Unit 2

DATE
PAGE

TOC

Notes By Sonam

Regular Expressions :-

⇒ is a way of showing how a regular language is built from base set of regular languages.

⇒ Symbols are identical which are used to construct languages & any given expression that has a language closely associated with it.

⇒ For each regular exp (E), there is a regular language L(E).

⇒ Let 'R' be a RE over alphabet Σ if R is a:

a.) ϵ is a RE denoting set $\{\epsilon\}$

b.) ϕ " " " empty set $\{\}$

c.) for each symbol $a \in \Sigma$, a is RE denoting set $\{a\}$

d.) Union of two RE is also regular.

e) Concatenation of two RE is also regular.

f) Kleene Closure * of RE is also regular.

g) if R is regular, (R) is also regular.

* $R = \epsilon$ $\quad L(R) = \{\epsilon\}$
- $R = \phi$ $\quad L(R) = \{\}$
- $R = a$ $\quad L(R) = \{a\}$
- $R_1 \cup R_2$ = Regular
- $R_1 \cdot R_2$ = regular.
- $a \cup b = \{a, b\}$
- $a^* = \{\epsilon, a, aa, aaa ----\}$

RE for finite lang:-

Eg:- $(a+b)^*$ → collection of all strings of ab.

Q:- Language with no string.

$\{\}$, ϕ

⇒ Language with length 0.

$\{\epsilon\}$, ϵ, λ

⇒ Length 1 $\quad \{a, b\}$

$(a+b)$ (means a or b)

⇒ Length 2 $\{aa, ab, ba, bb\}$
or $\{aa + ab + ba + bb\}$
or $a(a+b) + b(a+b)$
or $(a+b) \cdot (a+b)$

⇒ Length 3 $(a+b)(a+b)(a+b)$

⇒ Atmost 1 0 or 1
$\{\epsilon, a, b\}$
$RE = \epsilon + a + b$

⇒ Atmost 2 $(\epsilon + a + b)(\epsilon + a + b)$

⇒ RE for not more than 2 b's and 1 a.
$\{\epsilon, a, b, ab, ba, abb, bab, bba - - - -\}$
$RE = \epsilon + a + b + ab + ba + abb + bab + bba$.

RE for infinite language :-

⇒ All strings having single b
$a^* b a^*$

⇒ All strings having atleast one b
$(a+b)^* b (a+b)^*$

⇒ All strings having bbbb as substring
$(a+b)^* bbbb (a+b)^*$

⇒ All strings end with ab.

$(a+b)^*ab$

⇒ All strings start with ba

$ba\,(a+b)^*$

⇒ All strings beginning & end with a

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

⇒ All strings containing a

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

⇒ All strings starting & end with different i/p.

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

Operations on RE :-

⇒ Union :- $L1=\{(1+0)\cdot(1+0)\}$

$L2=\{\epsilon, 100\}$

$L1 \cup L2 = \{\epsilon, 00, 10, 11, 01, 100\}$

⇒ Concatenation :- $L1=\{0,1\}$

$L2=\{00,11\}$

$L1\cdot L2 = \{000, 011, 100, 111\}$

DATE
PAGE

⇒ Kleene closure :-

$L1 = \{0,1\} - \{\epsilon, 0, 1, 00, 01, 10, 11 \ldots\}$

L^* → all strings including null value.

Algebric properties of RE :-

Kleene closure * → unary operator

Union (+) & concatenation operator (.) → binary operators

1. Closure :- r1 & r2 are RE then,

⇒
- r1* is a RE.
- r1 + r2 is a RE.
- r1 . r2 is a RE.

2. Closure laws :-

⇒ $(r^*)^* = r$

⇒ $\phi^* = \epsilon$

⇒ $r^+ = r \cdot r^* = r^* r$

⇒ $r^* = r^* + \epsilon$

3. Associativity :-

i) $r1 + (r2 + r3) = (r1 + r2) + r3$

ii) $r1 \cdot (r2 \cdot r3) = (r1 \cdot r2) \cdot r3$

4. Identity :-

i) if $r + x = r$

$\Rightarrow x = \phi$

as $r \cup \phi = r$ $\therefore \phi$ is identity for +.

ii) if $r \cdot x = r$ for $x = \varepsilon$

$r \cdot \varepsilon = r \Rightarrow$ ε is identity for (·).

5. Annihilator :-

if $r + x = r$

$\Rightarrow r \cup x = x$

if $r \cdot x = r$ when $x = \phi$

then $r \cdot \phi = \phi$

So ϕ is annihalator for (·) operator.

6. Commutative property :-

$r1 + r2 = r2 + r1$

$r1 \cdot r2 \neq r2 \cdot r1$

7. Distributed property:-

$(r1+r2)\cdot r3 = r1\cdot r3 + r2\cdot r3$ → Right Distribution

$r1\cdot(r2+r3) = r1\cdot r2 + r1\cdot r3$ → Left Distribution

$(r1\cdot r2)+r3 \neq (r1+r3)(r2+r3)$

8. Idempotent law:-

$r1 + r1 = r1$ } union operator satisfies idempotent property

$\Rightarrow r1 \cup r1 = r1$ }

but $r\cdot r \neq r$

Identities for regular expression:

Let p, q, r are regular exp. :-

$\Rightarrow \phi + r = r$

$\Rightarrow \phi \cdot r = r \cdot \phi = \phi$

$\Rightarrow \epsilon \cdot r = r \cdot \epsilon = r$

$\Rightarrow \epsilon^* = \epsilon$ and $\phi^* = \epsilon$

$\Rightarrow r + r = r$

$\Rightarrow r^* \cdot r^* = r^*$

$\Rightarrow r \cdot r^* = r^* \cdot r = r^+$

$\Rightarrow (r^*)^* = r^*$

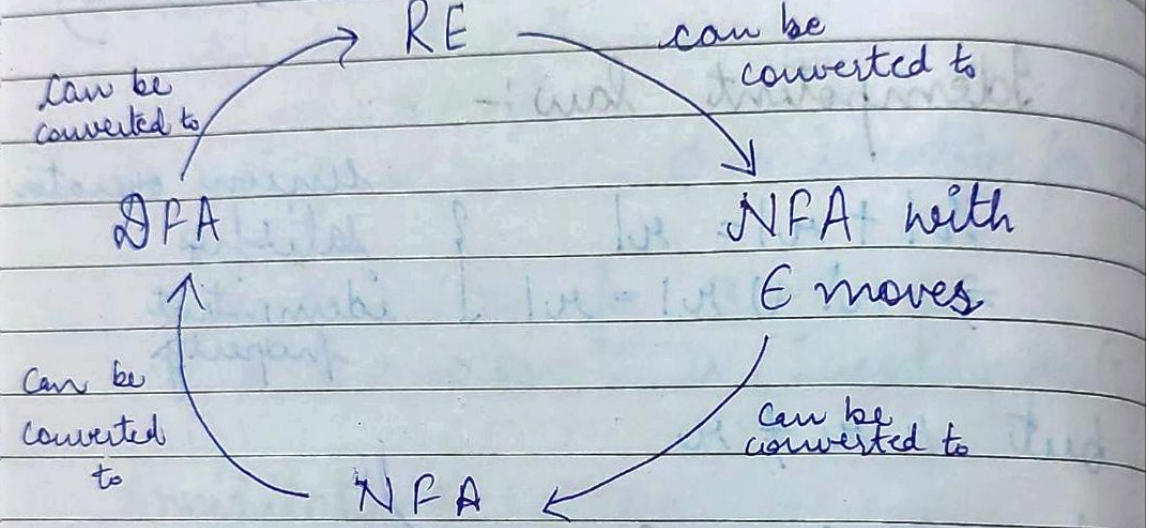
$\Rightarrow \epsilon + r \cdot r^* = r^* = \epsilon + r^* \cdot r$

