Theory of Computation:

→ Peter Linz
→ John C. Martin

Q. All the strings of ab begin and end with same symbol?

$$\left((a(a+b)^* a + b(a+b)^* b) + \begin{matrix} a \\ +b \end{matrix}\right)$$ for length 2 and above

→ Symbol have atleast one length, including null means zero symbol.

Properties of Regular Expression:

ϕ : Empty set
ϵ : Empty string
$s = \{\epsilon\}$
$|s| = 1$ (cardianality)
$|\phi| = 0$

1. $R.\phi = \phi.R = \phi$
2. $R.\epsilon = \epsilon.R = R$
3. $R.R^* = R^*.R = R^+$
4. $(R^*)^* = R^*$
5. $\epsilon + RR^* = RR^* + \epsilon = R^*$
6. $(P+Q)^* = (P^* + Q^*)^* = (P^* + Q)^* = (P + Q^*)^* = (P^*.Q^*)^* \neq (P^* Q)^* \neq (PQ^*)^* \neq (P^* Q^*)$
7. $(P+Q).R = PR + QR$
8. $P.(Q+R) = PQ + PR$
9. $P^*(Q.P^*)^* = (P^*Q)^* P^* = (P+Q)^*$
10. $\epsilon^* = \epsilon$, $\phi^* = \{\epsilon, \phi, \phi\phi, \ldots\}$

$\{\epsilon, \phi\} = \{\epsilon\}$

$\epsilon^* = \phi^*$

$\{\epsilon, \phi\} = \{\epsilon\}$

ϕ: universal subset of every set

$$\boxed{\epsilon^* = \phi^*}$$

Scanned with CamScanner

* Finite Automaton : [FA] :

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

- Q → set of finite states
- q_0 → an initial state, $q_0 \in Q$
- Σ → set of finite input symbols
- F → set of finite final states, $F \subseteq Q$

δ : Transition function (TF)

DFA
Deterministic finite automaton

$\delta : Q \times \Sigma \rightarrow Q$

or

$\delta(q_i, a) \rightarrow q_j$

where
$q_i, q_j \in Q$
$a \in \Sigma$

→ TF of DFA is always total function

Total function : From every state for each I/p symbol there must be exactly one transition.

NFA
Non-deterministic FA

~~$\delta : Q \times (Q \cup \epsilon$~~

$\delta : Q \times (\Sigma \cup \epsilon) \rightarrow 2^Q$

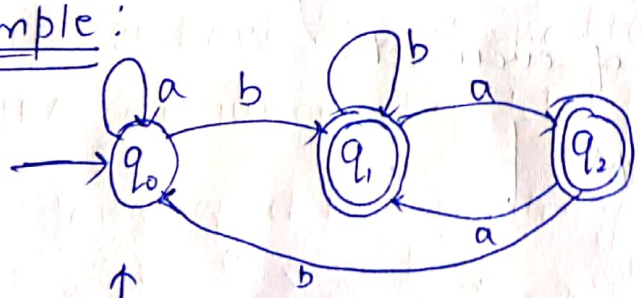
either reach to none of the state or multiple state $\{\phi \ldots\}$ → Power set

or

$\delta(q_i, a) \rightarrow A$

$q_i \in Q$
$A \subseteq Q$
$a \in \{\Sigma \cup \epsilon\}$

Example :



→ represents initial state, called empty edge

Scanned with CamScanner

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b\}$ → on which we are defining transitions

F = state with <u>double circles</u> $= \{q_1, q_2\}$

In FA, we can have single initial state but can have multiple finial states

$\delta_1(q_0, a) \rightarrow q_0$

$\delta_2(q_0, b) \rightarrow q_1$

***

→ Every DFA is also an NFA

→ Every NFA is not an DFA but we can convert it to equivalent DFA.

* <u>For checking DFA:</u>

Total function:

$$|Q \times \Sigma| = |Q| \times |\Sigma|$$

This is DFA as for every Q element we have exactly one transition:

q_0 से a and b का transition है।

q_1 से a and b का एक-एक transition है।

q_2 से a —, —, . ———

⇒ DFA ✓

aabba

q_0 q_0 q_0 q_1 q_1 q_2

final state

This string is acceptable

Scanned with CamScanner

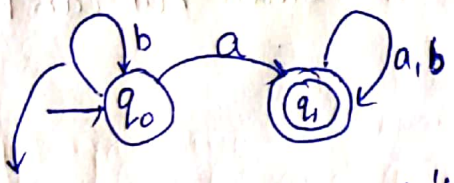
2. Find the DFA for the following Regular languages:

→ all the strings of ab have atleast one a

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

→ a is compulsory

No. of states are also not fixed.

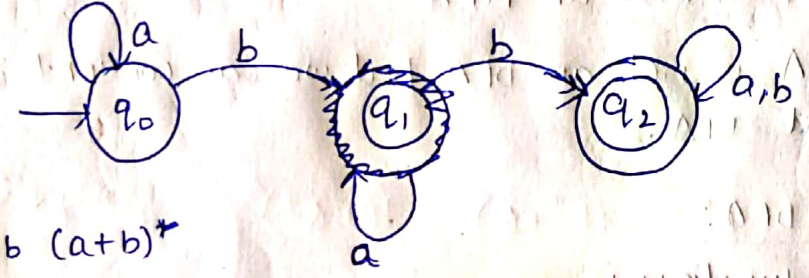


we can choose any no. of 'b'

↓

क्योंकि b के सारे function आगए

→ all the strings of ab have atleast 2 b's.

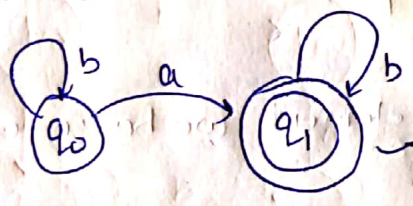


$(a+b)^* b b (a+b)^*$

→ all the strings of ab have exactly one a'

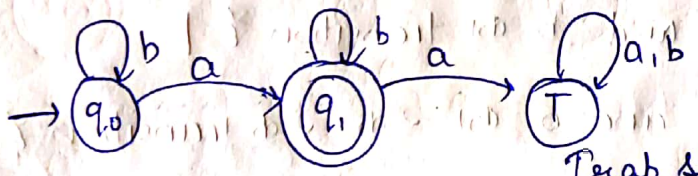
$b^* a b^*$

NFA → ✗



→ (T.F) is not followed

DFA

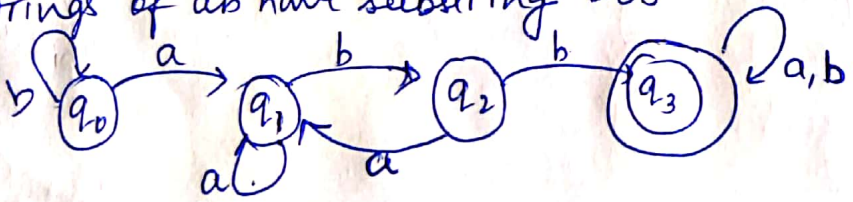


Trap state / Dummy state

→ Imp: DFA can have at most one trap state.

For NFA, no requirement of total function.

