Module - IV

Regular expression

→ Regular expression (RE) is defined as

→ ϵ is RE corresponding to the $L = \{\epsilon\}$

→ ϕ is RE " " " $L = \{\}$

→ x " " " " $L = \{x\}$

Regular language

→ * If x is RE over language L(x)

Y " " " L(Y)

Union

→ x+y is RE corresponding to $L(x) \cup L(y)$

⇒ where $L(X \cup Y) = L(X) \cup L(Y)$

→ $x.y$ is RE corresponding to

$L(x) . L(y)$

where $L(x.y) = L(x) . L(y)$

closure

→ R^* is RE corresponding to

$L(R^*) = (L(R))^*$

Applications

→ Data validation

eg: compiler

Regular Language

Is Language that can be defined by ~~Expressions~~.

→ DFA

→ NFA with ϵ

→ NFA

→ Regular expression

(1) ⇒ RE denoting language with strings have any number of a's over.

$\Sigma = \{a\}$

$RE = a^*$

(2) any no's of 'a' any no's b's

$RE = (a+b)^*$

(3) RE denoting / with strings starts with 'a' and end with 'b'

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

If P and Q are two RE over Σ, and if P does not contain ϵ, then the following equation in R given by $R = Q + RP$ has a unique solution

i.e $R = QP^*$

Soln

$$R = Q + RP \rightarrow (1)$$

Step 1

$$= Q + QP^*P \qquad \boxed{R = QP^*}$$

$$= Q(\epsilon + P^*P) \qquad (\epsilon + R^*R = R^*)$$

$$= QP^*$$

Hence proved

$$R = Q + RP$$

$$= Q + [Q + RP]P$$

$$= Q + QP + RP^2$$

$$= Q + QP + (Q + RP)P^2$$

$= Q + QP + QP^2 + RP^3$

$\vdots$

$= Q + QP + QP^2 + \cdots QP^n + RP^{n+1}$

$[R = QP^*]$

$= Q + QP + QP^2 + \cdots + QP^n + QP^* P^{n+1}$

$= Q[\epsilon + P + P^2 + \cdots + P^n + P^* P^{n+1}]$

$R = QP^*$ (closure of P)

Identity Rules for RE

* Simplify the RE

* ϕ → null set

  ϵ → null string

If P, Q, r are the RE then

Rule 1 :- $\phi + r = r$

2 :- $\phi \cdot r = r \cdot \phi = \phi$

3 :- $\epsilon \cdot r = r \cdot \epsilon = r$

4 :- $r + r = r$

5 :- $r^* \cdot r^* = r^*$

6 :- $r^* r = r r^* = r^*$

7 :- $(r^*)^* = r^*$

8 :- $\epsilon + r^* r = \epsilon + r r^* = r^*$

9 :- $(PQ)^* P = P(QP)^*$

Conversion of Finite automata to RE

I/P = FA

O/P = RE

→ Arden's method

$R = Q + RP$

↳

$R = QP^*$

→ State elimination method

eg: Conversion of RE from FA

Arden's method

→ equation for each state based on incoming edges.
→ Add ϵ to the equation of initial state
→ simplify the equation using Arden's

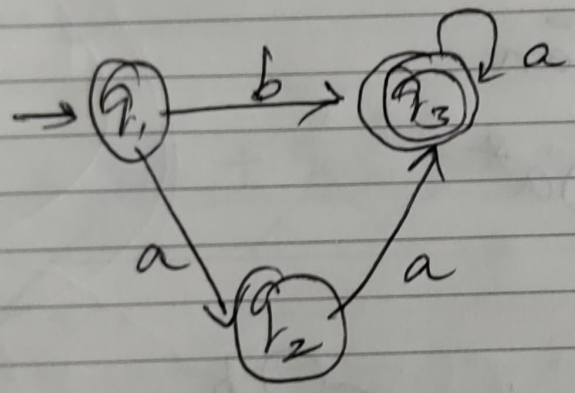
and find RE

conditions

→ FA should not contain ∈ transition

→ FA should have only on Initial state.

eg: 1



$q_1 = \epsilon$ — (1)

$q_2 = q_1 a$ — (2)

$q_3 = q_1 b + q_2 a + q_3 a$ — (3)

Apply (1) & (2) in (3)

$q_3 = \epsilon.b + q_1aa + q_3a$

$q_3 = b + \epsilon.aa + q_3a$

$q_3 = (b + aa) + q_3a$

$R = Q + RP$

$R = QP^*$

unique solution

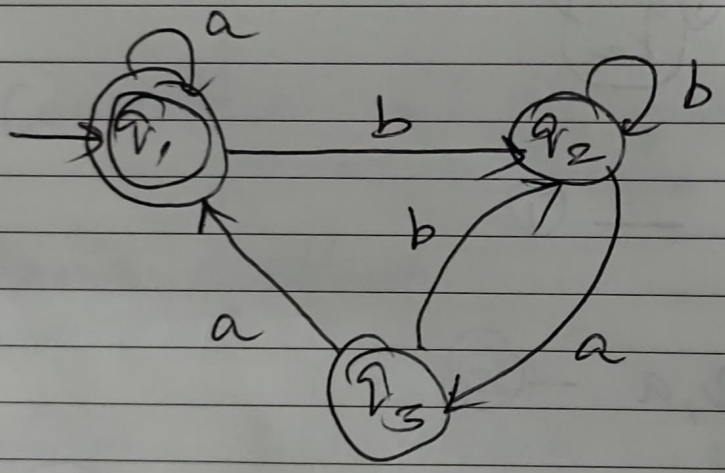
$R = (b + aa)a^*$

(Identity Rule)