$\Rightarrow (p\cdot q)^* \cdot p = p \cdot (q\cdot p)^*$

$\Rightarrow (p+q)^* = (p^*\cdot q^*)^* = (p^* + q^*)^*$

⇒ $(p+q)\cdot r = p\cdot r + q\cdot r$ and
$r\cdot(p+q) = r\cdot p + r\cdot q$

Finite Automata & RE :-



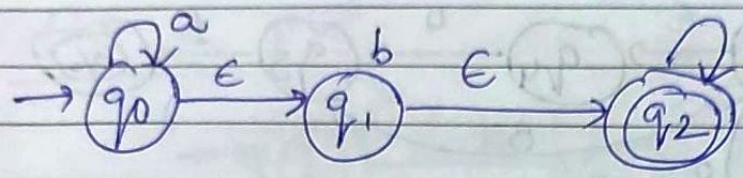
⇒ RE to NFA with ϵ moves
⇒ NFA with ϵ moves to without ϵ mov
⇒ NFA without ϵ moves to DFA
⇒ DFA to RE.

ε-NFA (Epsilon NFA)

DATE
PAGE

$(Q, \Sigma, q_0, \delta, F)$

$\delta: Q \times (\Sigma \cup \varepsilon) \to 2^Q$

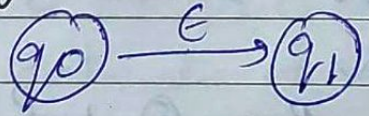


Here 3 states are there,

so $2^3 = 8$ transitions are possible.

(When we are not sure about current state, then we can do an epsilon move).

Eliminating ε moves :-



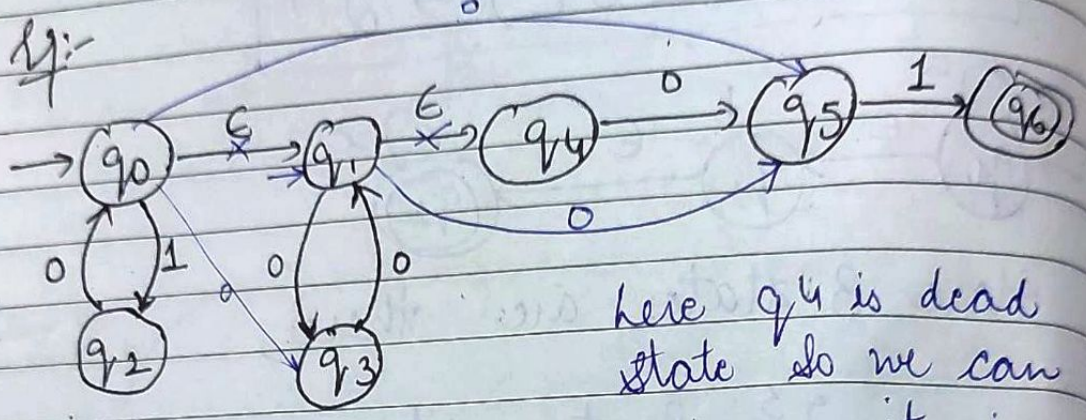
⇒ find all edges starting from q_1.

⇒ Duplicate all edges to q_0 without changing edge labels.

⇒ Changing edge labels.
↳ if q_0 is initial state make q_1 initial.

if q_1 is final state, make q_0 final
⇒ remove dead state

Ex:-



Here q_4 is dead state so we can remove it

Arden's Theorem:-

Let P & Q be two RE.

If P does not contain null string, then $R = Q + RP$ has unique solution $R = QP^*$

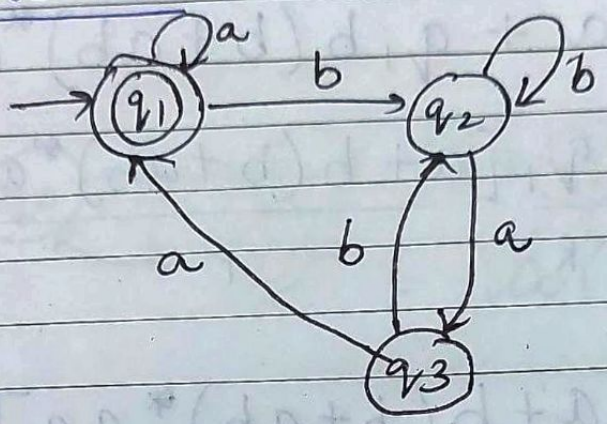
Proof:-

$$R = Q + RP$$
$$= Q + (Q + RP)P$$
$$= Q + QP + RP^2$$
$$= Q + QP + (Q + RP)P^2$$
$$\vdots$$
$$= Q[\epsilon + P + P^2 + P^3 + \cdots]$$

So $R = Q + RP$

$\Rightarrow \boxed{R = QP^*}$

FA to R.E :-



(see how to reach q_1) $(q_1, a) \to q_1$

$(q_3, a) \to q_1$

so, $q_1 = q_1 a + q_3 a + \epsilon \quad \to (1)$

$q_2 = q_1 b + q_2 b + q_3 b \quad \to (2)$

$q_3 = q_2 a \quad \to (3)$

Find value of final state i.e q_1.

Substitute values.

$q_2 = q_1 b + q_2 b + q_2 ab$

$\underset{R}{q_2} = \underset{Q}{q_1 b} + \underset{R}{q_2} \underset{P}{(b + ab)}$

$$\Rightarrow q_2 = q_1 b(b+ab)^* \rightarrow (4)$$

Now,

$$q_1 = q_1 a + q_2 aa + \epsilon$$

$$\Rightarrow q_1 = q_1 a + q_1 b(b+ab)^* aa + \epsilon$$

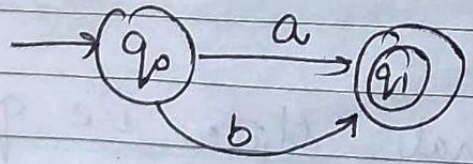
$$\Rightarrow \underset{R}{q_1} = \underset{Q}{\epsilon} + \underset{R}{q_1}\underbrace{[a+b(b+ab)^* aa]}_{P}$$

$$\Rightarrow q_1 = \epsilon[a+b(b+ab)^* aa]^*$$

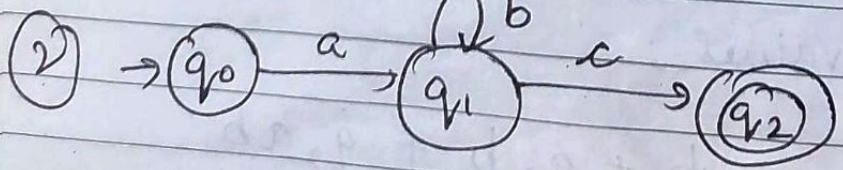
$$\Rightarrow q_1 = [a+b(b+ab)^* aa]^*$$

$\longleftrightarrow$

Examples:- FA to RE

(1) 

$= (a+b)$

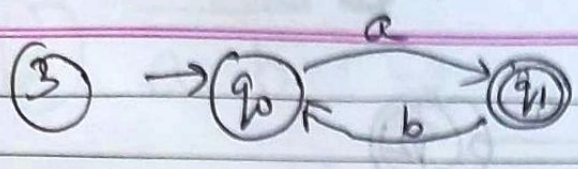
(2) 

$= ab^* c$

whenever initial state is as final state also, then it can accept null string also.

