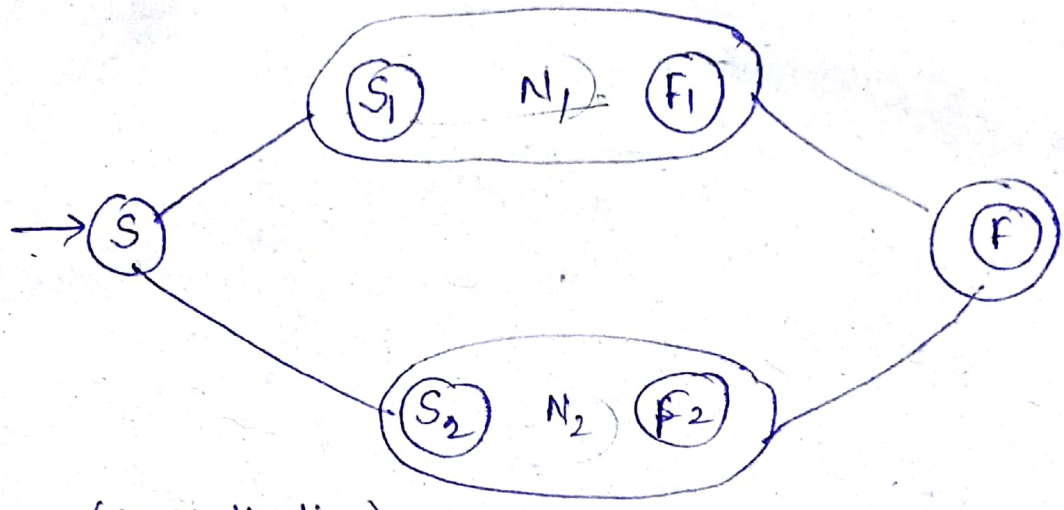
Case (2) :- * if $|r| \geq 1$ (↑ length of RE)

r_1 is a R.E having an F.A of N_1

r_2 is a N_2

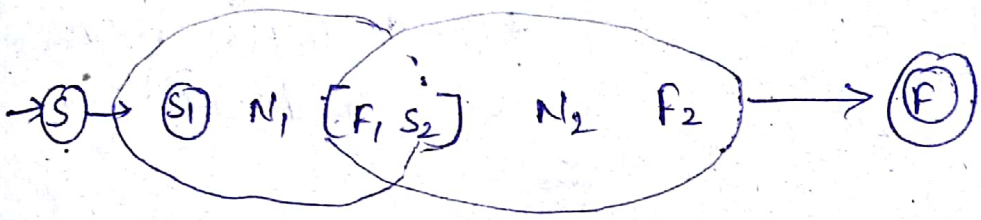


(i) i) $r_1 + r_2 \Rightarrow r_1 / r_2 \Rightarrow r_1 \cup r_2$

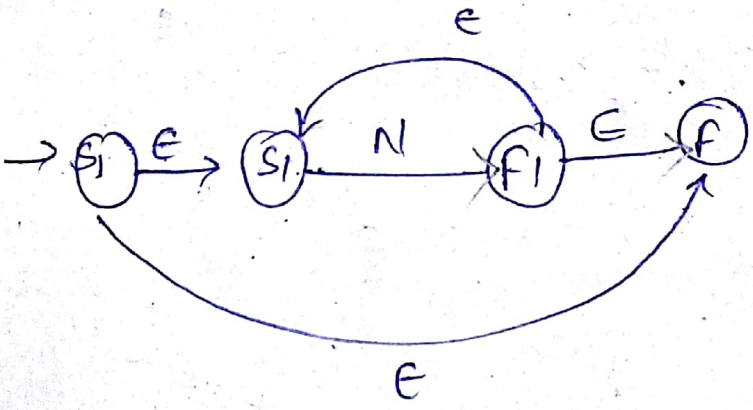


(Concatination)