$\epsilon.b = b$

$\epsilon.aa = aa$

eg2:-



$$q_1 = q_1 a + q_3 a + \epsilon \quad -(1)$$

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

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

(3) en (1)

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

$q_2 =$

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

$$= q_1 a + (q_2 b + q_1 b + q_3 b) a + \epsilon$$

$$= q_1 a + q_2 ab + q_1 ab + q_3 ab + \epsilon$$

$$q_1 = q_1 a + q_2 ab (q_1 + q_2 + q_3)$$

$$R = Q + RP$$

(3) in (2)

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

$$q_2 = q_1 b + q_2 (b + ab)$$

$$R = \alpha + R\beta \rightarrow R = \alpha\beta^*$$

$$\boxed{q_2 = q_1 b (b+ab)^*} \quad -(4)$$

Apply (3) in (1)

$$q_1 = q_1 a + q_2 aa + \epsilon \rightarrow (5)$$

Apply (4) in (5)

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

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

$$R = \alpha + R\beta \rightarrow R = \alpha\beta^*$$

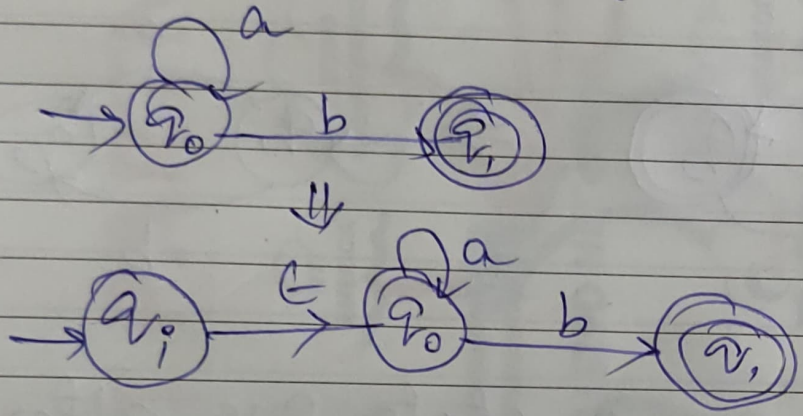
$$q_1 = \epsilon . (a + b(b+ab)^* aa)^*$$

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

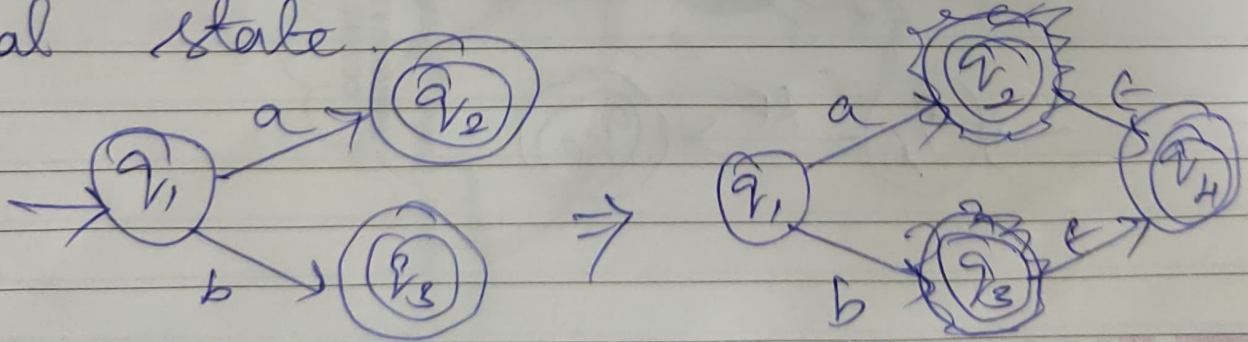
Conversion of FA to RE

(State eliminating method)

(1) → Initial state should not have incoming edge. If exist create new state and make it as intial state.

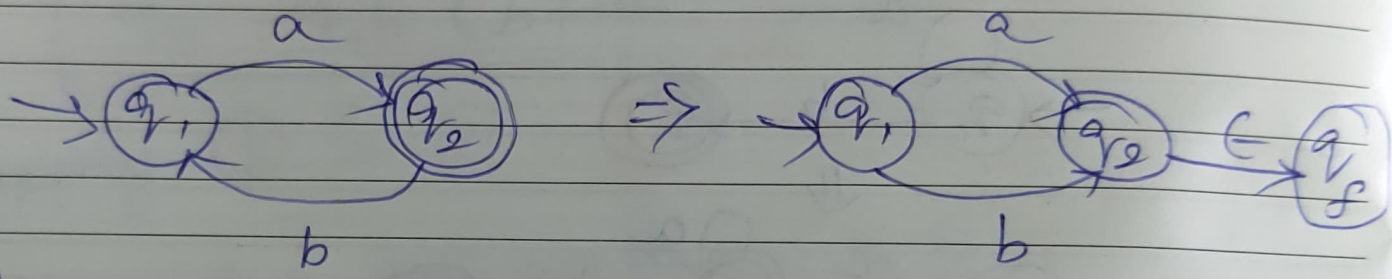


(2) → FA should have single final state If there exist multiple final states, Create new state and make it as final state.



(3)

→ Final state should not have outgoing edge. If exist create new state and make it as a final state.