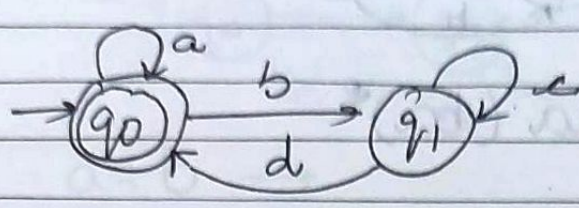
DATE
PAGE

(3)



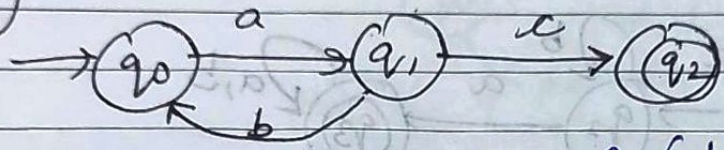
$= a(ba)^*$

(4)



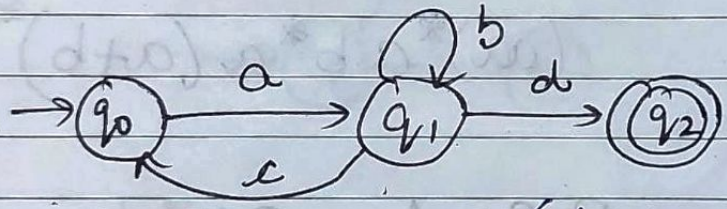
$= [a+bc^*d]^*$

(5)



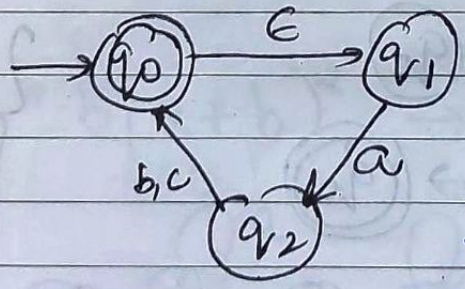
$= a(ba)^*c$

(6)



$= a(b+ca)^*d$

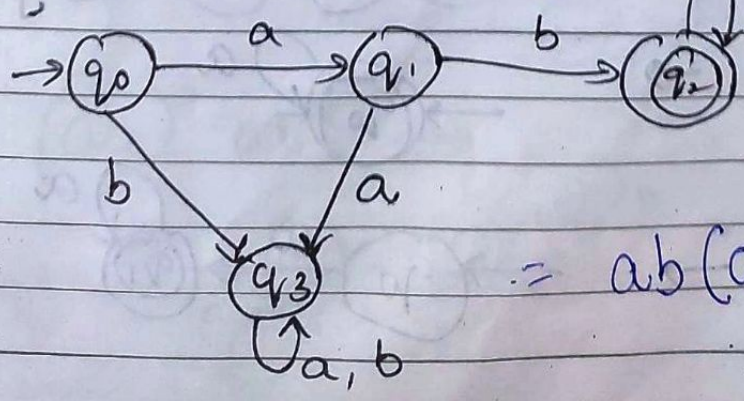
(7)



$= [\epsilon a(b+c)]^*$

$= [a(b+c)]^*$

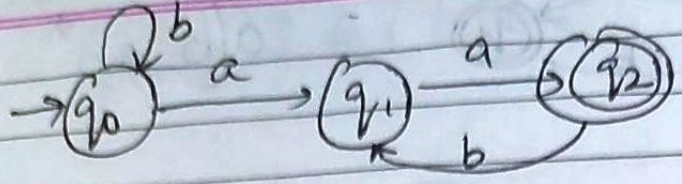
(8)



$= ab(a+b)^*$

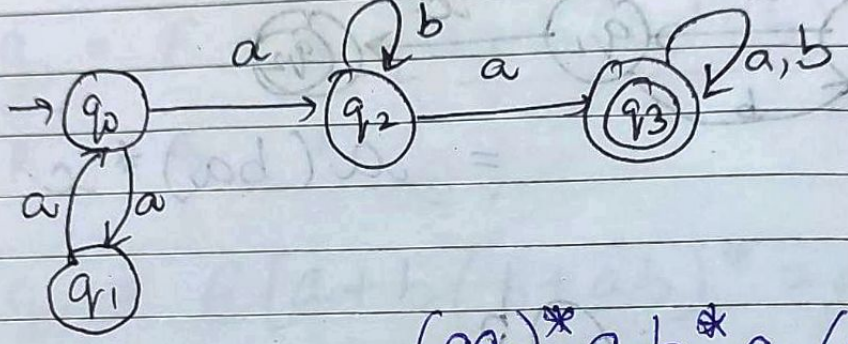
DATE
PAGE

(9)



$= b^* aa (ba)^*$

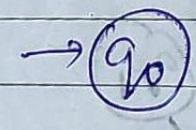
(10)