→ all the strings of ab have substring abb



Scanned with CamScanner

** $\boxed{\text{First prefrence to self loop}}$ → Very Imp remember this

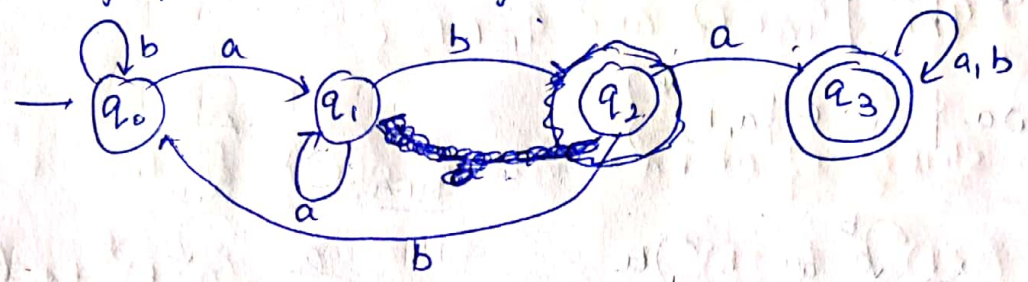
→ Here regenerating from $q_1 \xrightarrow{a} q_0$ and making aabb willn't be accepted by machine due to language constraint.

→ For trap state, if no options are there then choose trap state - Last option

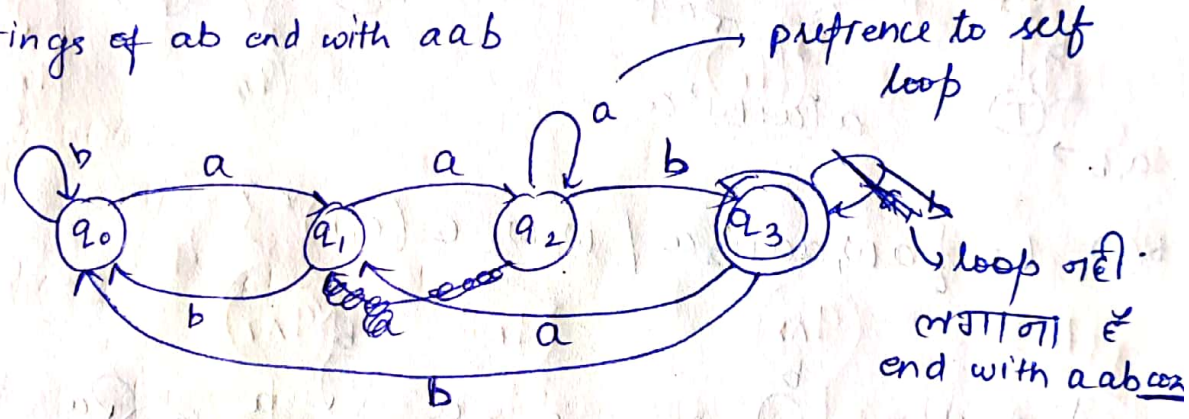
→ Here for q_2 trap state is also not acceptable as ababb will be there and machine willn't accept. → Trap state में जाके वापस नहीं आते।

* First self loop then uske near state then uske near to near state.

→ all the strings of ab have substring aba



→ all the strings of ab end with aab

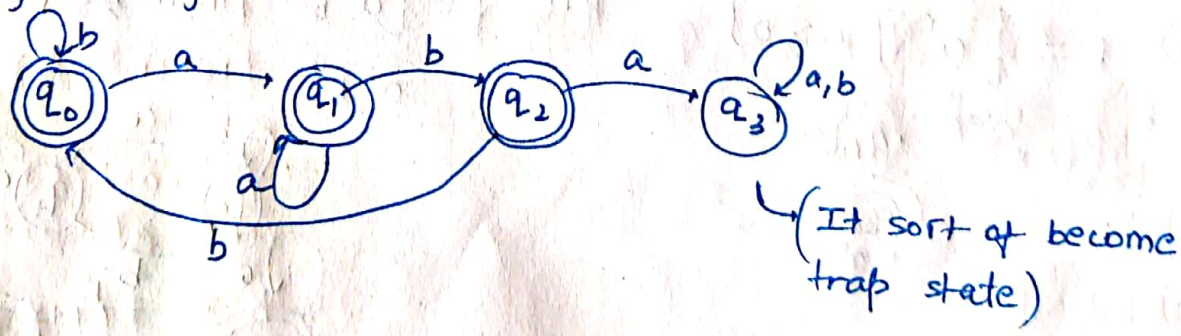


→ all the strings of ab do not have substring aba.

Complement of DFA:

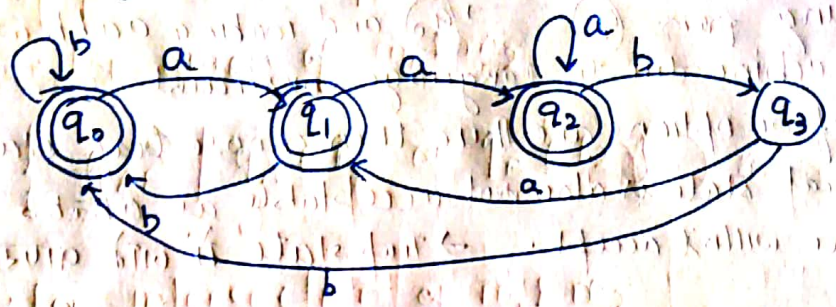
let $M(Q, q_0, \Sigma, \delta, F)$ is a DFA then its compliment $\bar{M}$ is M_1

$\bar{M}_1(Q_1, q_0, \Sigma, \delta, Q-F)$. In other words change final to non-final and non-final to final.



Scanned with CamScanner

→ All the strings of ab do not end with substring aab



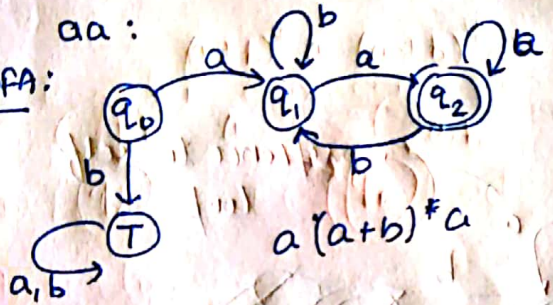
Language Pattern

$L = \{ \text{def}^n \mid \text{condition} \}$

(1) $L = \{ awa \mid w \in \{a,b\}^* \}$

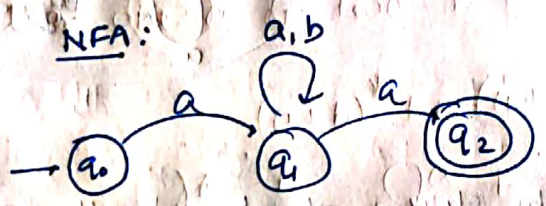
aa :

DFA:

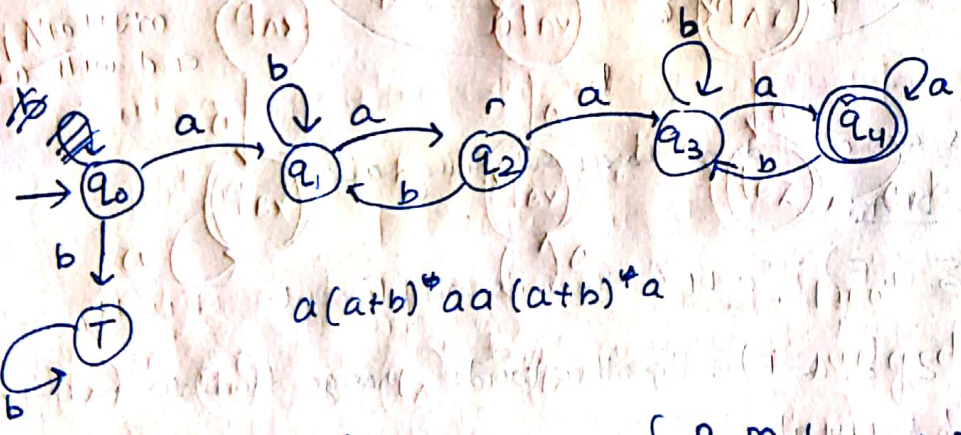


$a(a+b)^*a$

NFA:



(2) $L = \{ aw_1aaw_2a \mid w_1, w_2 \in (a,b)^* \}$



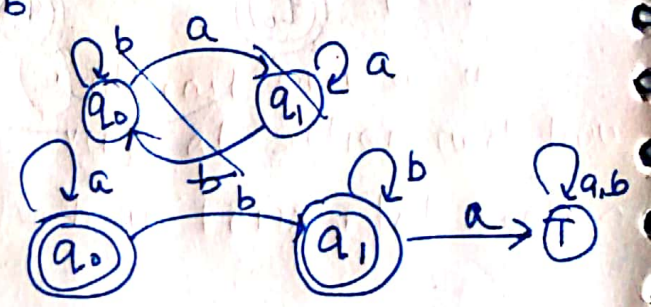
$a(a+b)^*aa(a+b)^*a$

(3) $L = \{ a^n \mid n \geq 0 \}$

a^*

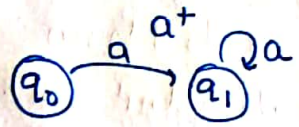
(4) $L = \{ a^n b^m \mid n, m \geq 0 \}$

$a^* b^*$



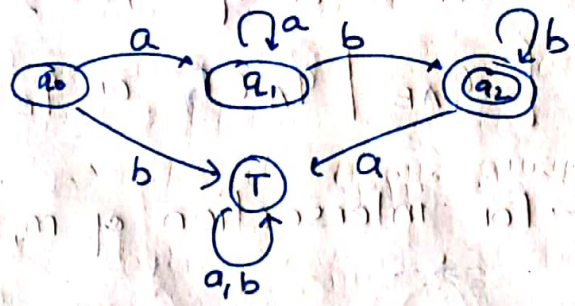
Scanned with CamScanner

* $L = \{ a^n \mid n \geq 1 \}$



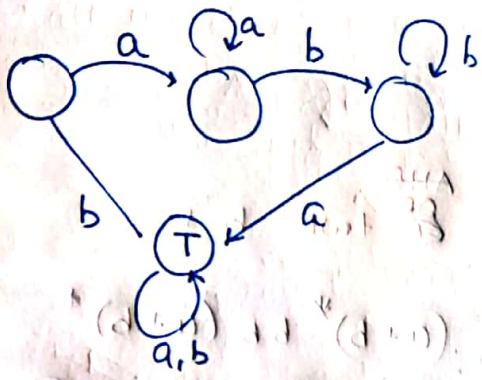
* $L = \{ a^n b^m \mid n, m \geq 1 \}$

$a^+ b^+$



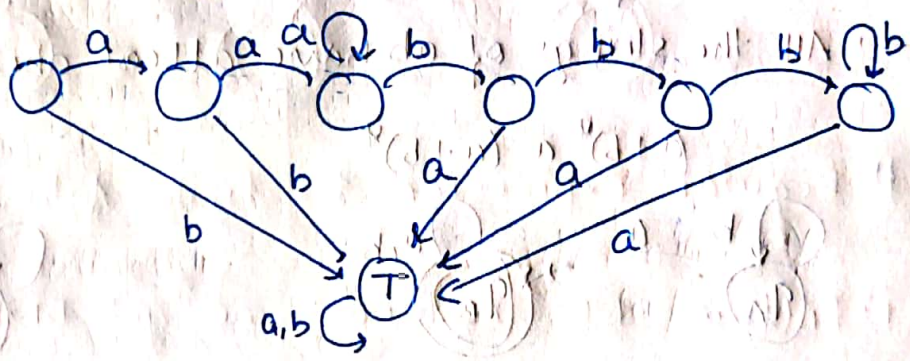
* $L = \{ a^n m^n \mid n \geq 1, m \geq 0 \}$

aa^*b^* or a^+b^*



* $L = \{ a^n b^m \mid n \geq 2, m \geq 3 \}$

aaa^*bbbb^*



* <u>Union and Intersection of Two DFA's</u>:

let m_1 & m_2 are two DFA's

where $m_1 (Q_1, q_x, \Sigma, \delta_1, F_1)$

$m_2 (Q_2, q_y, \Sigma, \delta_2, F_2)$

Now define one more DFA $m_3 (Q_3, q_z, \Sigma, \delta_3, F_3)$ by cross product of m_1 & m_2 where

$$Q_3 = Q_1 \times Q_2$$

$$q_z = q_{xy}$$

$$\delta_3 = \delta_1 \times \delta_2$$

$$\forall \ \delta_1 (q_A, a) \longrightarrow q_C \in \delta_1$$

$$\forall \ \delta_2 (q_B, a) \longrightarrow q_D \in \delta_2$$

$$\forall \ \delta_3 (q_{AB}, a) \longrightarrow q_{CD} \in \delta_3$$

Scanned with CamScanner

F_3 = Set of final states of m_3 $\quad F_3' \in Q_3$

→ For union of m_1 & m_2

$$F_3 = \{ q_{AB} \mid q_A \in F_1 \text{ or } q_B \in F_2 \}$$

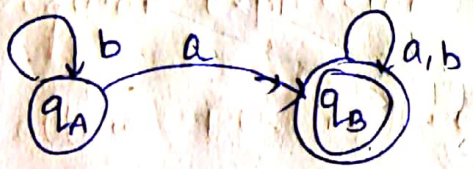
→ For intersection of m_1 & m_2

$Q_1 \times Q_2$

$$F_3 = \{ q_{AB} \mid q_A \in F_1 \text{ and } q_B \in F_2 \}$$

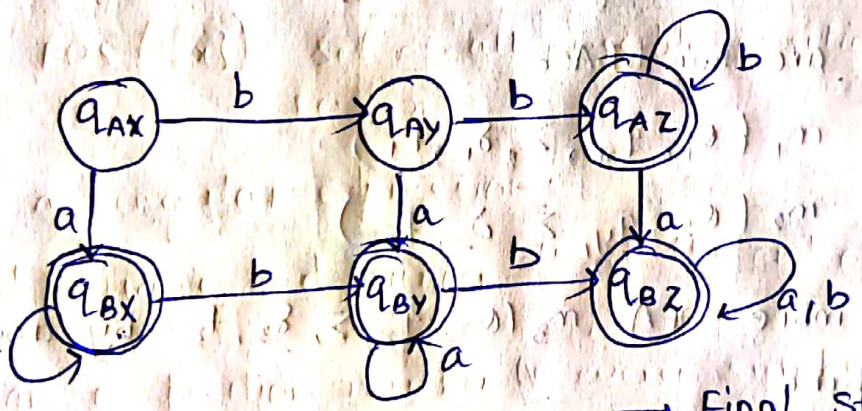
→ All the strings of ab have atleast one a ~~AND~~ OR two b's