(4)

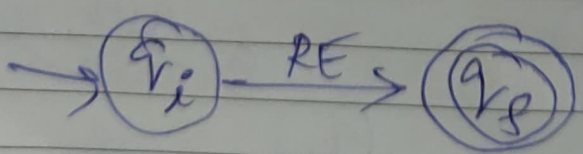
→ Eliminate every state one after another

(5) (Don't have any order)

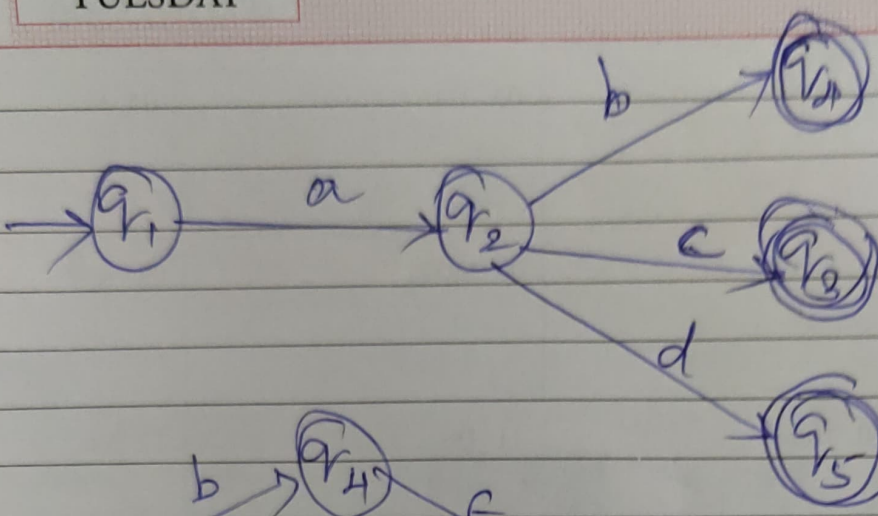
→ FA should have only ~~one~~ two states