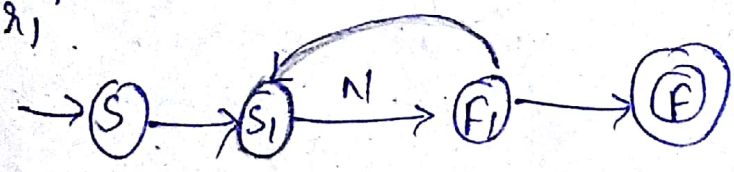
ii) $r_1 r_2 \Rightarrow r_1 \cdot r_2$



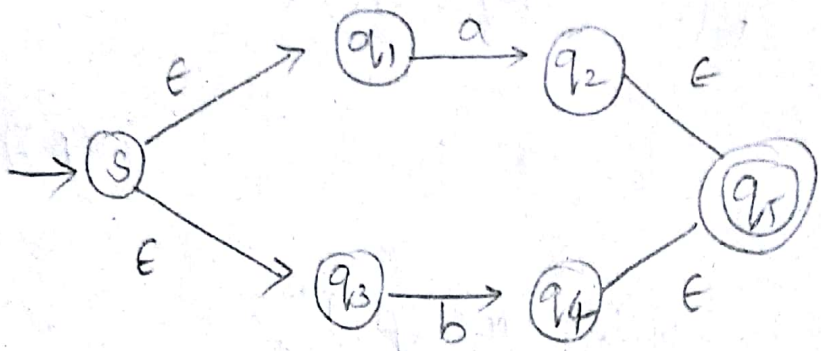
iii) r_1^*



iv) r_1^+



1) Construct NFA for R.E ; RE = a/b



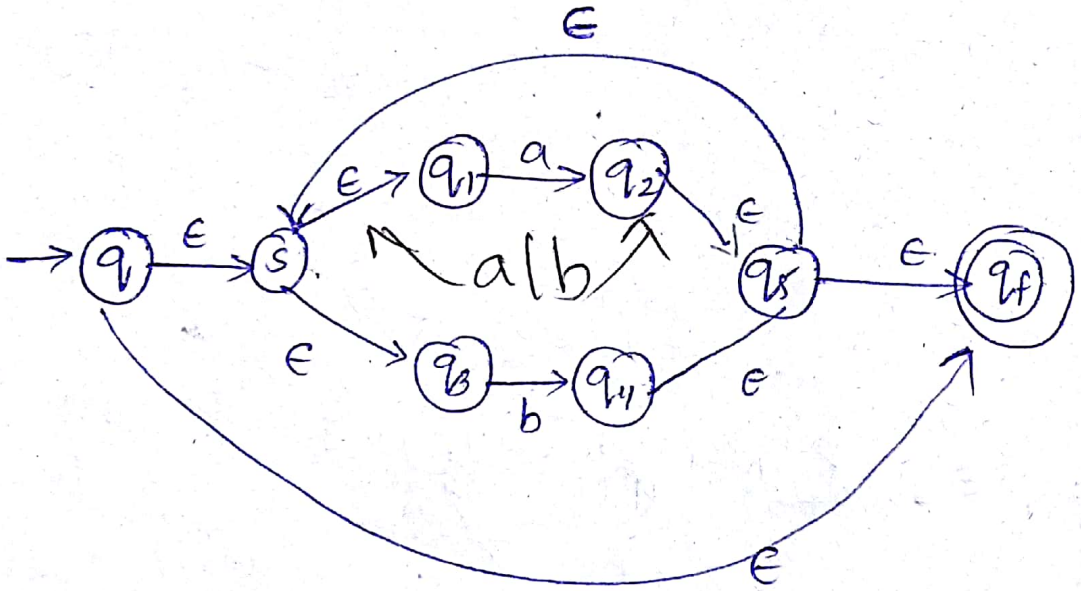
---

2) Construct NFA for R.E RE = (a/b)*

↓

for * initial to final

it should have ε

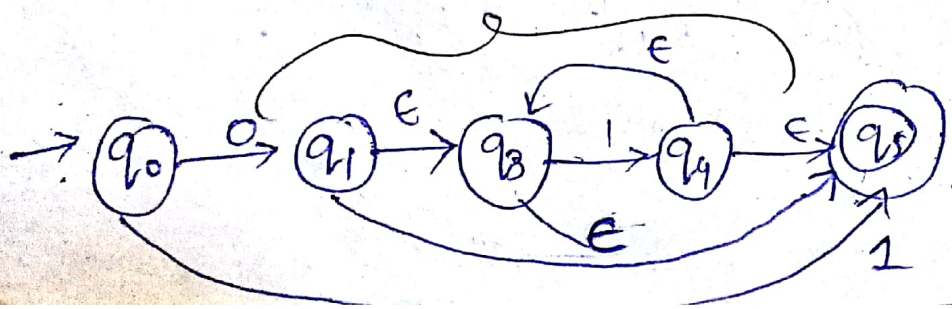


---

3) (O(1*)) + 1 = O1* + 1 O1* | 1

concadination ↓ union ↓ or

1*



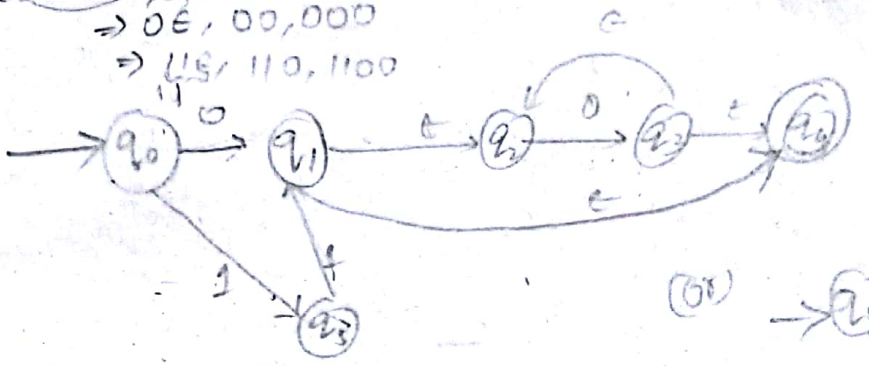
4) $(0+11)0^*$

$0^* = \{\epsilon, 0, 00, 000 \ldots\}$ ▷ So we can use self loop

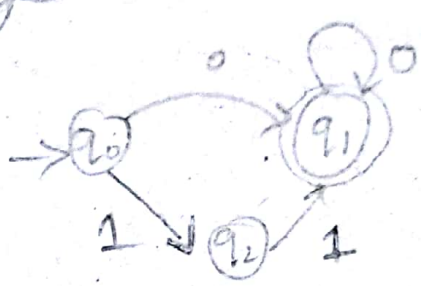
$= (0+11)(\epsilon+0+00\ldots)$

$\Rightarrow 0\epsilon, 00, 000$

$\Rightarrow 11, 110, 1100$



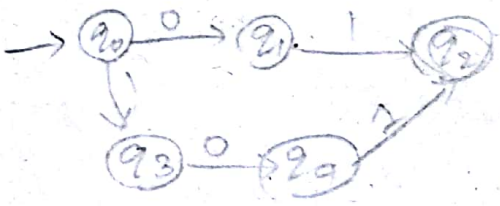
(or)



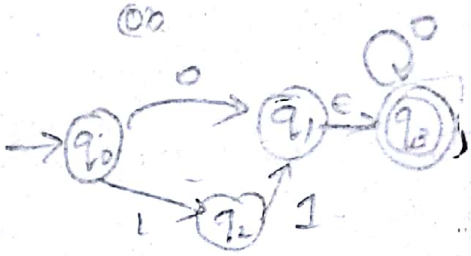
$\Rightarrow$ 1) $0, 00, 000 \ldots$

2) $11, 110, 1100 \ldots 110^n$

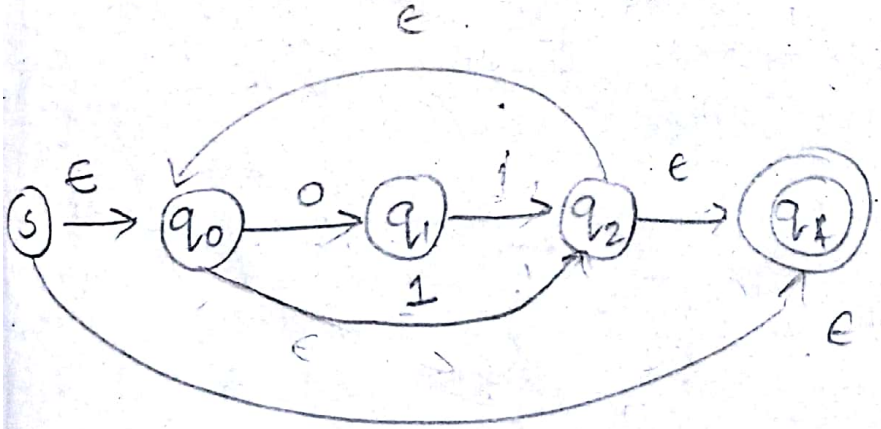
5) $01+101$



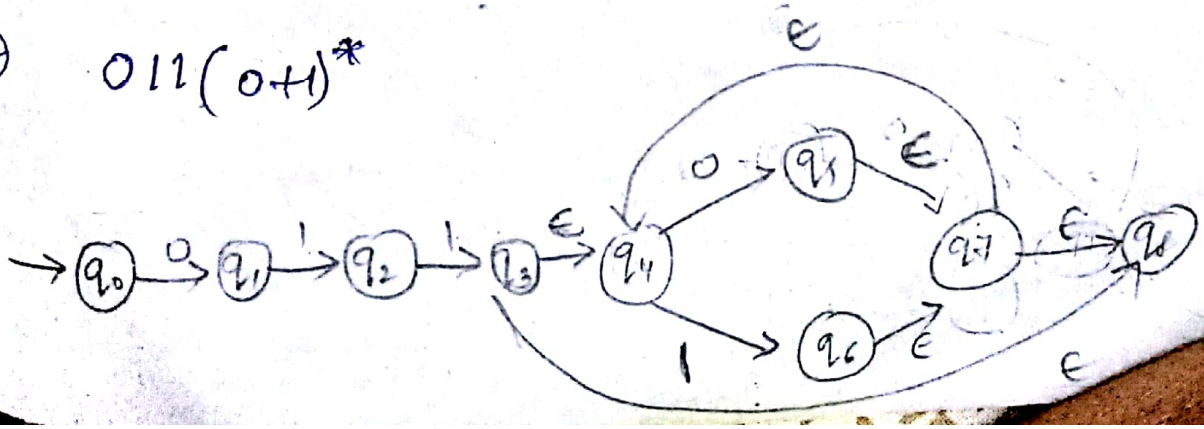
(or)