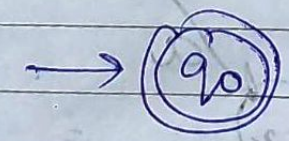
$(aa)^* ab^* a (a+b)^*$

Examples RE to FA

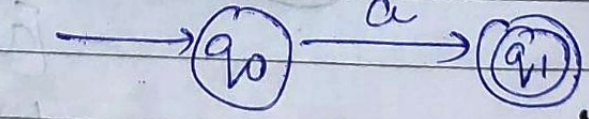
(1) ϕ →



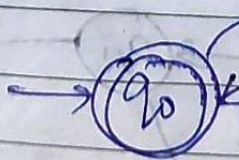
(2) ϵ →



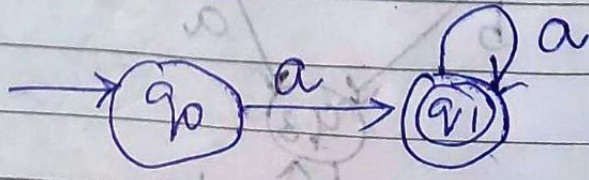
(3) a →



(4) a^* →



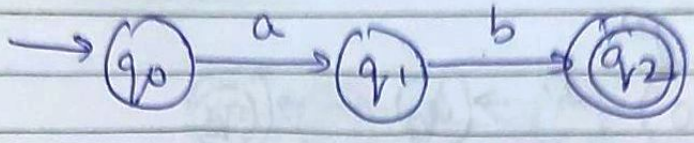
(5) a^+ →



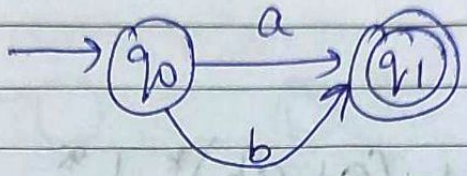
} primitive RE's

DATE
PAGE

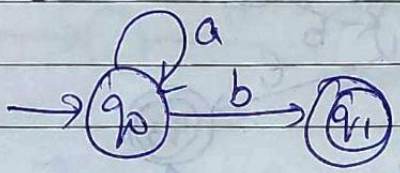
(6) $a \cdot b \rightarrow$



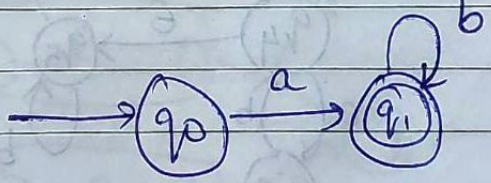
(7) $a+b \rightarrow$



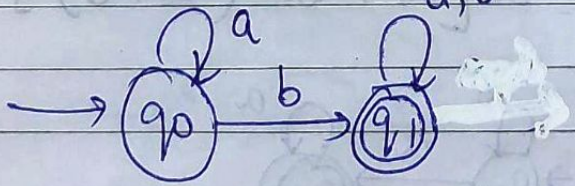
(8) $a^* b \rightarrow$



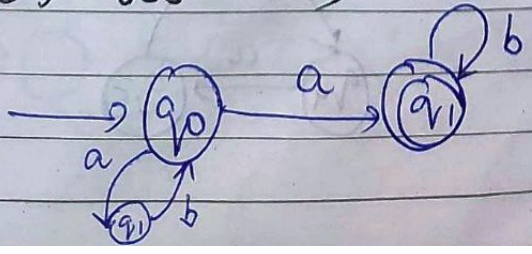
(9) $ab^* \rightarrow$



(10) $a^* b(a+b)^* \rightarrow$

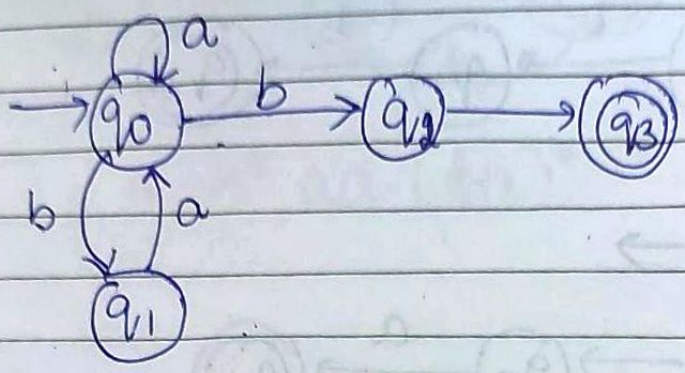


(11) $(ab)^* ab^* \rightarrow$



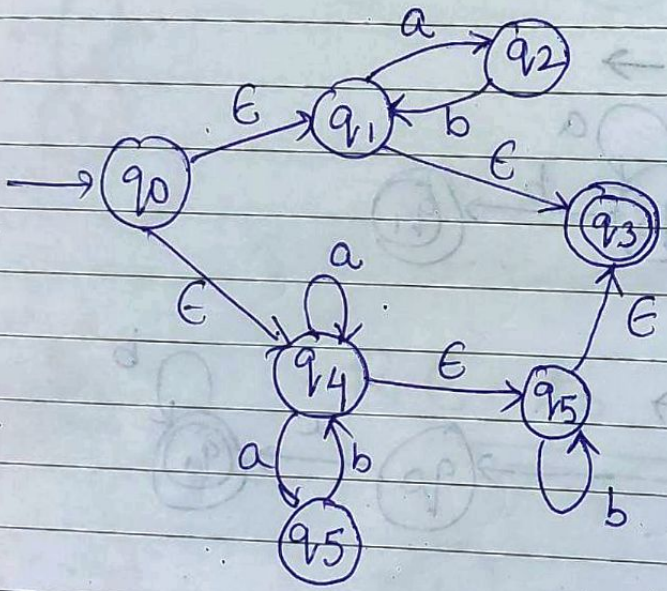
DATE
PAGE

(12) $(a+ba)^* ba$

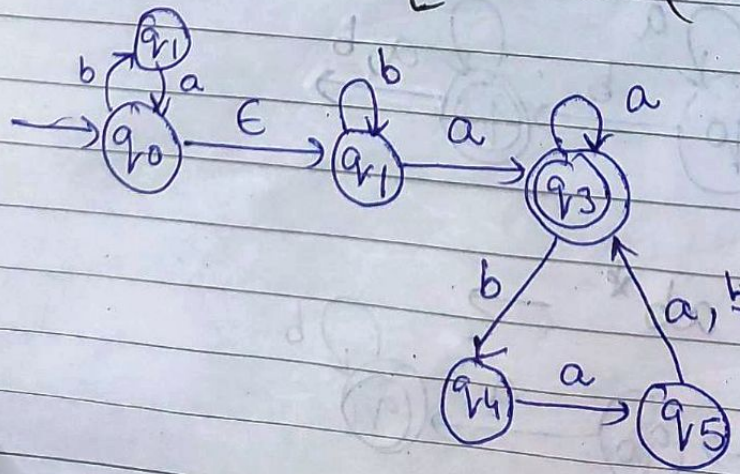


(13) $(ab)^* + (a+ab)^* b^*$

in $A^* . B^*$ use ϵ move



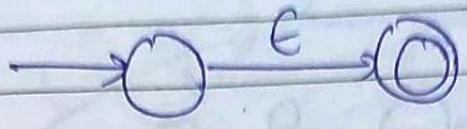