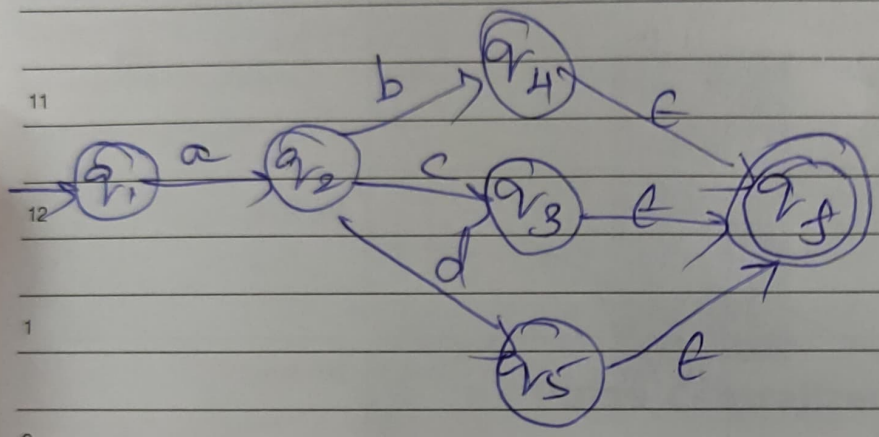
(i) initial state

(ii) final state

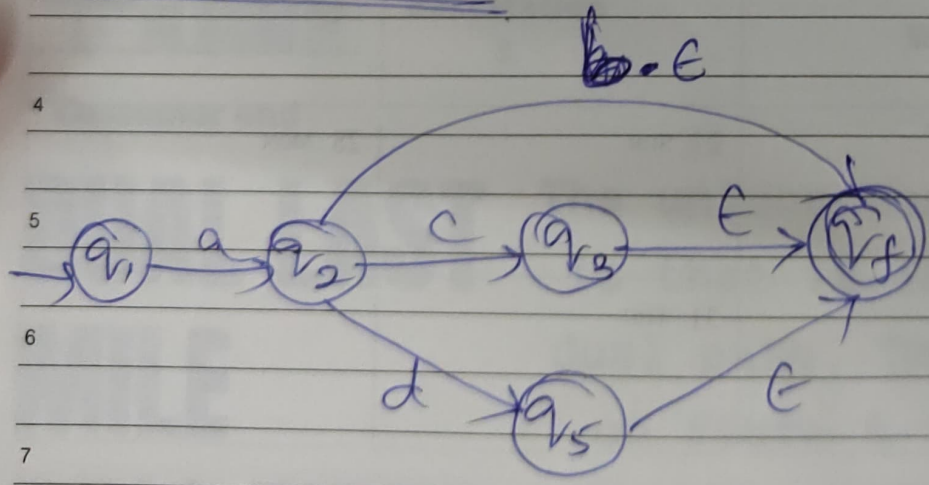


eg : 1

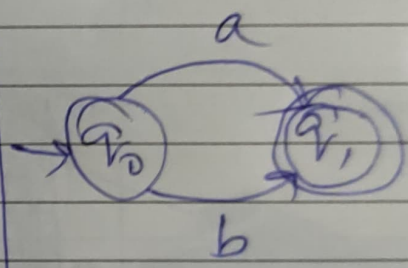




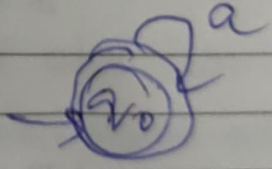
Eliminating q_4



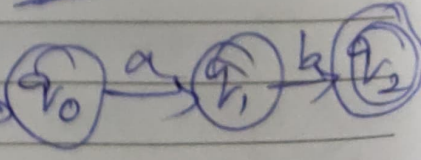
a+b



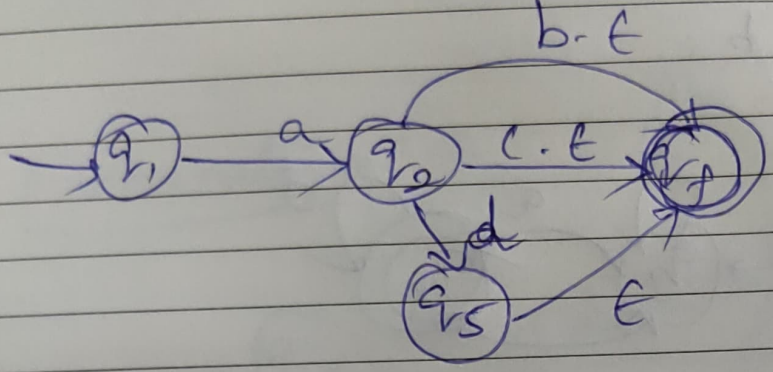
a^*



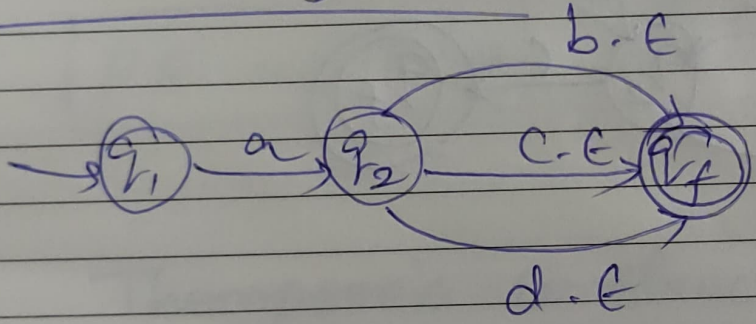
ab



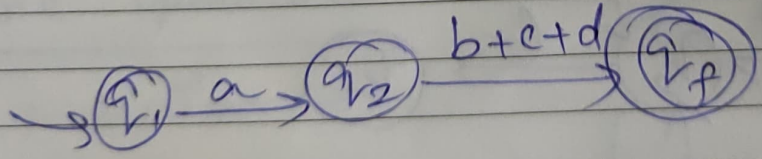
Eliminating q_3



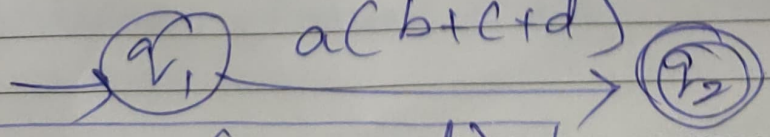
Eliminating q_5



~~Eliminating~~

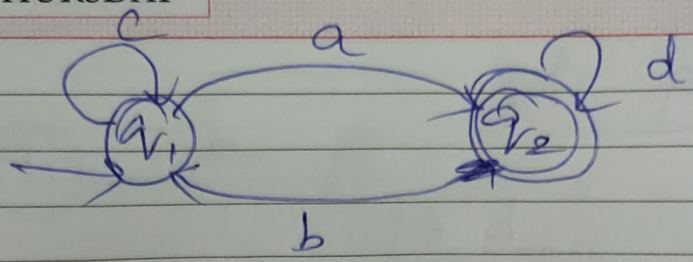


Eliminating q_2

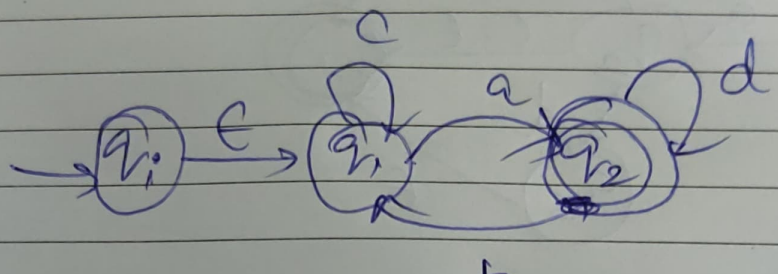


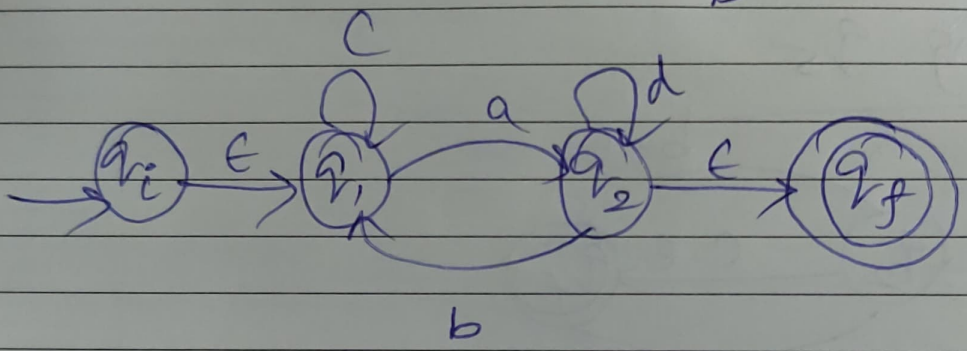
$$RE = a(b+c+d)$$