6) $(01+1)^*$

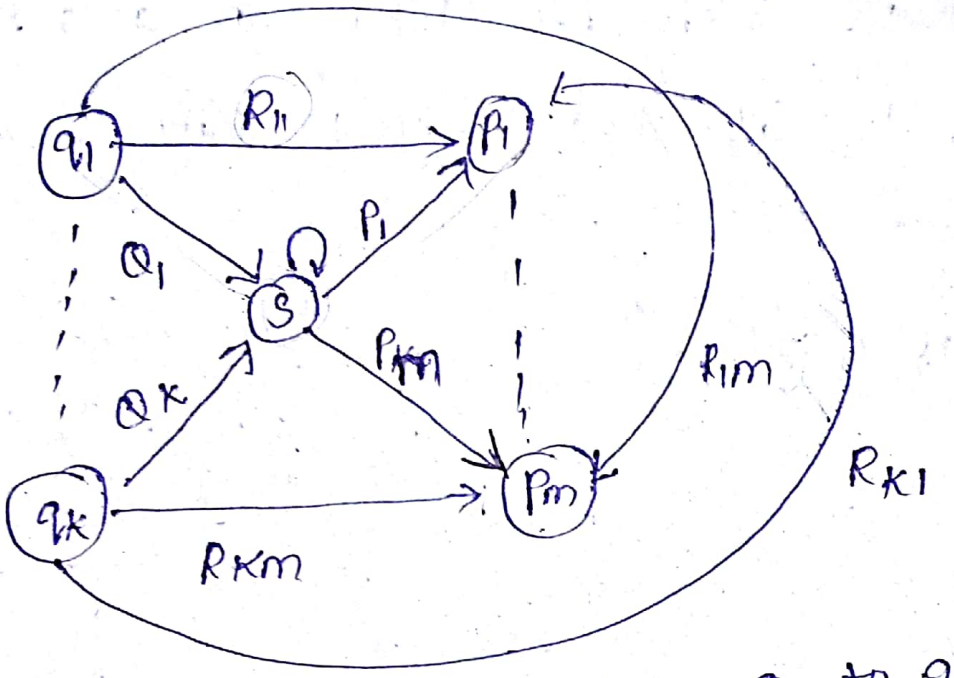


7) $011(0+1)^*$



9/08/18 :-

Converting DFA to RE by elimination of states:

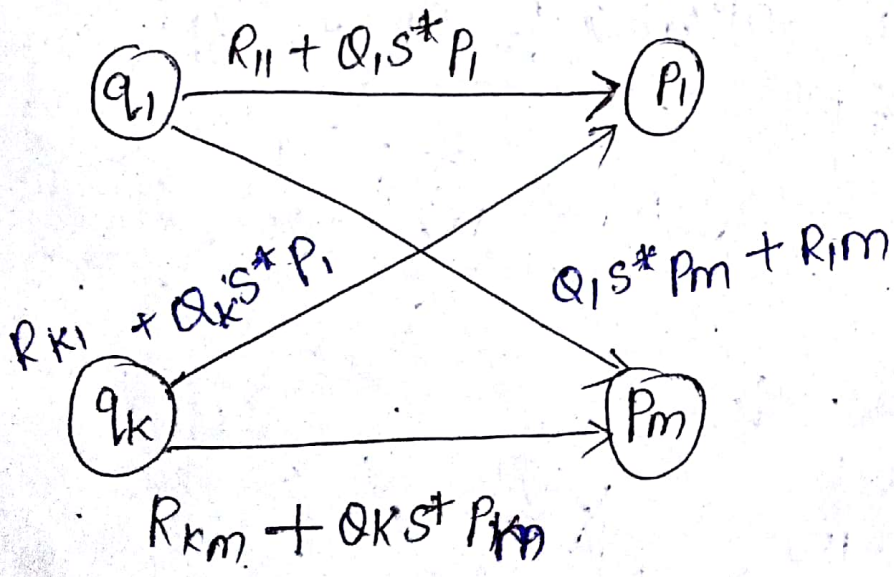


→ The initial state are from q_1 to q_k

→ final state p_1 to p_m

→ S is a intermediate state

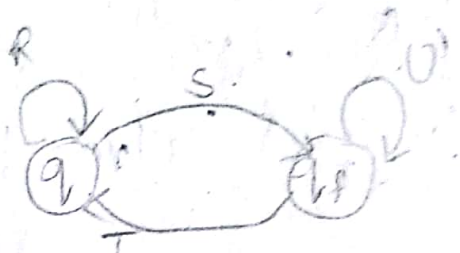
→ if we want to delete the 'S' intermediate state



→ ① for each accepting state q_i apply reduction process to produce an equivalent automata with R.E labels on arcs.

Eliminate all states except initial & final states.

→ ② if $q \neq q_f$

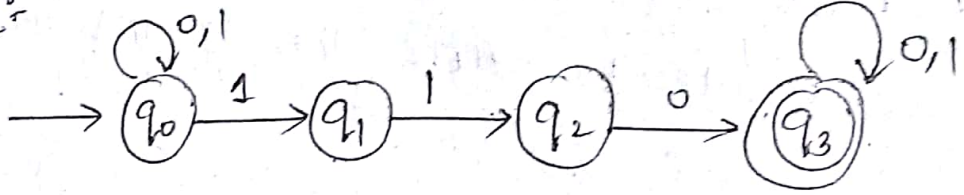


(4 possible cases only)

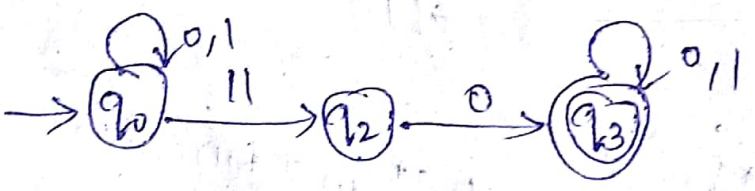
$$p^* + Q^* = (p+Q)^*$$

$$\Rightarrow (R + SU^*T)^* SU^*$$

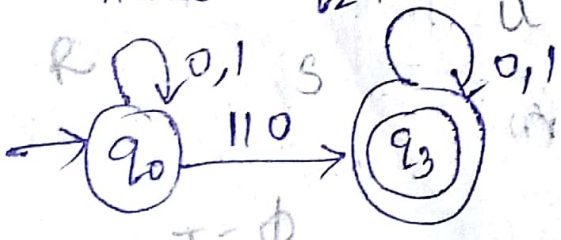
eg:-



eliminate intermediate nodes (q_1).



eliminate q_2.