(14) $(ba)^* b^* a[a+ba(a+b)]^*$



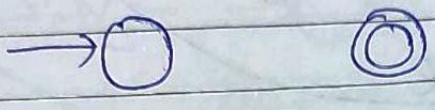
DATE
PAGE

RE to ϵ-NFA :-

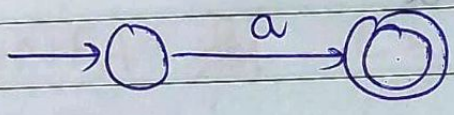
1. ϵ →



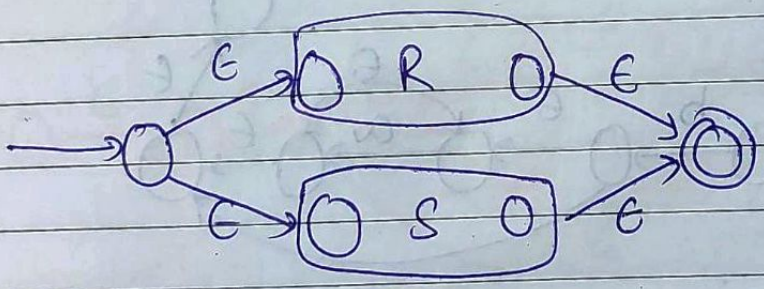
2. ϕ →



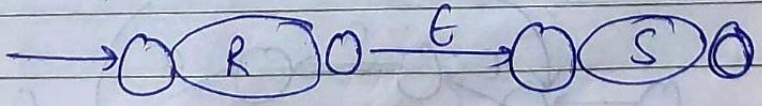
3. a →



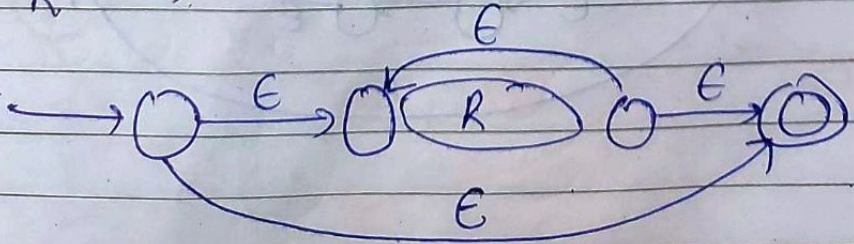
4. R+S →



5. R.S →



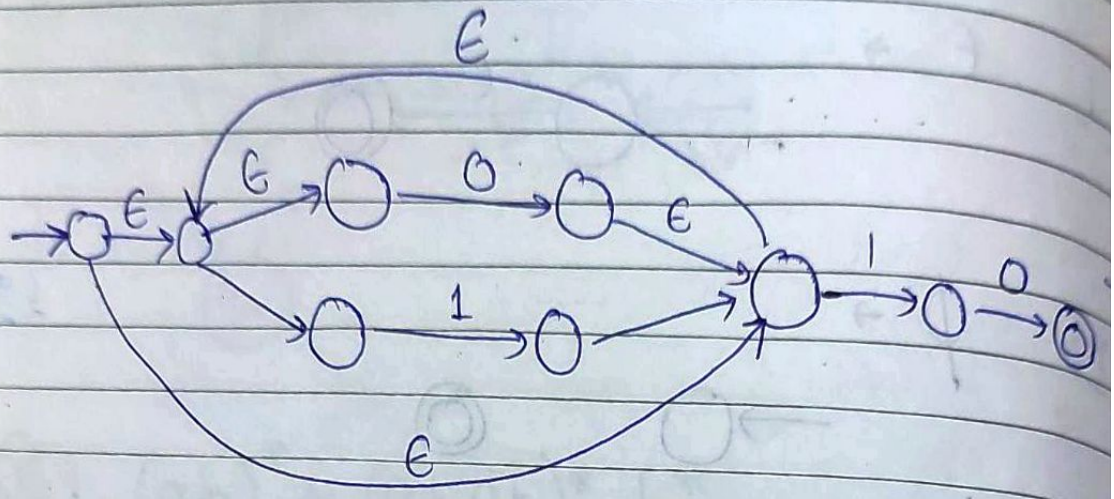
6. R* →



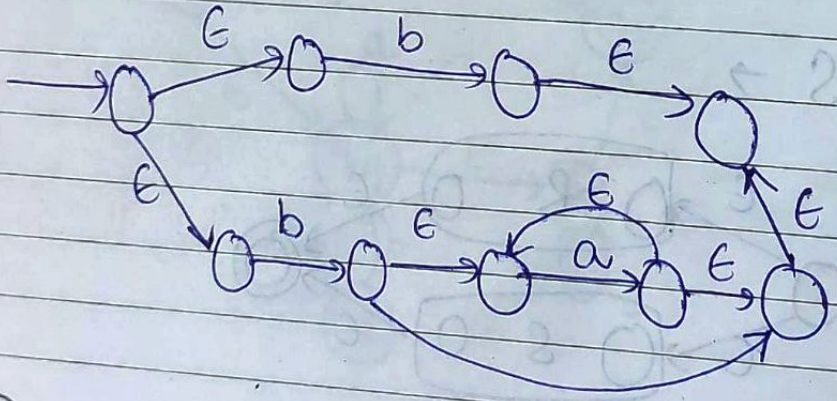
Eg:-

DATE
PAGE

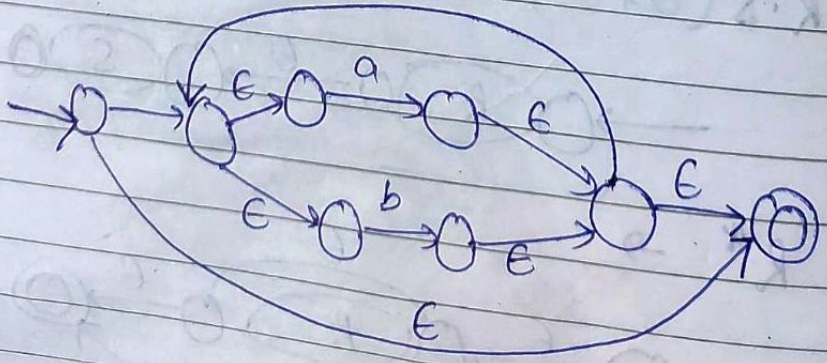
① Convert RG $(0+1)^* 10$ to an ϵ-NFA



② $b + ba^*$



③ $(a+b)^*$

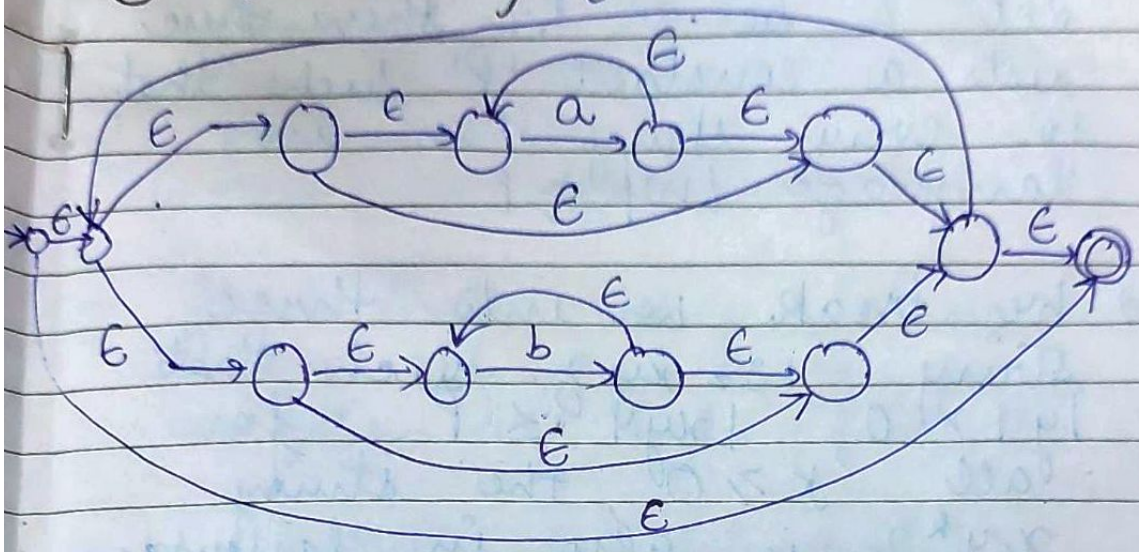


Sowam

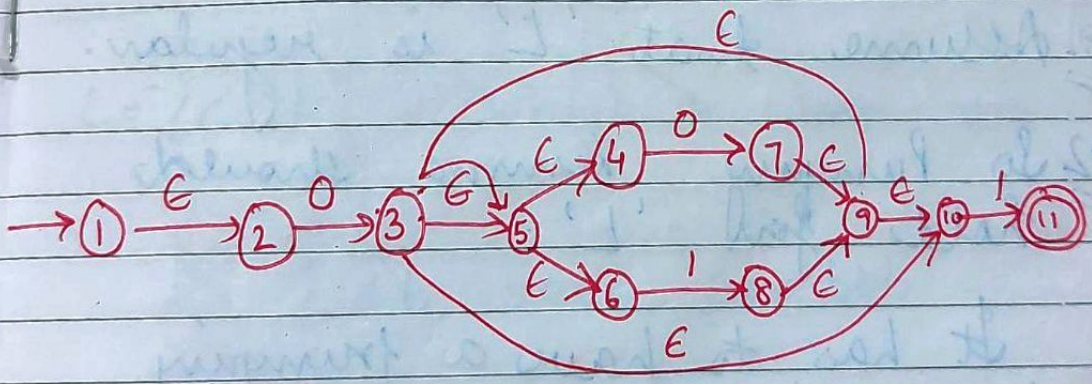
DATE

PAGE

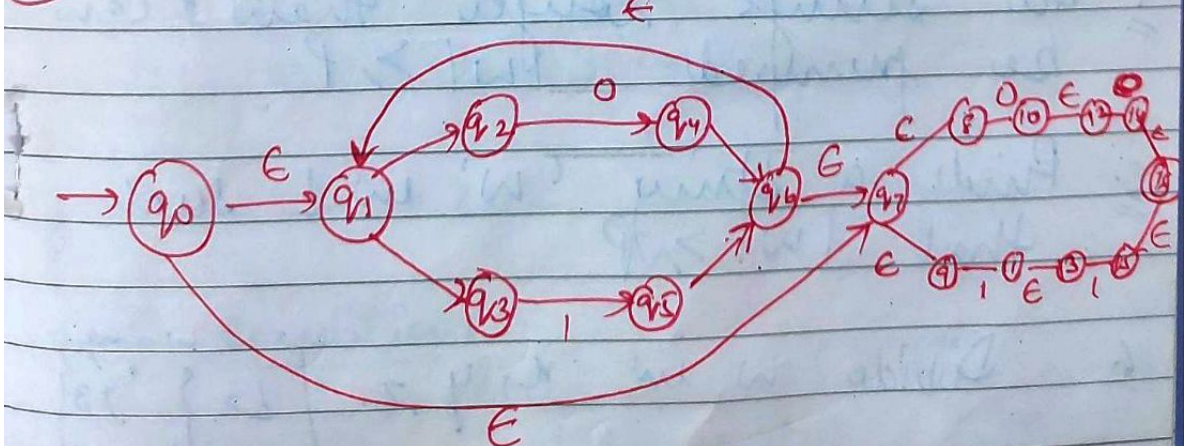
(4) $(a^* + b^*)^*$



(5) ϵ-NFA for $L = 0(0+1)^*1$



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



Pumping Lemma :-

- to prove a lang. as non-regular

Let 'L' be a R.L then there exists a constant 'P' such that for every string 'w' in language $|w| \geq P$.