eg 2:-

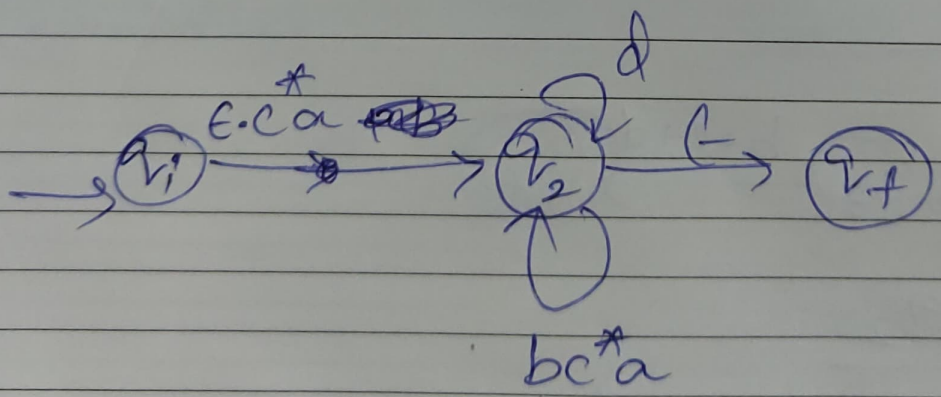


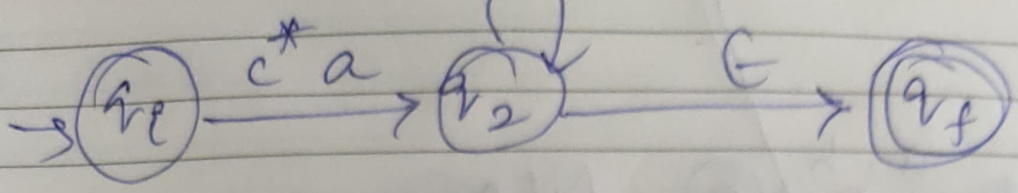
Soln



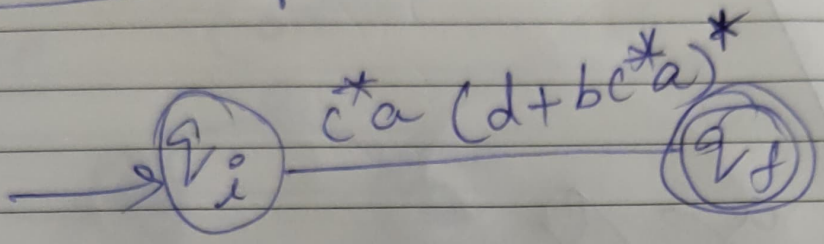


Eliminate q_1





Eliminating q_2



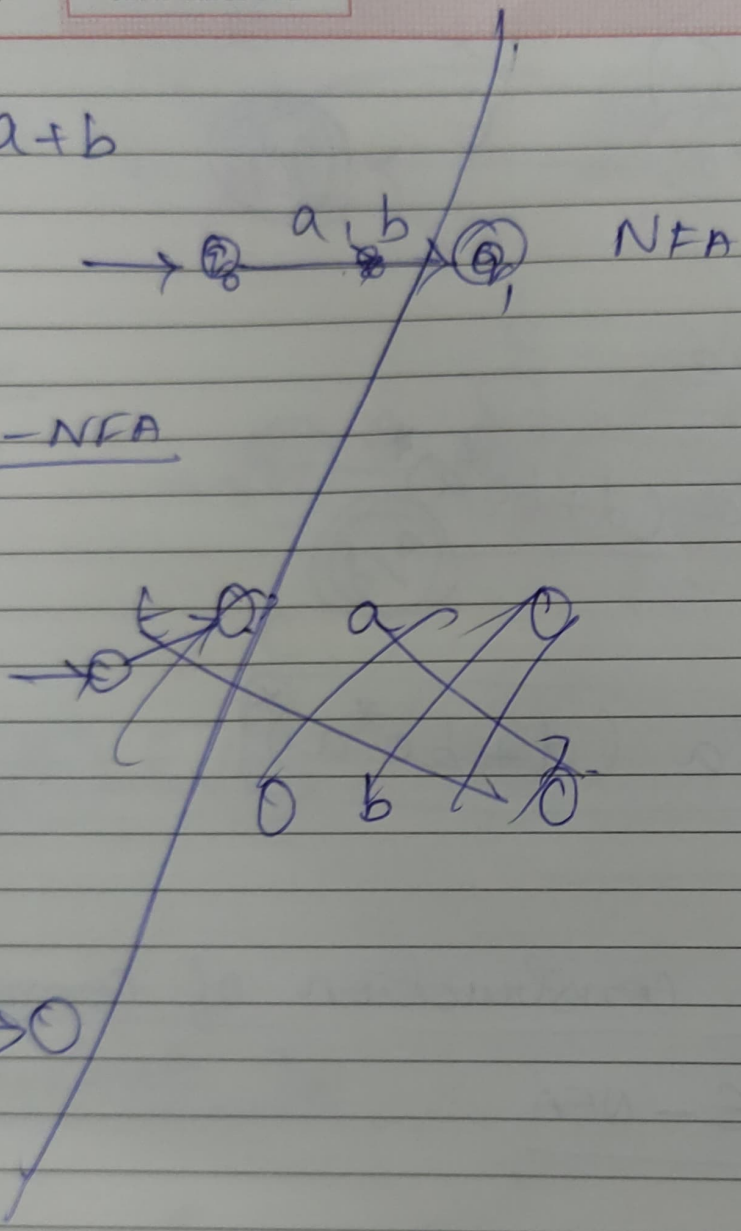
$$RE = c^*a\,(d+bc^*a)^*$$

Thompson's construction of Convert RE to ϵ-NFA

Basic operators

1. Union (+)
2. Concatenation (.)
3. Kleene closure (*)

8 1) a+b



9

10

11 ∈-NFA

12

1

2

3

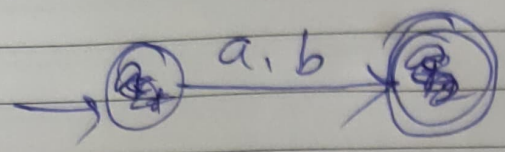
4

5

6 It is also known as MC-Naughton

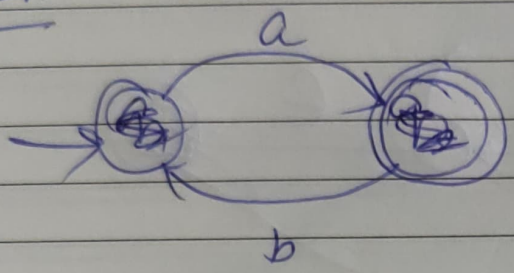
7 Yamada Thompson Algorithms, is a method of transferring a regular expression into an equivalent NFA / ∈-NFA and then one can convert to D