$T = \phi$

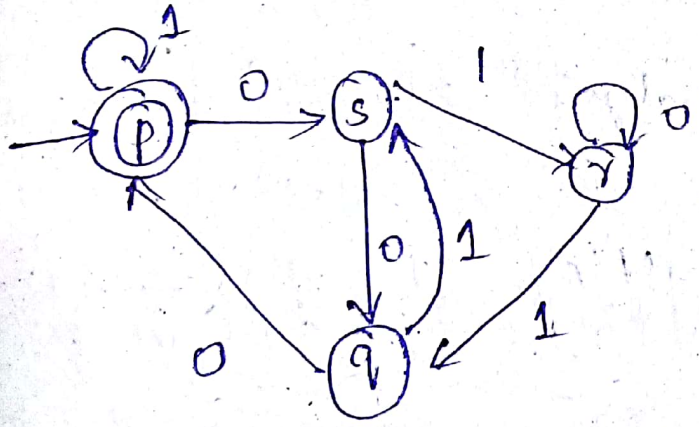
$= (R + SU^*T)^* SU^*$

$= ((0+1) + (110(0+1)^* \underbrace{\phi}))^* 110(0+1)^*$

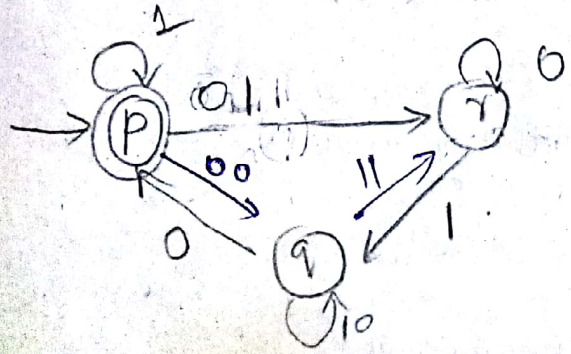
$= (0+1)^* 110(0+1)^*$

eg :-

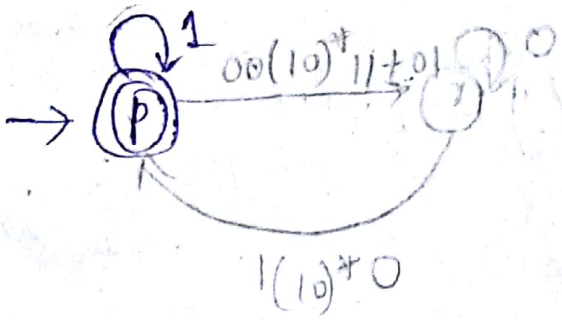
	0	1
→ * p	s	p
q	p	s
r	r	q
s	q	r



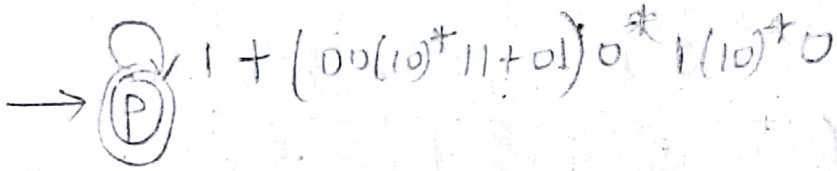
→ eliminate s



eliminate q



eliminate R



Applications of RE :-

→ RE in unix

→ lexical Analysis

→ pattern matching

Unix :-

→ (.) dot → any character

→ $[a_1, a_2 \cdots a_k]$ → $a_1 + a_2 + \cdots + a_k$.

→ [0-9] range [A-Z] upper case alphabets
(a-z) lower case alphabets

[A-Za-z0-9]

→ [+-0-9] → signed number

→ | → + union

→ ? → 0 (or) 1

→ + → 1 or more

→ n → n · no of elements

one phase of

↑

Lexical Analysis :-

else → {(return else)};

[A-Z] [A-Za-z0-9]*

>= {return GE};

= {return EG};

Pattern matching :

[A-Z a-z]

10/8/18 :- Algebraic laws of R.E :-
LMN → languages

Associativity communicative

Associative law :- $L+(M+N) = (L+M)+N$ → for union

$L(MN) = (LM)N$ → for concatination

(we should not change the order) $L(MN) = (MN)L$ ✗

Communicative law :- $L+M = M+L$

(only for union not for concadination)

Identities & Annililators :-

→ $\phi + L = L + \phi = L$

→ $\phi L = L\phi = \phi$ (concadination)

→ $\epsilon L = L\epsilon = L$

Distributive law :-

$L(M+N) = LM+LN$ } we should not change Order

$M+N(L) = ML+NL$

Idempotent law :-

$L+L = L$

laws involving closure :-

→ $(L^*)^* = L^*$

→ $\phi^* = \epsilon$, $\phi^+ = \phi$
↳ because it doesn't include Null value

→ $\epsilon^* = \epsilon$

→ $L^+ = LL^* = L^*L$

→ $L^* = L^+ + \epsilon$
↳ opeator taken as star closure

→ $L? = \epsilon + L$

→ $(L+M)^+ = (L^* + M^*)^+ = (L^*M^*)^+$

examples :-
→ R(L), R(M), R(N) all the Regular expressions

for communitative
eg L_1L_2 $L_1 = \{a, aa\}$
$L_2 = \{b, bb\}$

$L_1L_2 = \{ab, abb, aab, aabb\}$ ✓

$L_2L_1 = \{ba,$ } ✗ → not because it is from L_2L_1 $L_1L_2 = L_2L_1$ ✗

for idempotant law

eg: $L_1 = \{a, b\}$ $L_1 = \{a, b\}$

$L_1L_1 = \{aa, ab, ba, bb\}$ ✗ not possible concadination.

for laws involving closure.

$\phi^* = \{\phi^0, \phi^1, \phi^2 \ldots\}$

1. $\{\epsilon, \phi, \phi, \phi\}^*$

$\phi^* = \epsilon$

2nd unit - 2nd part

Properties of Regular (Languages) Expression :-

↓

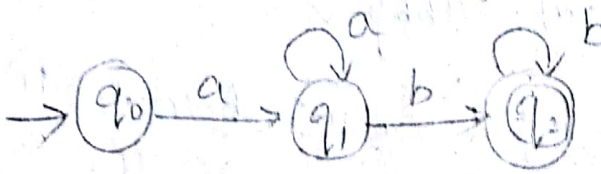
The language that is accepted by F.A

proving languages not to be regular :

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

n=1 n=2 n=3

{ab, aabb, aaabbb}



→ so this is for consecutive a & b's

here it should get equal a's & equal b's that not consecutive a & b's.

→ All languages are not Regular languages So here we uses a concept called

Pumping lemma → used to conform that languages are not regular

↳ This is a theorem & it has for all languages (CFL, CSL, UL).

For Regular language

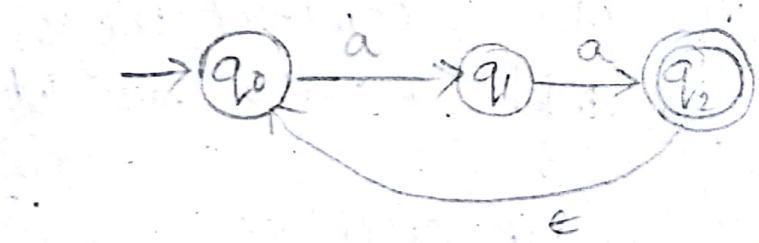
$$w = xy^iz$$

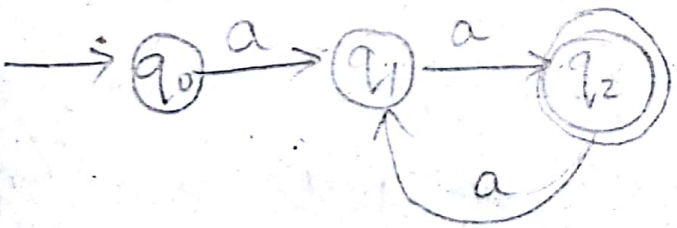
for context Free = $w = uv^ixy^iz$

→ It uses a principle called pigeonhole

⇒ $L = \{a^{2n} / n \geq 1\}$

$a^2 = aa$ $a^4 = aaaa$ $a^{2\times3} = aaaaaa$





Theorem :-

Let L be a R.L then there exist a constant 'n' for every string 'w' in L such that $|w| \geq n$, we can break w into 3 strings. $w = xyz$ such that we have to follow 3 conditions