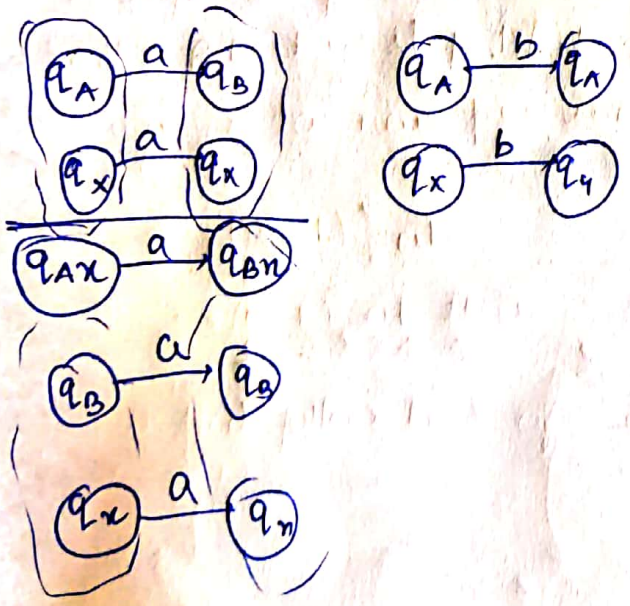
$(a+b)^* a (a+b)^*$



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

DFA





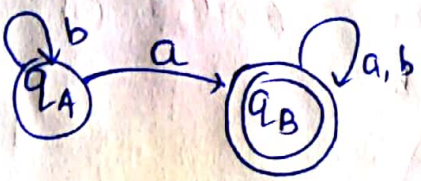
Union

→ Final state will be judged by in which final state of either m_1 is present or of m_2

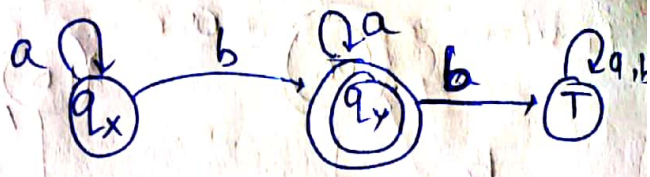
→ Atleast one a in vertical section and two b in horizontal section

→ All the strings of ab have atleast one a & exactly one b

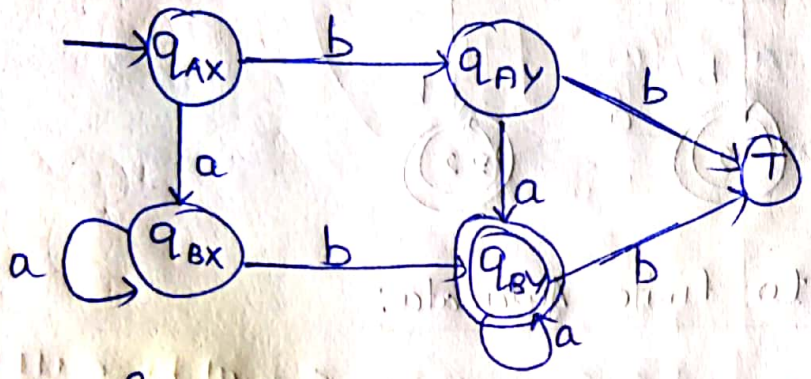
$(a+b)^* a (a+b)^*$



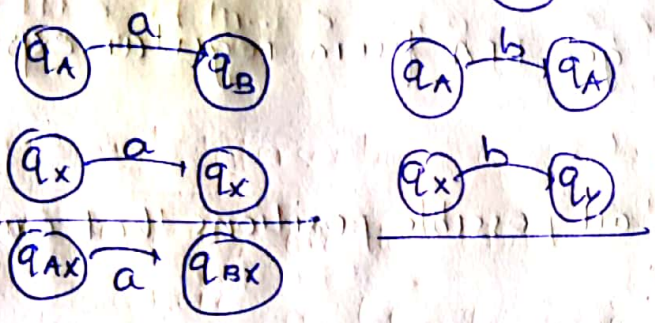
$a^* b a^*$



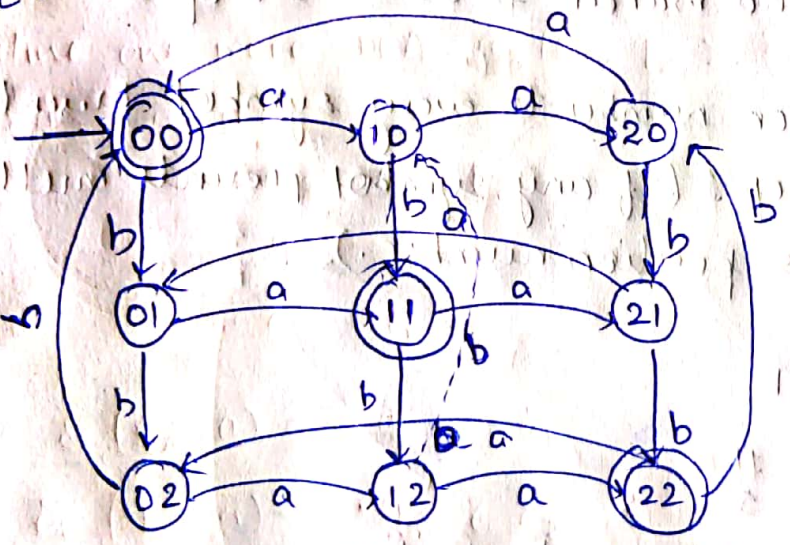
→ No need to prepare combination of Trap state, only single trap state.



→ Here final state will be q_{BY} as final state of both m_1 & m_2 needs to be included.