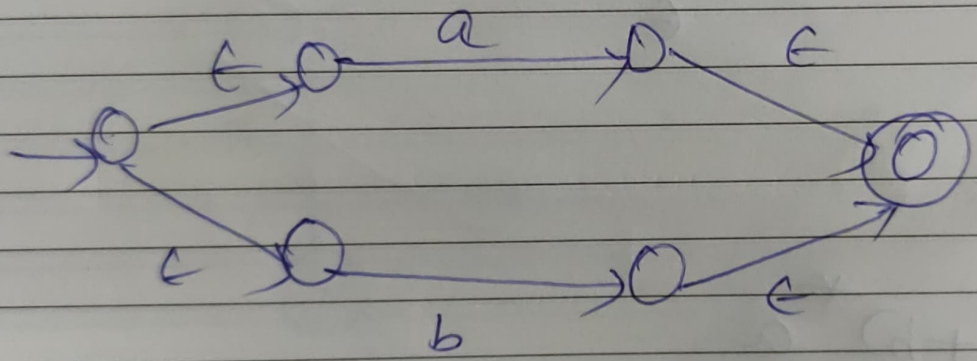
1) a+b



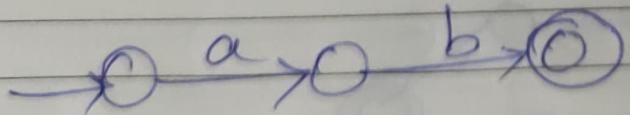
NFA

ϵ- NFA (ON)



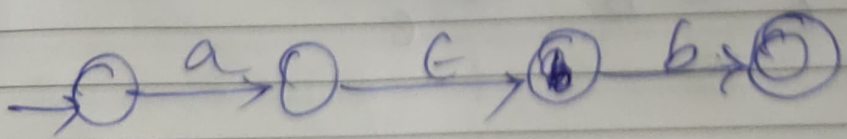


2) ~~(a+b)~~ ab

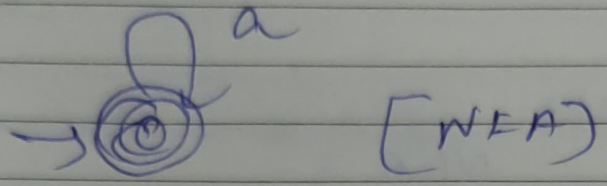


[NFA]

ϵ-NFA

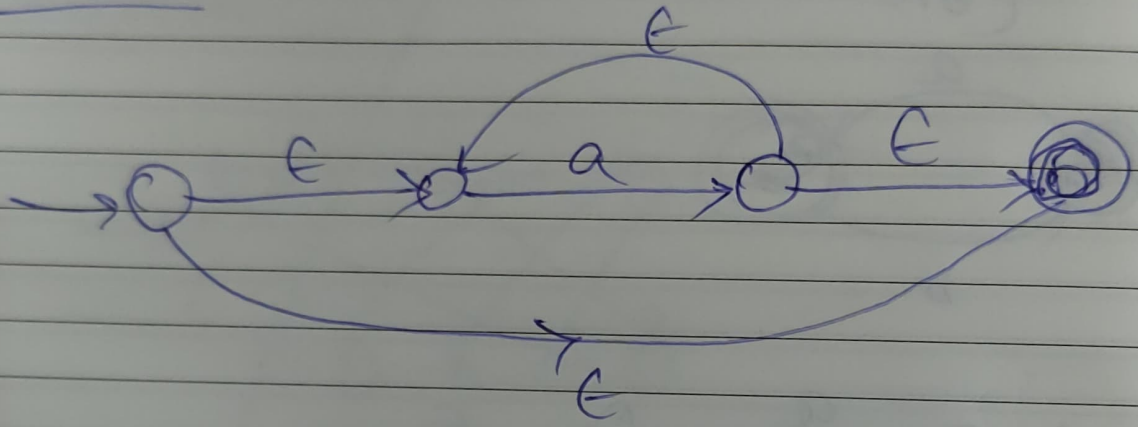


(3) a^*



[NFA]