(i) $y \neq \epsilon$

(ii) $|xy| \leq n$

(iii) for all 'K' the string xy^kz is also in L for $K \geq 0$

⇒ let the string

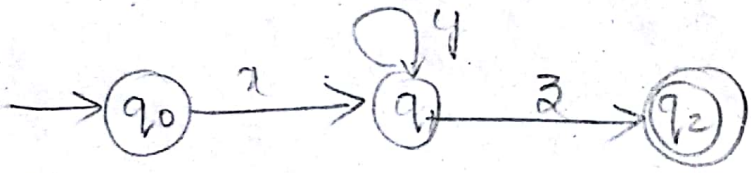
$w = a_1 a_2 a_3 \cdots | a_i a_{i+1} a_{i+2} \cdots | a_j a_{j+1} a_{j+2} \cdots a_m$

$x = a_1 \cdots a_i$ | $y = a_i a_{i+1} \cdots a_j$ | $z = a_j a_{j+1} a_{j+2} \cdots a_m$

we are dividing into 3 number of strings so we have to take 4 states but when $w = xy^kz$ we can take self loop

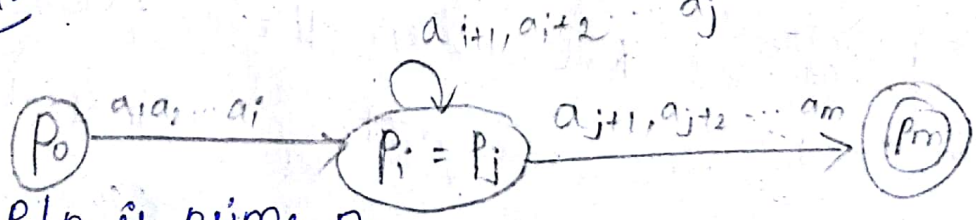
$L = a^p | p$ is prime no.

✗ not a regular language



$p_i = p_j$

13/08/18:-



$L = a^p | p$ is prime no

$L = \{ aa, aaa, aaaaa, \ldots \}$

2 3 5 7 11 13 17 19 23 .....

(i) $x=a$, $y=\epsilon$, $z=\epsilon$

$w = xy^i z$

$a\epsilon\epsilon = a$

(ii) $y=a \longrightarrow i=1$; $i=2$; $i=3$.

$\epsilon a\epsilon = a$ $\quad y=a^2$ $\quad y=a3$

$\epsilon aa\epsilon = aa$ $\quad \epsilon aaa\epsilon = aaa$

$L = \{a^n / n$ is even number$\}$

$w = xy^i z$ $\quad L = \{aa, aaaa, 6a, 8a, 10a \ldots\}$

$i=1$

let $x=a$ $\quad x=\epsilon, y=a, z=\epsilon$; $i=2$

$a\epsilon\epsilon = a$ ✗ $\quad \epsilon a\epsilon = a$ ✗ $\quad eaa\epsilon = aa$

$i=3$ ✗

$\epsilon aaa\epsilon = aaa$

$i \geq 2$ ✗ Not regular

$\Rightarrow L = \{ab^n\}$

$x=a$

$y=b \quad n=i=1$

$\{ab, abb, abbb, \ldots\}$ → it is regular language

Closure Properties of Regular Language :

1, Union
2, Intersection
3, Complement
4, Difference
5, Reversal
6, Closure
7, Concatenation
8, Homomorphic
9, Reverse - Homomorphic

Reversal :

$L_1 = \{ab, ac, ad\}$ Reversing each & every
$L_1^R = \{ba, ca, da\}$ element of language

Closure :

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

$L_1^* = \{\epsilon, a, b, ab, ba, aa, bb, \ldots\}$

Homomorphic :

1, $\Sigma = \{a, b\}$ $\Delta = \{0, 1, 2\}$

$h(a) = 0110$ $h(b) = 1002$

then :

$h(aba) = \underset{a}{0110}\ \underset{b}{1002}\ \underset{a}{0110}$

Reverse

Theorem :- 1

The union of regular languages

statement If L & M are R.L then so is $L \cup M$

proof : since L & M are regular they have R.E say $L = L(R)$ & $M = L(S)$ then $L \cup M = L(R+S)$

By the defination of '+' operator for R.E

Theorem :- 2 :

The intersection of two R.L

statement If L & M are R.L then so is $L \cap M$

proof : By Demorgan's law

$$L \cap M = \overline{\bar{L} \cup \bar{M}}$$

$$\Sigma^* - (\bar{L} \cup \bar{M})$$

$$\Sigma^* - ((\Sigma^* - L) \cup (\Sigma^* - M))$$

By the theorem of compliment

$\Sigma^* - L$, $\Sigma^* - M$ are Regular

if $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ is a Automata accepting 'L_1'.

and if $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ is a automata accepting L_2, then the automata accepting $L_1 \cap L_2$ will be regular

$L_1 \cap L_2$ is $M = (Q_1 \times Q_2, \Sigma, \delta, [q_1, q_2], F_1 \times F_2)$

Edit: Homomorphic of B = V

where $\delta((p,q_1),a) = [\delta_1(p,a), \delta_2(q_1,a)]$ but all the members of $Q_1 \times Q_2$ may not necessarily represent reachable state of "M"

Hence reduce amount of work, we start with a pair $[q_1,q_2]$ and find the transitions every member of Σ from $[q_1,q_2]$ to a new pair they only generate that pair, because it will represent a reachable state of "M"

We next consider the newly generated pairs to find out transitions for them

---

Theorem : 3

The compliment of R.L

Statement :- Start with a DFA and construct the

Step

a DFA and accept the compliment

Thus we could start with the R.E and find the R.E for its compliment

Step

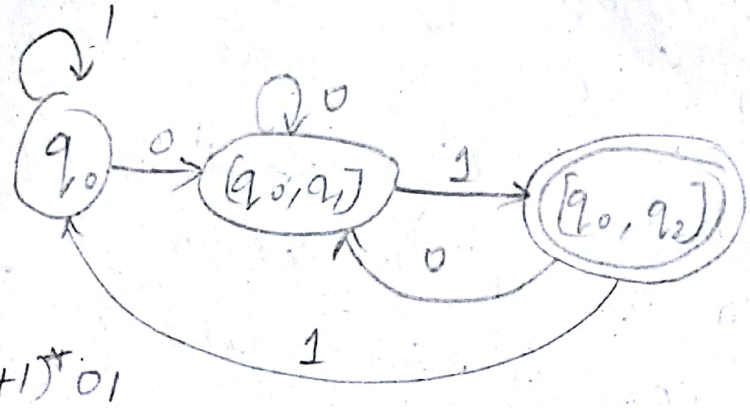
1) convert the R.E to an ϵ-NFA

2) convert the ϵ-NFA to DFA by subset construction

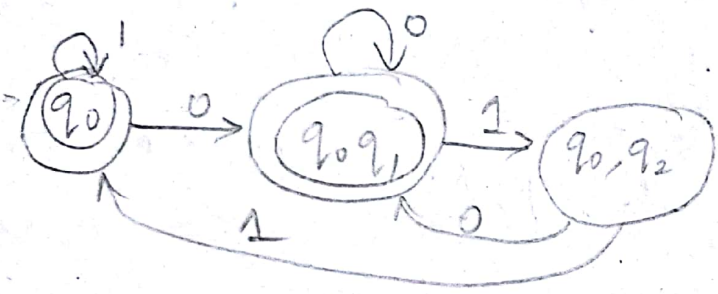
3) compliment the accepting states of that DFA