* $L = \{ |n_a(w) - n_b(w)| \bmod 3 = 0 \mid w = \{a,b\}^* \}$



0 0 → Represent b

↓ Represent a

Final state

$n_a(w) - n_b(w)$

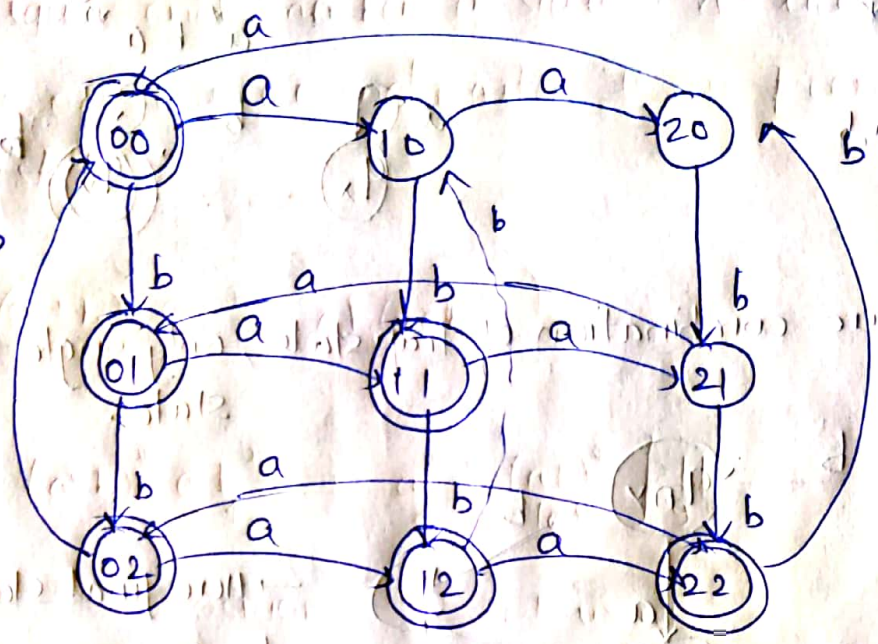
No. of strings in a − No. of strings in b

$|0-0| = 0$

$|0-1| = 0$

Scanned with CamScanner

* $L = \{ n_a(w) \bmod 3 \le n_b(w) \bmod 3 \mid w \in (a,b)^* \}$



Regular expression to Finite Automata:

(i) Follow POS method: we get DFA from regular expression directly.

Follow POS ({x}) = set of all possible positions that can follow POS 'x'.

Let R is a regular expression:

Step 1: Add unique end marker '#' at the end of R. R#

Step 2: Assign a position number to every symbol present in R updated R.E by step 1 (if any symbol presents multiple times, we assign it a new position number.)

Eg: $R = (a+b)^* ab$

$R\# = \underset{1}{(}\underset{2}{a+b}\underset{1}{)}^* \underset{3}{a}\underset{4}{b}\underset{5}{\#}$

Step3: Now find follow POS of each position.

Scanned with CamScanner

Positions	Follow-POS
{1}	{1, 2, 3}
{2}	{1, 2, 3}
{3}	{4}
{4}	{5}
{5}	—

a · a a · b a · a
1 1 1 2 1 3

b के बाद a आ सकता है after loop

$(a+b)^*$ a b #
1 2 3 4 5

→ ~~All the positions a~~

Step 4: Identify initial state: set of all positions from where regular expression can began will represent initial state.

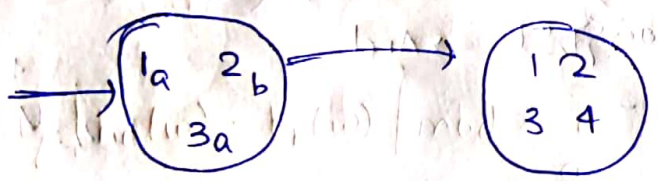
→ can began from 3 when $(a+b)^*$ is empty

Step 5: To identify transistions of each state, first find positions that represent input symbol.

$\delta(\{1,2,3\}, a)$ = Follow POS of position

Position 1, 3 represents a = Follow POS ({1,3}) ∪ Follow POS ({3})

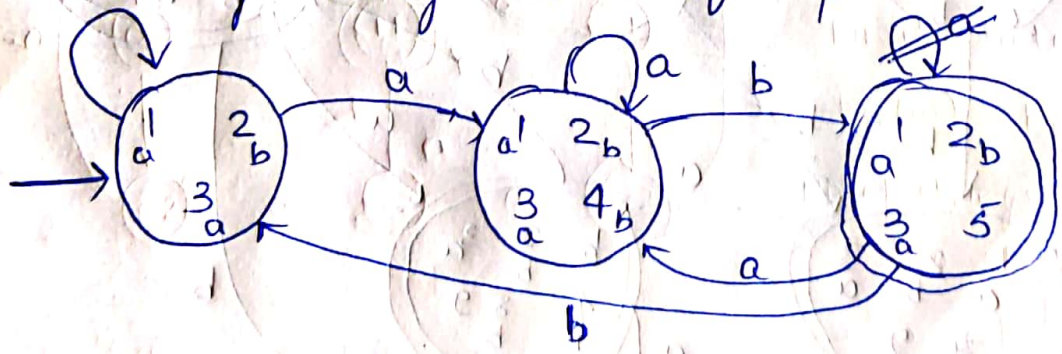
= {1, 2, 3} ∪ {4} = {1, 2, 3, 4}



$\delta(\{1,2,3\}, b)$ = Follow POS ({2})

= {1, 2, 3}

Already existing means self loop



Step 6: To identify final states, we have to check the position of unique end marker in states. All the states in which position of unique end marker '#' are present will become the final state.

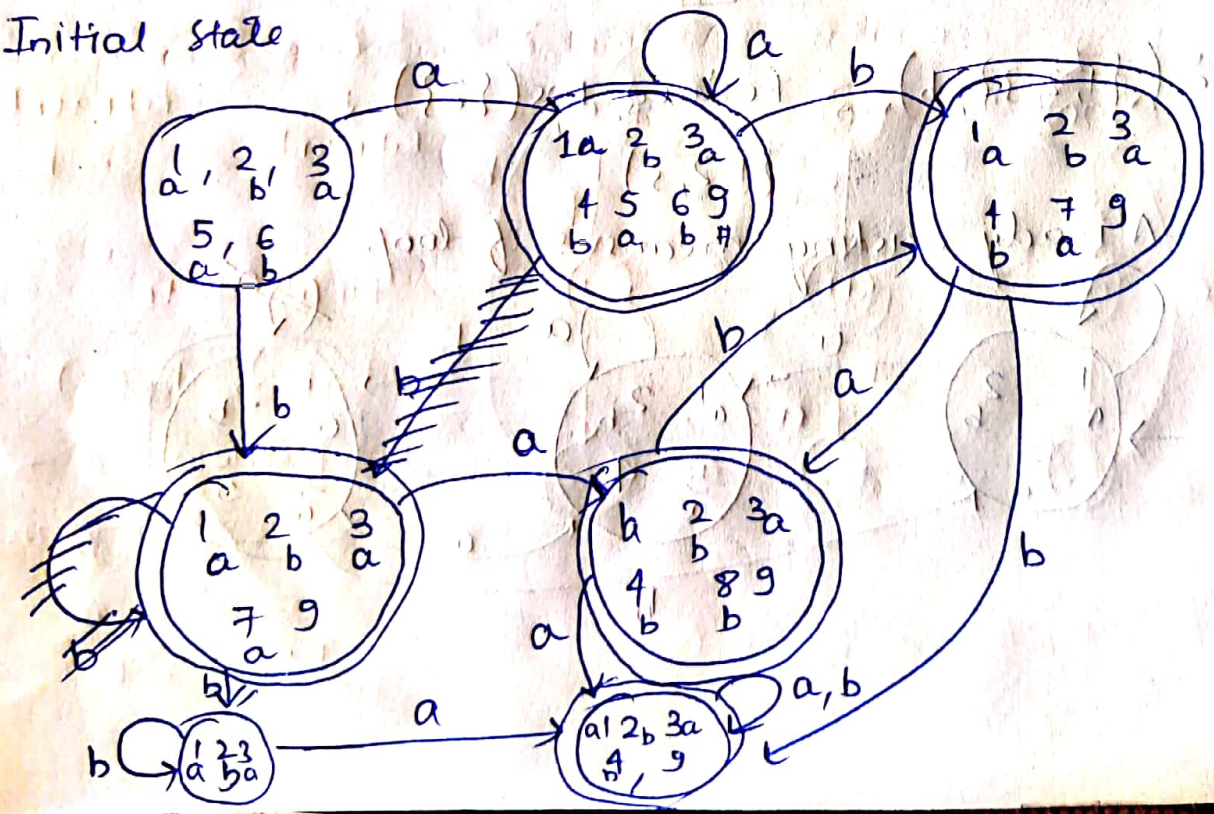
Q. $\left((a+b)^* a b^* + ab(ab)^*\right)$