⇒ we break w into three string $w = xyz$ such that $|y| > 0$, $|xy| \leq P$, for all $k \geq 0$ the string $xy^k z$ is also in language.

Steps :-

1. Assume that 'L' is regular.
2. So Pumping Lemma should hold for 'L'.
3. It has to have a pumping length p
4. All strings longer than P can be pumped $|w| \geq P$
5. Find a string 'w' in L such that $|w| \geq P$
6. Divide w in x, y, z $\quad L = \{ a^n b^n \mid w = aabb, xyz \}$

7. Show that $xy^kz \notin L$ for some k.

8. Then consider all ways that w can be divided into xyz.

9. Show that none of these can satisfy all the 3 pumping condition at same time.

10. w cannot be pumped.

Example :-

$L = \{a^n b^n : n \geq 1\}$ is non-regular.

$\Rightarrow L \in RL$

$L = \{ab, aabb, aaabbb \text{ ---}\}$

P (pumping length) = 3

$a^n b^n \Rightarrow a^p b^p \Rightarrow a^3 b^3$

$\Rightarrow aaabbb = w$

Now $w = xyz \Rightarrow \underbrace{a}_{x}\underbrace{aa}_{y}\underbrace{bbb}_{z}$

Chk conditions :-

↳ (1) Is $|y| > 0$

$|ab| = 2 > 0$ ✓

DATE
PAGE

② Is $|xy| \leq P$

$\Rightarrow |aaab| = 4 \leq 3$ (false)

Out of three conditions if any one is false then the lang. is non regular.

③ Is $xy^k z \quad k \geq 0$

Let $xy^2 z \quad k=2$

$aa(ab)^2 bb$

$\Rightarrow$ aa'ab ab bb] Now this does not belong to L.

So this language is irregular.

$\longleftrightarrow$

Myhill Nerode Theorem :-

↳ Draw a table for all pairs of state (P, Q).

⇒ Mark all pairs where $P \in F$ & $Q \notin F$ & vice versa.

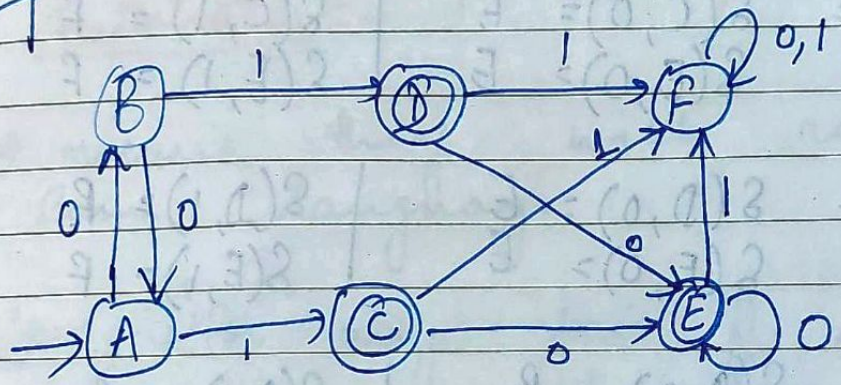
⇒ If there are any ✗ unmarked

pairs (P, Q) such that $[\delta(P,x), \delta(Q,x)]$ is marked then mark (P, Q)

⇒ Repeat till no more markings are left.

⇒ Combine all unmarked pairs & make them a single state.

Eg:-



AB
BD
AC

Draw table of n states.

Here states = 6

So make a 6 × 6 table

	A	B	C	D	E	F
A						
B						
C	✓	✓				
D	✓	✓				
E	✓	✓				
F	✓	✓	✓	✓	✓	

→ Removed.

DATE
PAGE

Unmarked:-

$(A,B) = \delta(A,0) = B \mid \delta(A,1) = C$
$\delta(B,0) = A \mid \delta(B,1) = D$
$(B,A) \qquad (C,D)$

As BA is unmarked so leave it
if CD is " " " "

$(C,D) = \delta(C,0) = E \mid \delta(C,1) = F$
$\delta(D,0) = E \mid \delta(D,1) = F$

$(C,E) = \delta(C,0) = E \mid \delta(C,1) = F$
$\delta(E,0) = E \mid \delta(E,1) = F$

$(D,E) = \delta(D,0) = E \mid \delta(D,1) = F$
$\delta(E,0) = E \mid \delta(E,1) = F$

$(A,F) = \delta(A,0) = B \mid \delta(A,1) = C$
$\delta(F,0) = f \mid \delta(F,1) = f$

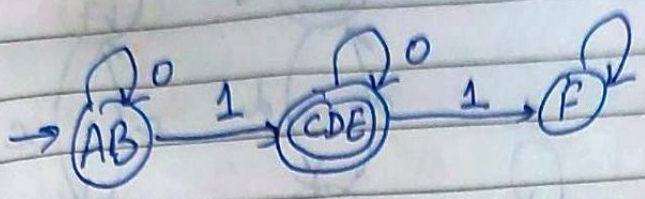
Here as (C,F) is marked so if any of two is marked so we will marked (A,F)

$(B,F) = \delta(B,0) = A \mid \delta(B,1) = D$
$\delta(F,0) = f \mid \delta(F,1) = f$

So here also marks (B,F)

Now make a single state of all unmarked pairs.

(A,B), (C,D), (C,E), (D,E)



⇒ Minimal DFA.

Equivalence of two finite automata :-

⇒ means they will accept same language.

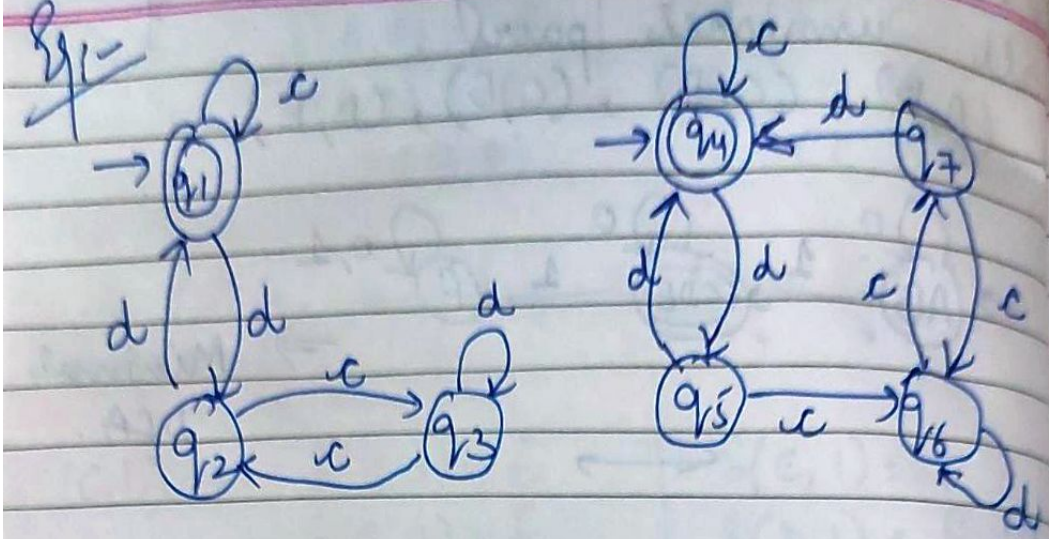
Steps: (1) For any pair of states $\{q_i, q_j\}$, the transitions for $\forall p$ $a \in \Sigma$ is defined by $\{q_a, q_b\}$ where $\delta\{q_i, a\} = q_a$

$\delta\{q_j, a\} = q_b$

⇒ Two automata are not equivalent if for a pair $\{q_a, q_b\}$, one is intermediate & other is final state.