2)] but
t
of "M"
ve start
sitions
new
because
"M"
rated
?

ii) Turn the compliment DFA back into an R.E using the construction



$= (0+1)^*01$



↳ It does not accept 01 at final

Statement If L is a R.L over alphabet Σ then $\bar{L} = \Sigma^* - L$ is also a R.L

Proof :- Let $L = L(A)$ for some DFA A

$A(Q, \Sigma, \delta, q_0, F)$ then $\bar{L} = L(B)$ where B is the DFA with $B(Q, \Sigma, \delta, q_0, Q-F)$

If is B is exactly like A for the accepting state of A become non-accepting state of B & viavella

Then w is in $L(B)$ iff $\hat{\delta}(q_0, w)$ is in $Q-F$ which occcurs iff w is not in $L(A)$

Theorem 4 :-

The difference of two R.L is Regular

Statement of L & M are R.L then so is L-M

Proof :- Observe that $L-M = L \cap \bar{M}$ By compli-ment theorem $\bar{M}$ is Regular and by intersection theorem $L \cap \bar{M}$ is Regular which implies

L-M is Regular

Theorem 5 :

The Reversal of RL is Regular

Description :-

Given a language 'L' i.e L(A) for some F.A collapse with non determinism and ϵ - Transitions we may construct an Automata for L(R) by

Steps

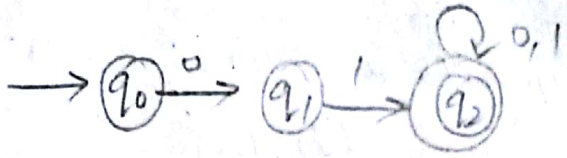
1) Reverse all the arcs in the Transition diagram of A

2) Make the start state of 'A' be the only accepting state for new automata

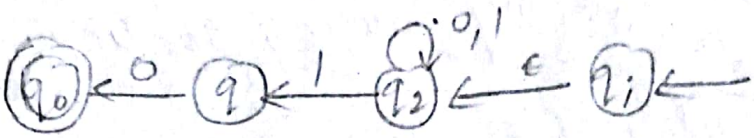
3) create a new start state p_0 with transition on ϵ to all the accepting states of 'A'

The result is an Automata that stimulates

a "inreversed" automata



Reverse



Theorem 5

statement :- If L is a R.L so is $L(R)$ (L^R)

proof :- As L is regular then we can construct 'M' then $L = L(M)$ when $M = (Q, \Sigma, \delta, q_0, F)$ then we construct $\overline{M}$ with reversing directions set initial state of M which are defines as final state of $\overline{M}$ M' and q_0 is the only final state of $\overline{M}$ $M' = (Q, \Sigma, \delta', F, \{q_0\})$

i.e $L = a^* b^*$

$L^T = b^* a^*$ is also regular

Decision properties of R.L :-

1) Emptiness property

2) membership property.

1) emptiness

↳ $L_1 \cup L_2$ [if both are empty] (L_1, L_2)

↳ $L_1 L_2$ [any one can be empty]

↳ $L_1^* = \{\epsilon, \rightarrow \text{empty}\}$

↳ $L = L_1$

2) membership :-

↳ (accepted for the language which is accepted by a F.A)

↳ if the string is accepted by the F.A of a language then that string is a member of that language

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

$L = \{ab, aabb, aaabbb \ldots\}$

Theorem(6)

Regular language are closed under closure

Statement :- If R is a regular set then R^* is always regular

Proof :- since R is a Regular set there exists FA

$M = (Q, \Sigma, \delta, q_0, F)$ accepting R

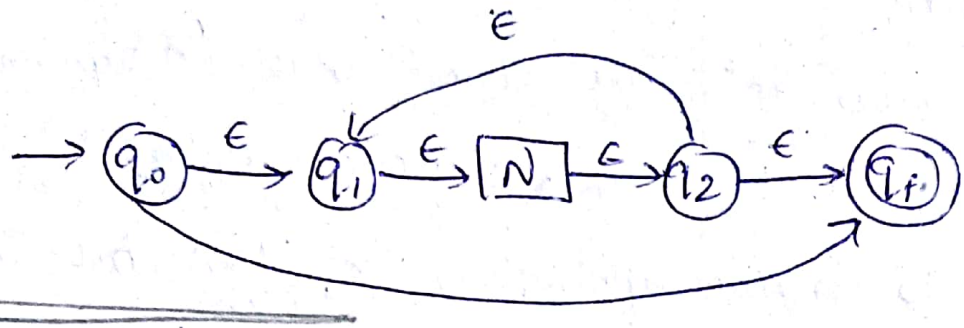
Let us construct FA M_1 using M as follows

$$M_1 = (Q \cup \{q_1, q_f\}, \Sigma, \delta_1, q_1, \{q_f\})$$

where $\delta_1(q_1, \epsilon) = \delta_1(q_f, \epsilon) = \{q_1, q_f\}$

$\delta_1(q, a) = \delta(q, a)$ for every 'q' in $Q - \{f\}$

and 'a' in $\Sigma_1 \cup \{\epsilon\}$



Theorem (7)

Regular languages are closed under concatenation

Statement If R_1 & R_2 are the regular then $R_1 \cdot R_2$ is always regular

Proof: R_1 & R_2 are regular sets, there exist FA

$M_1 = (Q_1, \Sigma_1, \delta_1, q_1, \{f_1\})$ & $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, \{f_2\})$

accepting R_1 & R_2 respectively

Let us construct 'M' using M_1 & M_2 as follows

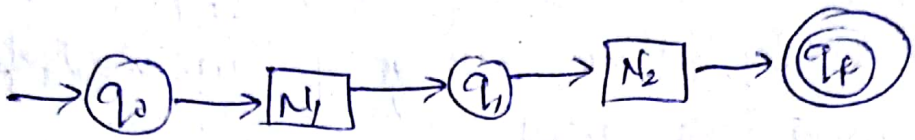
$$M = (Q_1 \cup Q_2, \Sigma_1 \cup \Sigma_2, \delta, q_1, \{f_2\})$$

where $\delta(f_1, \epsilon) = q_2$

$\delta(q, a) = \delta_1(q, a)$ for every q in $Q_1 - \{f_1\}$

& a in $\Sigma_1 \cup \{\epsilon\}$

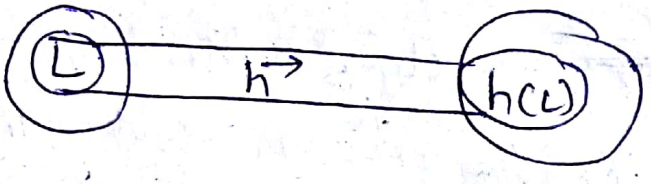
$\delta(q_1, a) = \delta_2(q_1, a)$ for every q in $Q_2 - \{f_2\}$ 1)
& a in $\Sigma_2 \cup \{\epsilon\}$



⑧ Theorem

The regular sets are closed under homomorphism 2)

Statement :- If L is a R.L over alphabet Σ and h is homomorphism on Σ then $h(L)$ is also regular



Proof :-

A homomorphism 'h' is a substitution such that $h(a)$ contains a string for each 'a'.