$$\left[\underset{1\ 2}{(a+b)^*}\ \underset{3}{a}\ \underset{4}{b^*} + \underset{5}{a^*}\ \underset{6}{b}\ \underset{7\ 8}{(ab)^*}\right]\underset{9}{\#}$$

Positions	Follow up
{1}	{1,2,3}
{2}	{1,2,3}
{3}	{4,9} when 4 will be null we will reach 9
{4}	{4,9}
{5}	{5,6}
{6}	{7,8,9}
{7}	{8} After 7 definitely 8 will be there
{8}	~~{9}~~ {7,9} And
{9}	—

1,2,3 4, 7,9
1,2,3 4,8,9

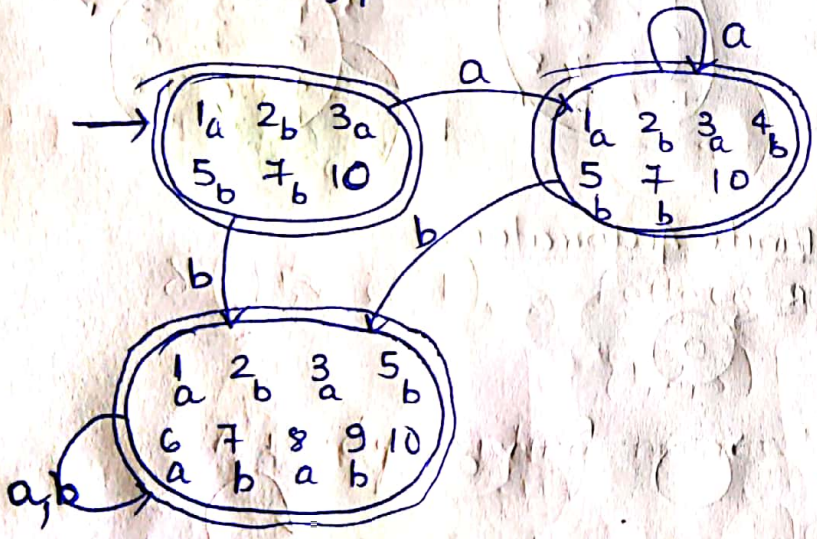
Initial state



Q. $((a+b)^* (ab)^* + (ba)^* b(a+b)^*)^*$

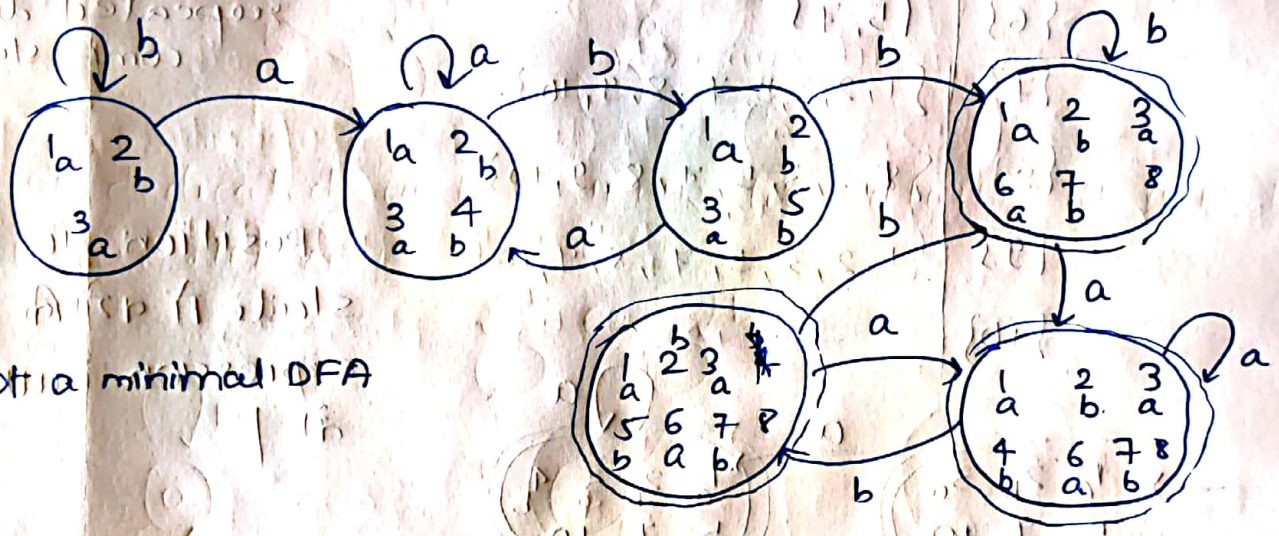
$$\left[\underset{1}{(a+b)^*} \underset{2}{(ab)^*} + \underset{5}{(ba)^*} \underset{6}{b} \underset{7}{(a+b)^*} \underset{8\ 9}{} \right]^* \underset{10}{\#}$$

{1}	{1, 2, 3, 5, 7, 10}	due to complete star
{2}	{1, 2, 3, 5, 7, 10}	
{3}	{4}	
{4}	{1, 2, 3, 5, 7, 10}	$(a+b)^*$ को repeat due to complete star
{5}	{6}	$(a+b)^* (ab)^*$ can be repeated due to comp. star
{6}	{5, 7}	
{7}	{1, 2, 3, 5, 7, 8, 9, 10}	
{8}	{8, 2, 3, 5, 7, 8, 9, 10}	
{9}	{1, 2, 3, 5, 7, 8, 9, 10}	Position 10 initial state में क्योंकि पूरा expression null हो रहा तो।
{10}	—	



$(a+b)^* \; abb \; (a+b)^* \; \#$
1 2 3 4 5 6 7 8

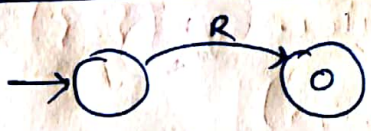
POS	Follow POS
{1}	{1,2,3}
{2}	{1,2,3}
{3}	{4}
{4}	{5}
{5}	{6,7,8}
{6}	{6,7,8}
{7}	{6,7,8}
{8}	{6,7,8}