(2) If initial state is final state of one automata, then in 2nd of automata also initial should be final, then they are equivalent.

Eg:-



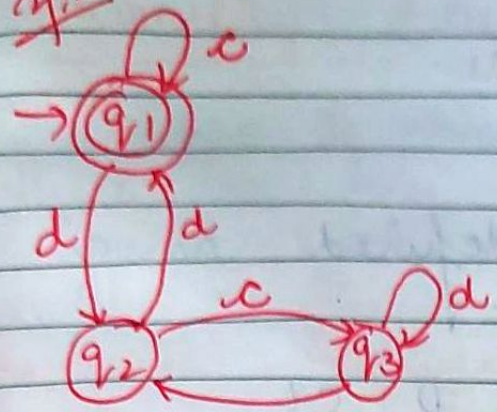
A = B =

I/P (Also both have same final & initial state)

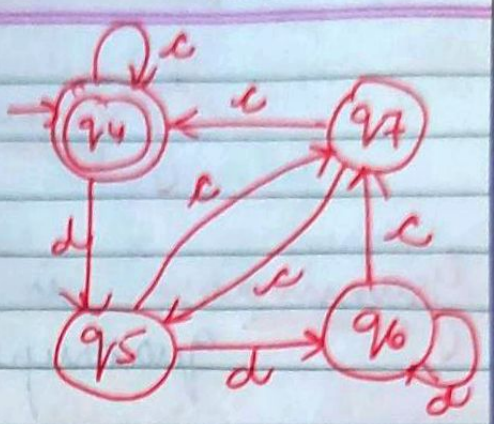
States	c	d
Let 1st pair = q_1, q_4		
(q_1, q_4)	(q_1^F, q_4^F)	(q_2^I, q_5^I) (Eq)
(q_2, q_5)	(q_3^I, q_6^I)	(q_1^F, q_4^F) (Eq)
(q_3, q_6)	(q_2^I, q_7^I)	(q_3^I, q_6^I) (Eq)
(q_2, q_7)	(q_3^I, q_6^I)	(q_1^F, q_4^F) (Eq)

⇒ A & B are equivalent.

Q4:-



A



B

States	c	d
$\{q_1, q_4\}$	$\{q_1, q_4\}$ (F, F)	$\{q_2, q_5\}$ (I, I) (eq)
$\{q_2, q_5\}$	$\{q_3, q_7\}$ (I, I)	$\{q_1, q_6\}$ (F, I)
		As q_1 & q_6 are different set of states so

A & B are not equivalent.

Unit 3

Saran

DATE

PAGE

Grammar

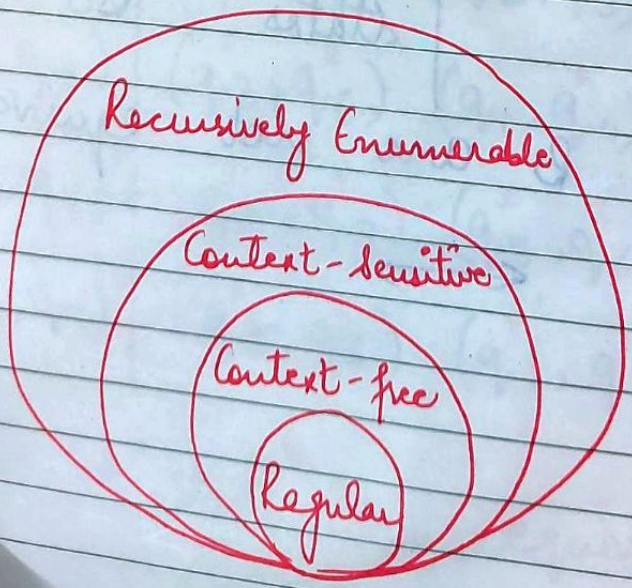
Grammar:- is defined as a quadruple.

$$G = \{V, T, P, S\}$$

V → Variable
T → Terminal
P → Production rule
S → Start symbol.

Eg:- aSb / ϵ

⇒ a, $aaSbb$, $aaaSbbb$
or ab, a^2b^2, a^3b^3,
a^n, b^n.



Chomsky Hierarchy

DATE
PAGE

Chomsky classification of Grammar

Acc. to Noam Chomsky, there are four types of grammars - Type 0, Type 1, Type 2 & Type 3.

Grammar Type	Grammar Accepted	Language Accepted	Automation
Type 0	Unrestricted grammar	Recursively enumerable language	Turing machine
Type 1	Context - sensitive grammar	Context sensitive lang.	Linear bounded automation
Type 2	Context free grammar	Context free language	Pushdown automation
Type 3	Regular grammar	Regular language	Finite state automation

DATE
PAGE

Type-3

↳ generate regular languages
↳ must have single non-terminal on L.H.S & R.H.S containing a single terminal or single terminal followed by a single non-terminal.

Eg:- $X \to a$ or $X \to aY$

where $X, Y \in N$ (Non-terminal)
$a \in T$ (Terminal)

$S \to \epsilon$ is allowed if S does not appear on Right side of any rule.

Eg:
$X \to \epsilon$
$X \to a \mid aY$
$Y \to b$

Type-2

↳ generate context free grammar.
↳ Production is of form $A \to \gamma$

$A \in N$
$\gamma \in (T \cup N)^*$ (string of terminals & non-terminals)

Sonam

DATE
PAGE

⇒ recognized by non-deterministic pushdown automation.

Eg:-
$S \to Xa$
$X \to a$
$X \to aX$
$X \to abc$
$X \to \epsilon$

Type-1

↳ generate context-sensitive languages.

↳ Production is of form:-

$$\alpha A \beta \to \alpha \gamma \beta$$

$A \in N$

$\alpha, \beta, \gamma \in (T \cup N)^*$ (String of terminals & non-terminals).

α, β may be empty but γ must be non-empty.

$S \to \epsilon$ is allowed if S does not appear on right side of any rule.

⇒ recognized by linear bounded automation.

DATE
PAGE

Eg:- $AB \rightarrow AbBc$
$A \rightarrow bcA$
$B \rightarrow b$

Type-0

↳ generate recursively enumerable languages.
↳ recognised by Turing machine.
↳ Production is of form :-
$\alpha \rightarrow \beta$ where
$\alpha \rightarrow$ string of terminals & non-terminals with atleast one non-terminal & α cannot be null.
$\beta \rightarrow$ string of terminals & non-terminals.