Suppose Σ & Δ are alphabets then a function $h : \Sigma \rightarrow \Delta^*$ is called homomorphism

If $w = a_1 a_2 \cdots a_n$

then $h(w) = h(a_1) h(a_2) \cdots h(a_n)$

If L is a language on Σ then homomorphic image is defined as

$h(L) = \{h(w) : w \in L\}$

1) $\Sigma = \{a, b\}$, $\Delta = \{a, b, c\}$ defined by

$h(a) = ab$, $h(b) = bbc$

$h(aba) = \underset{a}{ab}\ \underset{b}{bbc}\ \underset{a}{ab}$

→ Homomorphic image of $L = \{aa, aba\}$ is

$h(L) = \{abab, abbbcab\}$

2) $\Sigma = \{a, b\}$, $\Delta = \{0, 1, 2\}$

$h(a) = 2011$ $h(b) = 021$

$r = (a+b)^* (aa)^*$

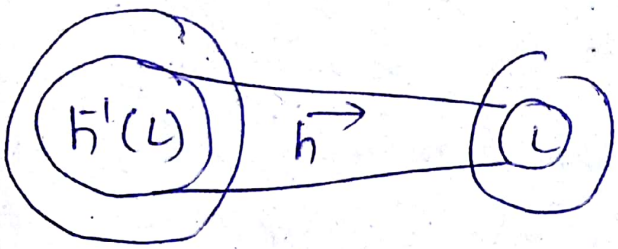
$= (2011+021)^* (2011.2011)^*$ denotes $h(L)$

Theorem ①

Inverse homomorphism of R.L is regular

Statement :- If h is homomorphism from alphabet Σ to alphabet T and L is R.L over T then $h^{-1}(L)$ is also R.L

Proof



Inverse homomorphism image of Language of L is defined as

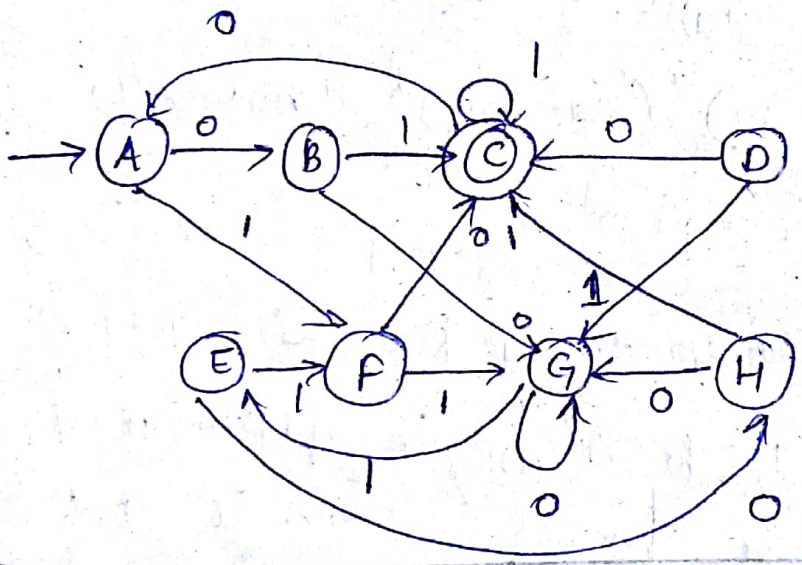
$h^{-1}(L) = \{x / h(x)$ is in $L\}$ for string w if is

$h^{-1}(w) = \{x / h(x) = w\}$

i.e suppose 'h' is a homomorphism from some alphabet Σ to strings in another alphabet T. Let 'L' be the language over alphabet T

Then $h^{-1}(L)$ is the set of strings 'w' in Σ^* such that $h(w)$ is in 'L'

20/8/18 :-



	A	B	C	D	E	F	G	H
A								
B	X BA							
C	X CA	X CB						
D	X DA	DB	X DC					
E	EA	X EB	X EC	X DE				
F	X FA	FE	X FC	X FD	X EF			
G	GA	X GB	X GC	X GD	EG	X GF		
H	X HA	X HB	X HC	HD	X EH	FH	X GH	

28/8/18 :-

Context free Grammer :-

↑Grammar

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

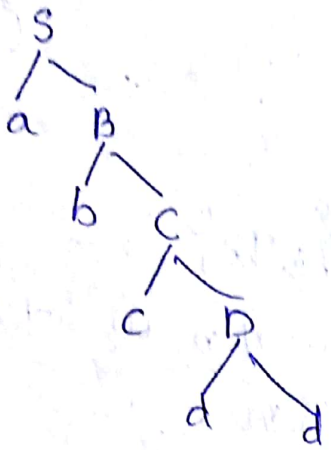
V = Variables (Non-Terminals)

T = Terminals

P = Production

S → start symbol (root)

eg:



Variables:- Non-leaf nodes (or non-terminal nodes in the tree. Represented by capital letters In the above tree, variables are: S, B, C, D. Tree does not terminate here

Terminals:- leaf node or terminal nodes in the tree Represented by small letters only In the above tree terminals are a, b, c, d. The tree terminates at the terminal

Start symbol:- Root node of the tree. The tree starts here. usually Represented by S

In the above diagram start symbol: S

productions:

Like transitions

Variable → variables

Variable → terminals

protection:

(productions) :-

Type 0, 1, 2, 3 differ by production set

production :-

(non terminal) Variable → Variable

Variable → Terminals

(leftside) it should contain → $(V \cup T)^*$ here we can take 'ε'

only variable

- only one variable:
  - V → V^*
  - V → T^*
  - V → V
  - V → T

eg: for unrestricted language

| here no 'ε'
any values

Problems

1) Construct the CFG for the language having any no of a's over the set $\Sigma = \{a\}$

Sol:- $G = (V, T, P, S)$

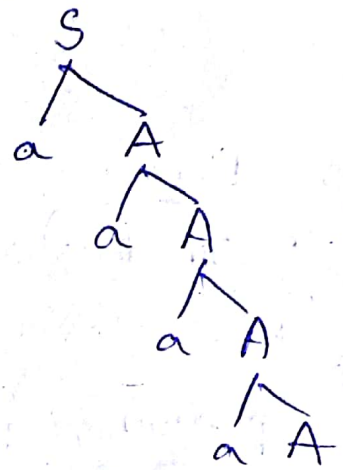
pictorial representation (PR)

$V = \{S, A\}$

$T = \{a\}$

$P = 3$ (rules) $S \to a$, $S \to A$, $A \to aA$

$S = S$



rules

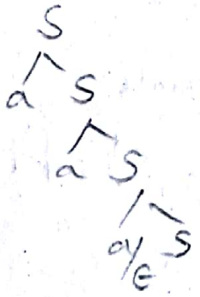
$S \to a$

$S \to A$

$A \to aA$

$S \to as$

$S \to \epsilon / a$

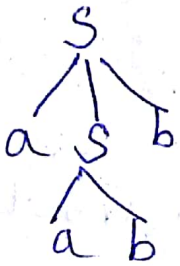


2) Try to recognize the language L for given CFG

$G = [\{S\}, \{a, b\}, P, \{S\}]$

P : $S \to asb$

$S \to ab$



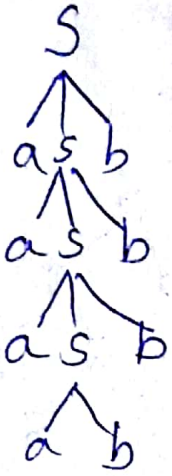
string (aabb)

here n ≥ 1

aaaa bbbb

⇒ $a^n b^n (n \geq 1)$

because no ϵ



3) construct the CFG for R.E $(0+1)^*$

Strings :- ϵ, 0, 1, 00, 11, 01 ...