→ Not a minimal DFA

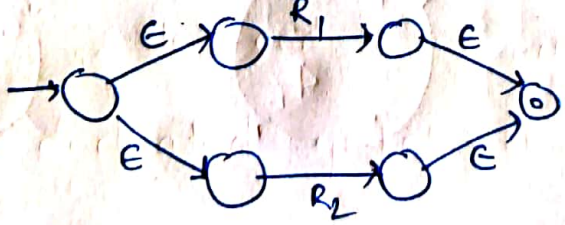
Regular Expression to Finite Automaton:

1. R



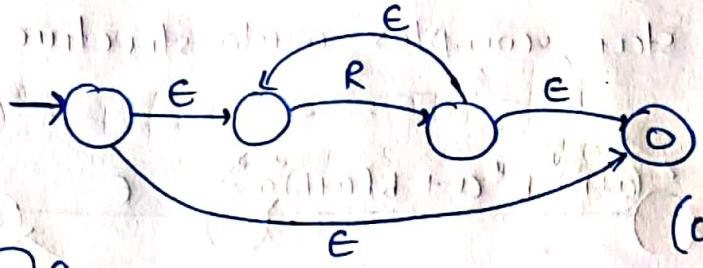
2. $R_1 . R_2$

3. $R_1 + R_2$



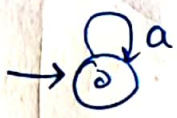
Scanned with CamScanner

4. R^*



⇒ For complex we prefer this one (complete star & complex str. हो तो ये use करना)

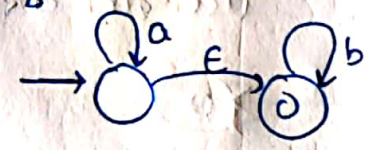
Eg: a^*



$(ab)^*$

$(a+b)^*$

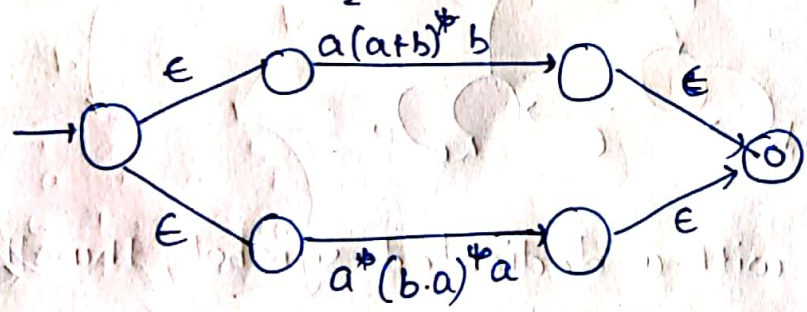
a^*b^*



5. ε

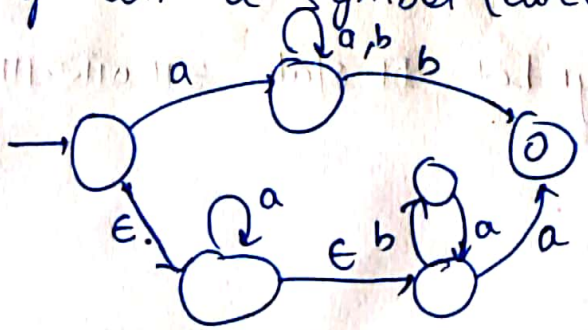
6. ϕ (Empty set)

Eg: $\underbrace{(a(a+b)^*b}_{R_1} + \underbrace{a^*(b.a)^*a}_{R_2})$



It's a FA not dFA

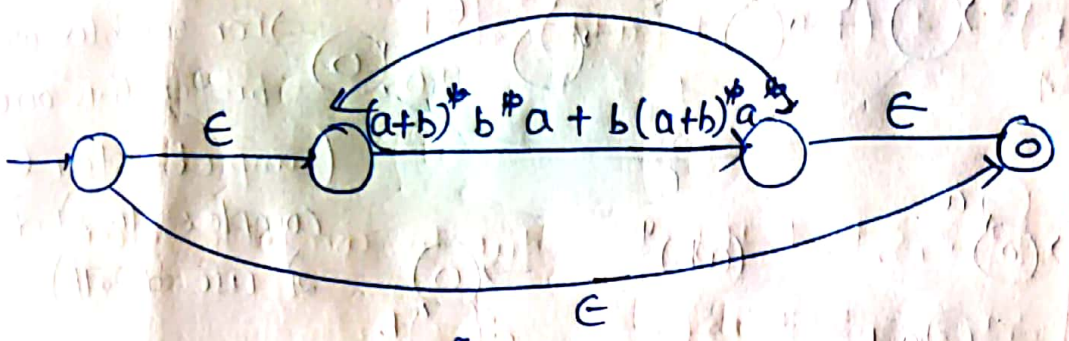
In R_1 we have $a(a+b)^*b$, we have particular symbol 'a', we can go with a symbol (directly start it).

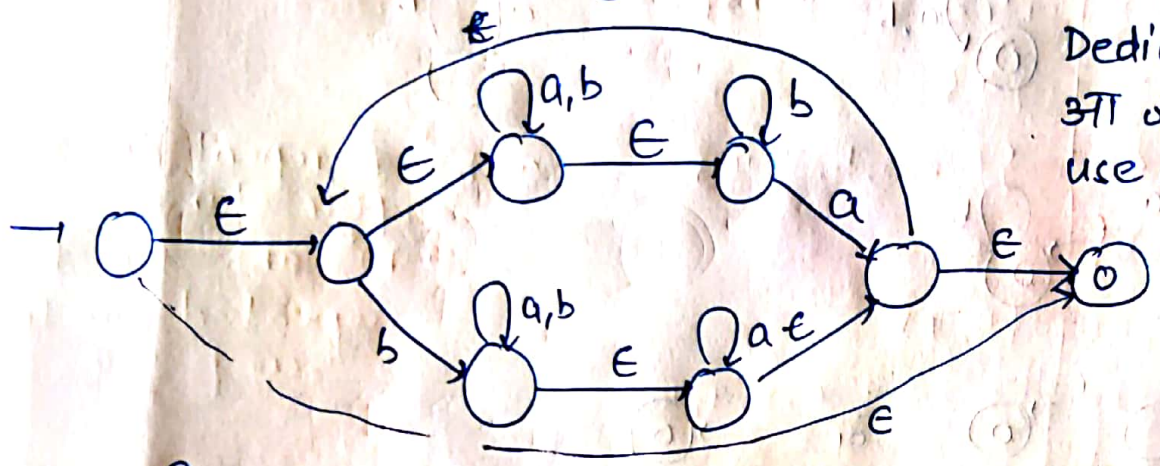


Scanned with CamScanner

Eg: $\left((a+b)^* b^* . a + b(a+b)^* a^*\right)^*$

Complete star, complex wala structure

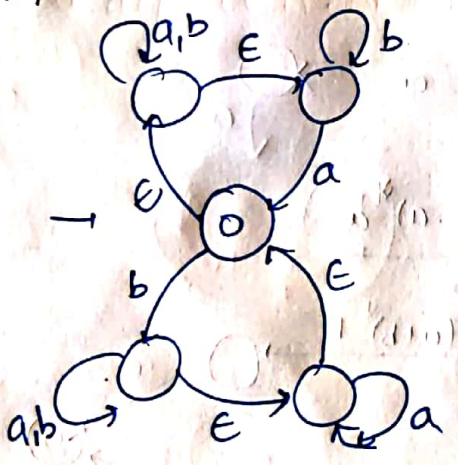




Dedicated symbol आ जाए तो null use नहीं करना।

→ एक भी dedicated symbol आ जाए like $a(a+b)^*$ तो $(a+b)^*$ में null use नहीं होगा