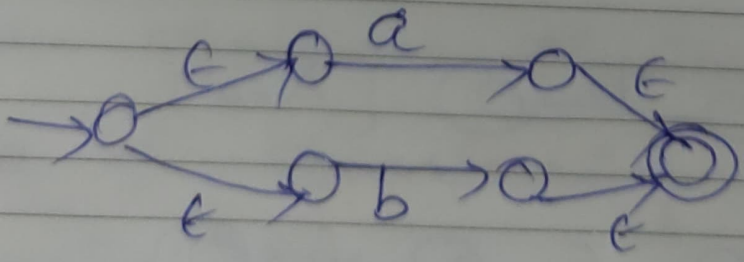
∈-NFA



Ex 1

$(a+b)^* a$

$(a+b)$



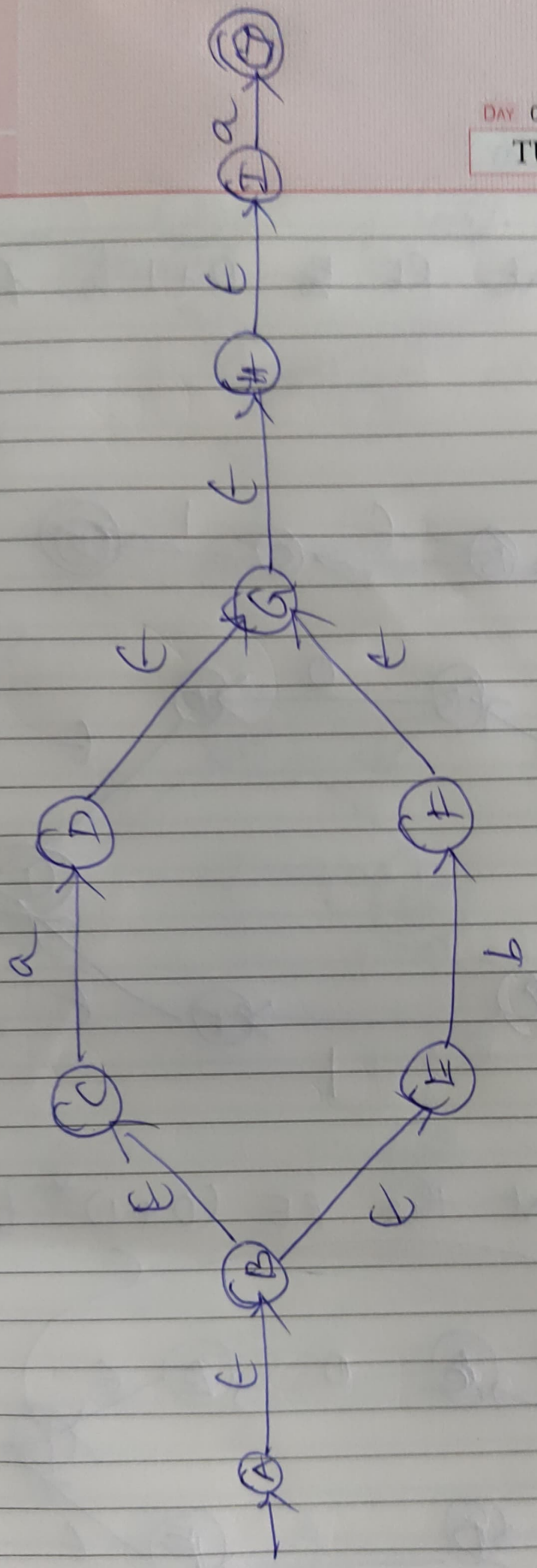
2022

MARCH

DAY 067-298 WEEK 11

TUESDAY

08

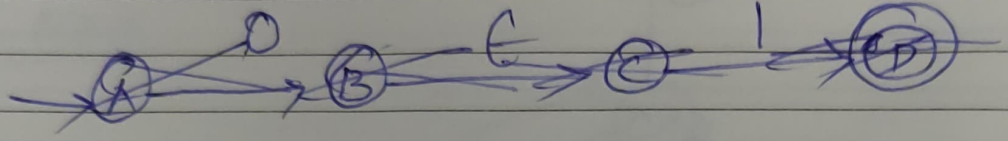


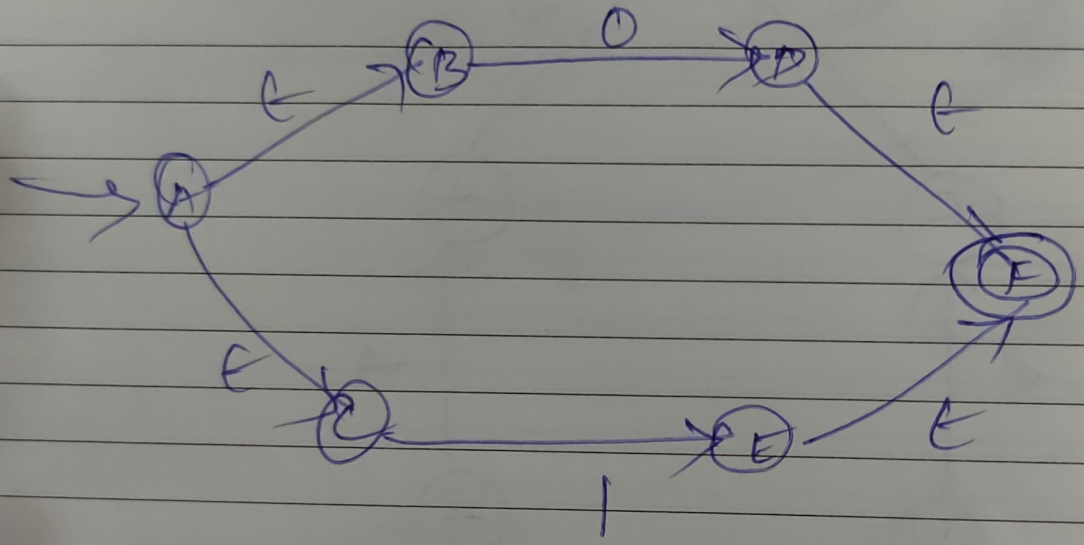
8
9
10
11
12
1
2
3
4
5
6
7

APR
MAY
JUN

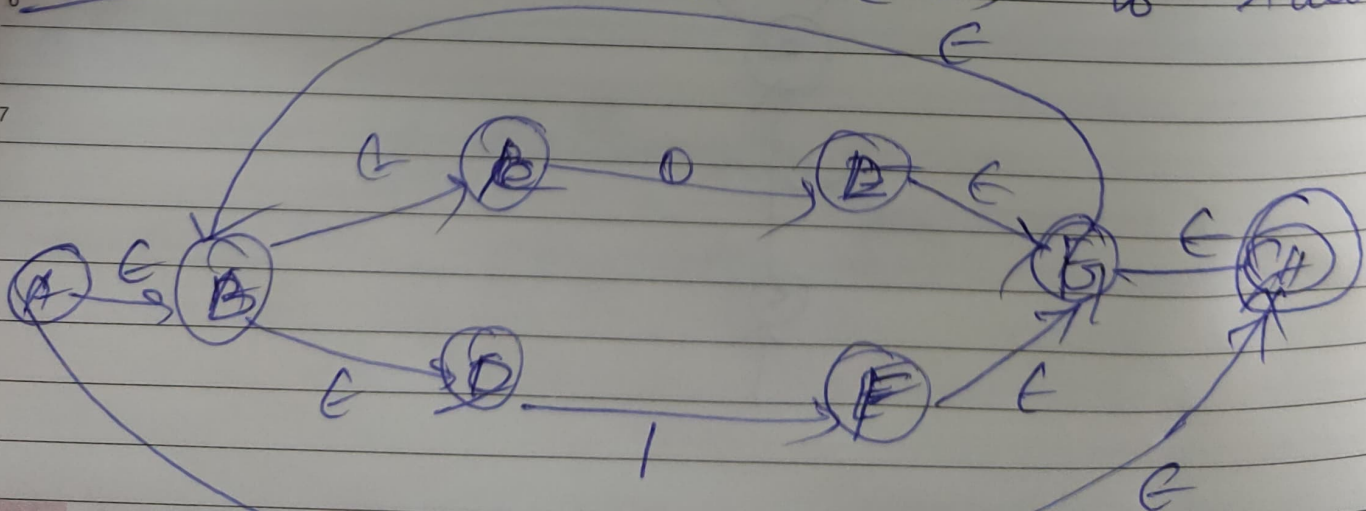
M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	APR
•	•	•	•	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	2022