$V = \{S\}$

$T = \{0, 1\}$

$P =$ 3. $S \to 0S \mid 1S \mid \epsilon$

$S = S$

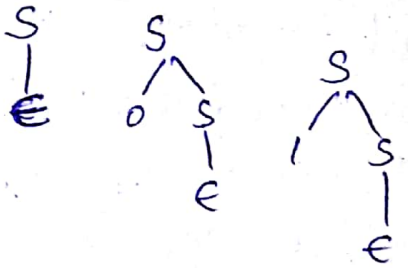
production :-

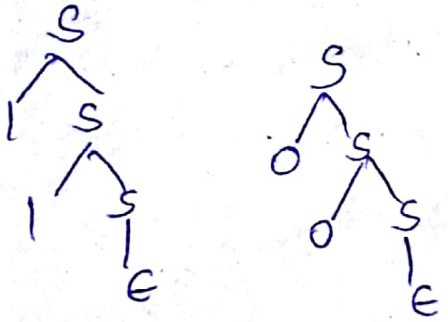
(3)

$S \to \epsilon$

$S \to 0S \mid 1S$

either (0 or 1)





4) construct a grammar for language containing strings of atleast two a's (ans of solution)

$V = \{S\}$

$T = \{a, b\}$

$P = 3(S \to aS$ (termination state) $S \to aa$ $S \to bS)$

$S = S$

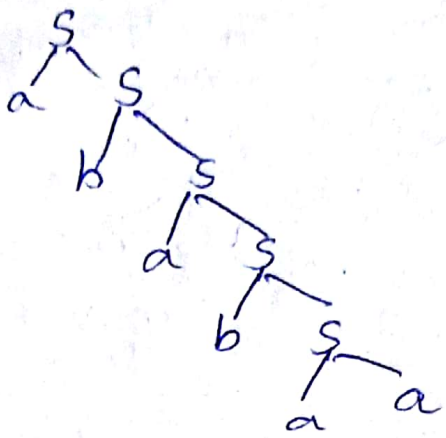
production :-

$S \to aS$

$S \to aa$

$S \to bS$

(or)



29/8/18 :-

5) $L = \{ w \, C \, w^T \}$ (C → Terminal)

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

$w = ab$

$w^T = ba$ (↓ Transpose)

If $w = a$, $w = b$ of length = 1

$L = \{ aca, bcb, abcba, bacab, aacaa, bbcbb \ldots \}$

(c → don't take production for it)

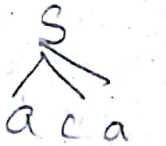
If length = 2

$w = ab$

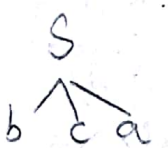
$w = ba$

$w = aa$

$w = bb$



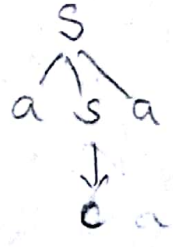
$S \to aca$



$S \to bca$

$S \to c$

replace c as s



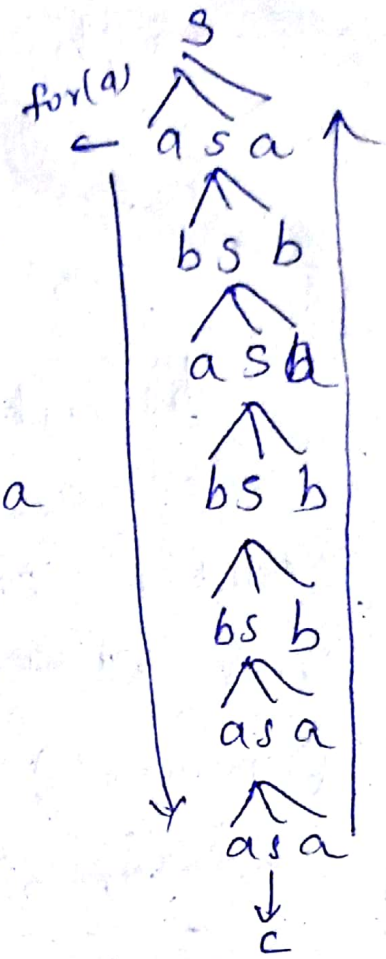
$S \to asa$ $S \to bsb$ $S \to c$

Production fn

$S \to asa, \; S \to bsb, \; S \to c$

Given string

ababbaacaabbaba



string accepted { ababbaaC
aabbaba

6) Construct CFG for language L which has all strings of palindromes over $\Sigma = \{a, b\}$

Here length of string is even

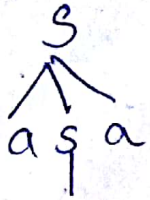
if middle element given then length is odd (eg:- $w c w^T$)

if length $L = 2$

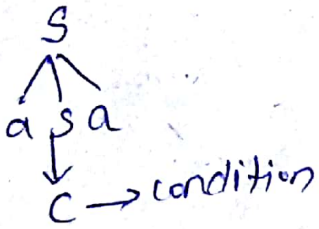
if $L = 1$
$L = \{a, b\}$

$L = \{aa, bb, aba, bab, aaa, bbb, \ldots\}$

$w = aa$



$\Rightarrow aa$ (even)



$\Rightarrow aca$ (odd)

aaa
↳ odd

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

↓ even palindrome

eg: baab (even)

babbabbab (odd) string length

⇒ construct a CFG which consists of all strings having atleast one occurence of 000 (consecutively) $\Sigma = \{0,1\}$

$S \rightarrow 0S0 \quad S \rightarrow 0 \mid 1$

⇒ 010 (X) no consecutive (000)

$L = \{000, 0001, 1000, 11000, 00011, \ldots\}$

I/ $S \rightarrow ABCDS000SABCD$

$A \rightarrow 0S0 \mid \epsilon$

$B \rightarrow 1S1 \mid \epsilon$

$C \rightarrow 0S1 \mid \epsilon$

$D \rightarrow 1S0 \mid \epsilon$

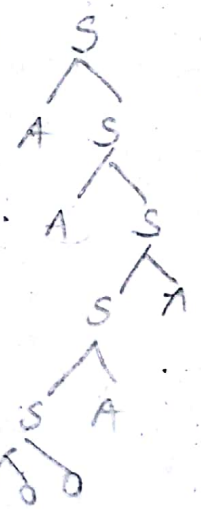
$S \rightarrow 0 \mid 1 \mid \epsilon$

II/ $S \rightarrow SA \mid AS$

$S \rightarrow 000$

$A \rightarrow 0 \mid 1$

(string) 1100011



8) Recognize the language

$S \rightarrow aB \mid bA$

$A \rightarrow a \mid as \mid bAA$

$B \rightarrow b \mid bs \mid aBB$.

9) Different first & last symbol in string.

5|09|18

construct CFG for the language in which there are No consecutive b's?

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

length 1	length 2	length 3
→ b	→ ba	bab
a	ab	.....
	aa	aba
		aab
		baa
		aaa

$S \rightarrow as \mid bsa$

$S \rightarrow as$

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