or



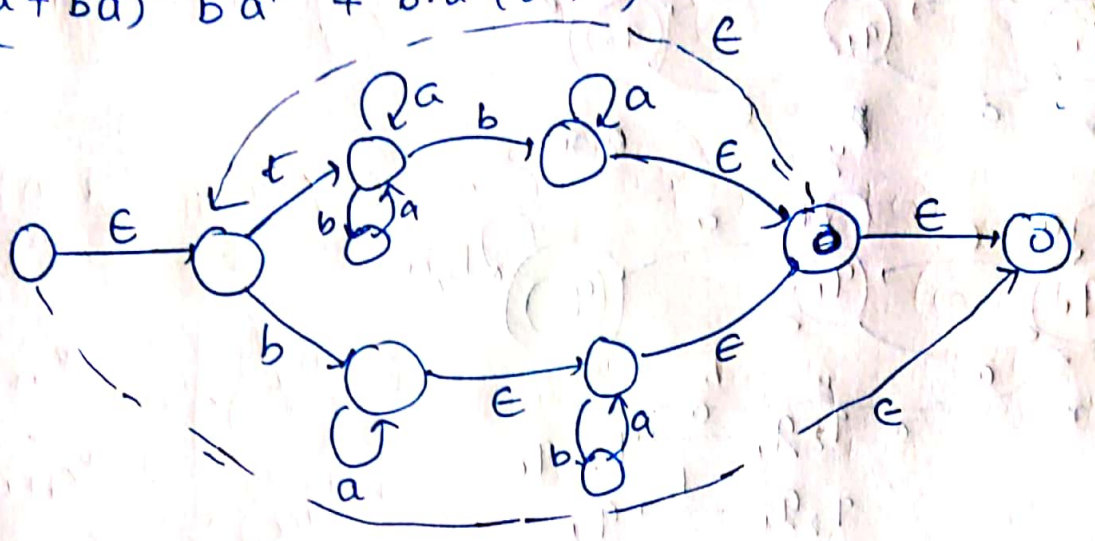
If we have subpart with a dedicated symbol, then there is no need to separate it out with a null.

ba^* ; No need to use null here

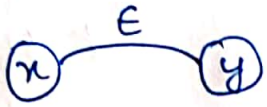
→ Null की जगह dedicated symbol का state आ जाएगा।

Scanned with CamScanner

Eg: $\left((a+ba)^* ba^* + b.a^*(ba)^*\right)^*$



Remove 'ε' transistions from F.A:



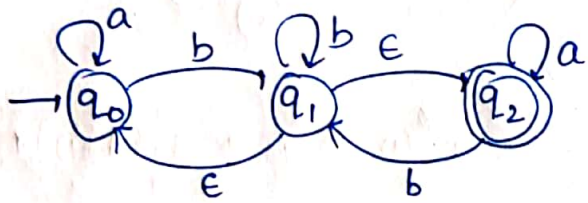
(i) If x is an initial state, then make 'y' also initial.

(ii) If y is a final state, then make 'x' also final.

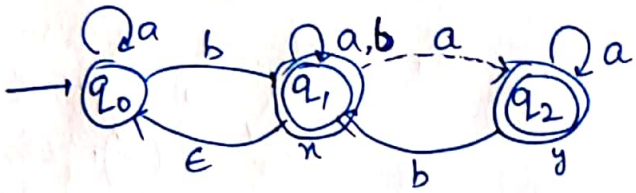
(iii) Replicate all the transitions of 'y' from 'x'.

Eg:

(i)

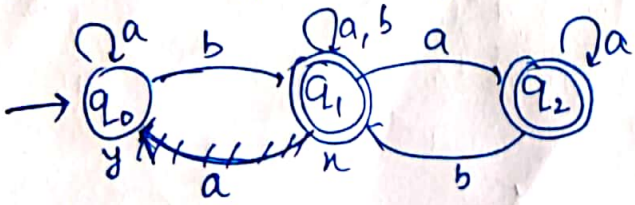


* This is F.A, Not DFA



$q_1 \cancel{q_2} \xrightarrow{a} q_2$

$q_1 \cancel{q_2} \xrightarrow{b} q_1$

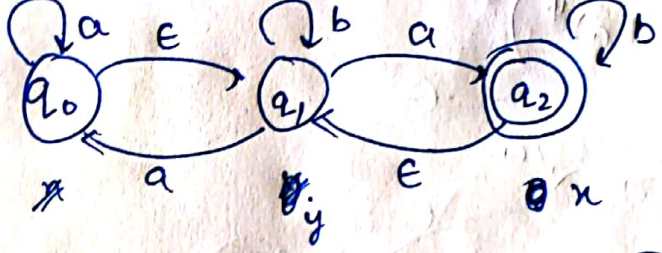


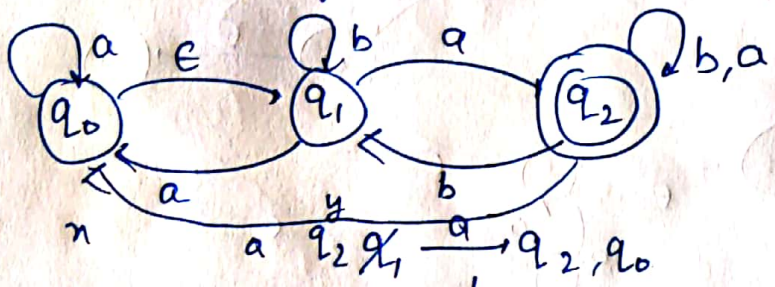
$q_1 \cancel{q_0} \xrightarrow{a} q_0$

$q_1 \cancel{q_0} \xrightarrow{b} q_1$

Scanned with CamScanner

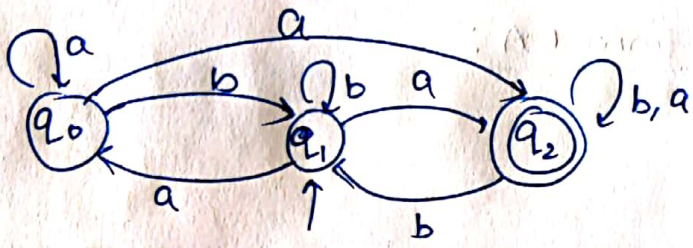
Eg:





$q_2 q_1 \xrightarrow{a} q_2, q_0$

$q_2 q_1 \xrightarrow{b} q_1$



$q_0 q_1 \xrightarrow{a} q_2, q_0$

$q_0 q_1 \xrightarrow{b} q_1$

But according to the def^n of FA it is not correct