25	26	27	28	29	30	•																						

eg 2: Convert RE 0+1 to FA

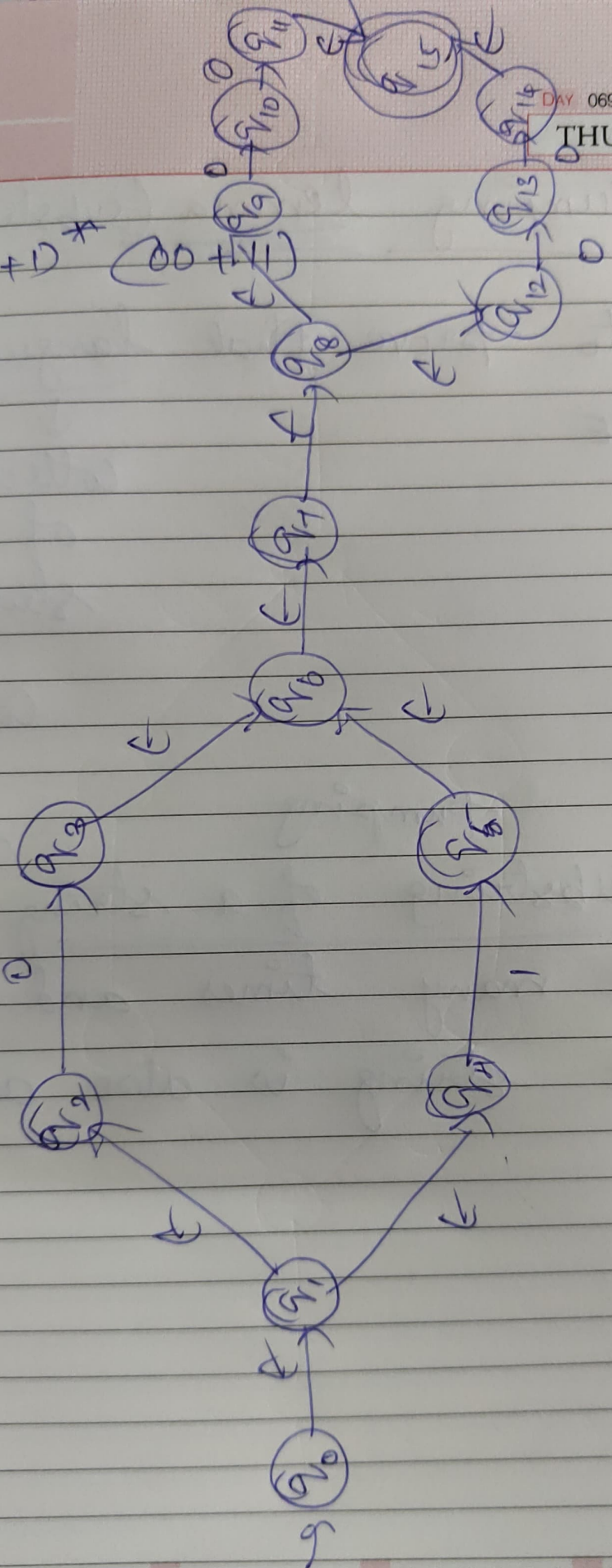




eg 3: Convert the RE $(0+1)^*$ to Automata



3) $(0+1)^* (00+11)$



Pumping lemma (substring)

→ used to prove that language is not RE

language
↓
collection of strings
↓
collection of symbols.

~~pumping~~

→ if substring of a string is repeated many times and if the resultant string is also available