Eg:- $S \rightarrow ACaB$
$Bc \rightarrow acB$
$CB \rightarrow DB$
$aD \rightarrow Db$

Regular Language to Regular Expression

I/P = {a, b}

1. Start with 'a' → $a(a+b)^*$

2. Start with 'aba' → $aba(a+b)^*$

3. End with 'ab' → $(a+b)^* ab$

4. Substring as 'aba' → $(a+b)^* aba (a+b)^*$

5. $|w| = 3$ → $(a+b)(a+b)(a+b)$

6. $|w| \geq 3$ → $(a+b)(a+b)(a+b)(a+b)^*$

7. $|w| \leq 3$ → $(a+b+\epsilon)(a+b+\epsilon)(a+b+\epsilon)$

8. $|w|_a = 2$ → (no. of a's = 2) $b^*ab^*ab^*$

9. $|w|_a \geq 2$ → $(a+b)^* a (a+b)^* a (a+b)^*$

10. $|w|_a \leq 2 \rightarrow b^*(a+\epsilon)b^*(a+\epsilon)b^*$

11. $|w| = \underset{\text{remainder}}{0} \pmod 3 \rightarrow$ 0, 3, 6, 9, 12 - - -
(rem should be 0 when divided by 3)

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

12. $|w| = 2 \pmod 3 \rightarrow$ 2, 5, 8, 11 - - -

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

Regular Expressions to Regular Languages :-

1. $RE = a \rightarrow L = \{a\}$
2. $a \cdot b \rightarrow L = \{ab\}$
3. $a+b \rightarrow L = \{a, b\}$
4. $(a+b) \cdot b \rightarrow L = \{ab, bb\}$
5. $a^* \rightarrow L = \{\epsilon, a, aa - - -\}$
6. $(a+b)^* \rightarrow L = \{\epsilon, a, b, aa, ab, ba, bb, aaa, aba, - - - - - - -\}$

$\{a, a, \epsilon \cup$ DATE PAGE $\}$

7. $a^* \cup a^+ \rightarrow \{\epsilon, a, aa, aaa \cdots\}$

$\Rightarrow L = \{\epsilon, a, aa, aaa \cdots\}$

or

$L = a^*$

8. $a^* \cap a^+ \rightarrow L = \{a, aa, aaa \cdots\}$

or

$L = a^+$

9. $(a^*)^* \rightarrow \{\epsilon, a, aa, aaa \cdots\}$

$= a^*$

10. $(a^+)^* \rightarrow a^*$

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

11. $(a^*)^+ \rightarrow a^*$

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

$\longleftrightarrow$

Linear Grammar :-

Any CFG that has atmost one non terminal in right hand side of each production is called a linear grammar.

eg:- $S \rightarrow aSb \mid bSa \mid ab \mid ba$

Right linear grammar

⇒ in which the only non-terminal is on the rightmost end of production

eg:-

$S \rightarrow aA \mid bB \mid \epsilon$
$A \rightarrow aS \mid a$
$B \rightarrow bS \mid b$

Left linear grammar

⇒ in which the only non-terminal is on leftmost end of production

$S \rightarrow Aa \mid Bb \mid \epsilon$
$A \rightarrow Sa \mid a$
$B \rightarrow Sb \mid b$

Right recursive grammar

⇒ in which the non-terminal is same as that of left hand side of production.

:- $S \rightarrow aS \mid bS \mid a \mid b$

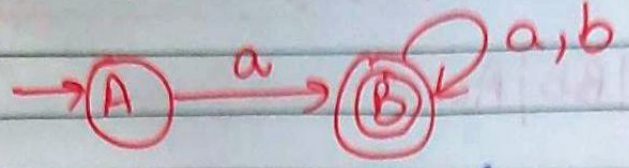
Left recursive grammar

⇒ in which the NT is same as that on the L.H.S of production

$S \rightarrow Sa \mid Sb \mid a \mid b$

DATE
PAGE

Q:= FA to RL.

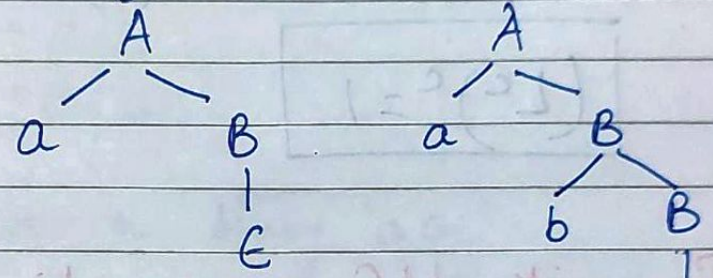


$A \rightarrow aB$ (A is taking a to B)

$B \rightarrow aB \mid bB \mid \epsilon$ (ϵ is written bcoz B is final state)

↓

Right linear grammar



= a

= ab

So these strings can be derived from this.

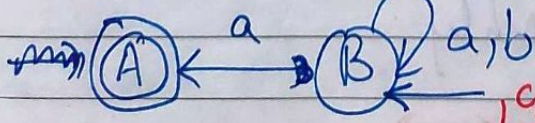
⇒ To convert RLG to LLG, we reverse the FA given.

FA → L

⇓

FA → L^C (final to non-final & vica-versa)

(arrows will be reversed)



$L^C = \begin{bmatrix} B \rightarrow aB \mid bB \mid aA \\ A \rightarrow \epsilon \end{bmatrix}$

now reverse this, $(L^c)^c = L$

$$L = \begin{bmatrix} B \to Ba \mid Bb \mid Aa \\ A \to \in \end{bmatrix}$$

Hence,

$$FA(L) \overset{R}{\Rightarrow} FA(L^c) \Rightarrow \underset{(L^c)}{RLG} \overset{R}{\Rightarrow} LLG$$

$$\boxed{(L^c)^c = L}$$

⇒ To write LLG corresponding to a language :—

L
⇓
$FA(L)$
⇓
$FA(L^c)$
⇓
$RLG(L^c)$
⇓
$LLG(L^c)^c = L$

⇒ To write RLG corr. to lang. :—
it is bydefault written in RLG only.

Saram

DATE
PAGE

Regular Language to Regular Grammar

1. $L = \{aa, ab, ba, bb\}$ (finite lang.)

$S \to aa \mid ab \mid ba \mid bb$

Or $R.E = \underbrace{(a+b)}_{A}\underbrace{(a+b)}_{B}$ for L

$S \to AB$
$A \to a \mid b$
$B \to a \mid b$

lets derive a string 'aa'

LMD → $S \to AB$
$S \to aB$ [$\because A \to a$]
$S \to aa$ [$\because B \to a$]

Or $S \to AA$
$A \to a \mid b$

2. $L = a^n \mid n \geq 0$ (infinite lang.)

$L = \{\epsilon, a, aa, aaa \cdots\}$

$S \to aS \mid \epsilon$

'to derive a' → $S \to aS$
$\to a\epsilon \Rightarrow a$

3. $L = (a+b)^*$

$S \to aS \mid bS \mid \epsilon$

4. $L = a^m b^n \mid m, n \geq 0$

$= a^* b^*$

$S \to AB$

$A \to aA \mid \epsilon \Rightarrow a^*$

$B \to bB \mid \epsilon \Rightarrow b^*$

5. $L = a^m b^n c^k \mid m, n, k \geq 0$

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

$S \to ABC$

$A \to aA \mid \epsilon \Rightarrow a^*$

$B \to bB \mid \epsilon \Rightarrow b^*$

$C \to cC \mid \epsilon \Rightarrow c^*$

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

$S \to AAB$

$A \to a \mid b$

$B \to aB \mid bB \mid \epsilon \Rightarrow (a+b)^*$

7. $L = a^n b^n \mid n \geq 0$

$= \{\epsilon, ab, aabb, a^3b^3 \ldots\}$

$S \to aSb \mid \epsilon$