REGULAR LANGUAGE & FINITE AUTOMATA

$F \subseteq Q$

- Machine notation :- $M(Q, \Sigma, \delta, q_0, F)$ 5-tuple

Final states

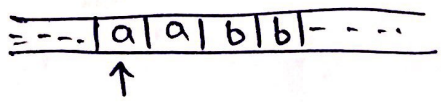
Set of internal states | input alphabet | state transition | starting state

$q_0 \in Q$

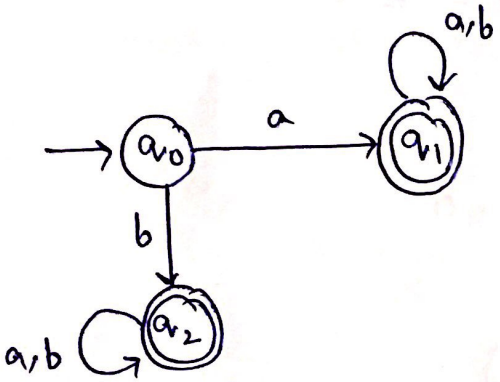
$\delta_{Q \times \Sigma \to Q}$ (DFA)

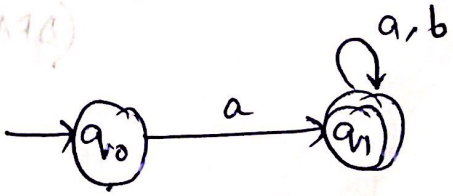
$\delta_{Q \times \Sigma \to 2^Q}$ (NFA)

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



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





Dead Config (no state for I/p b at q_0)

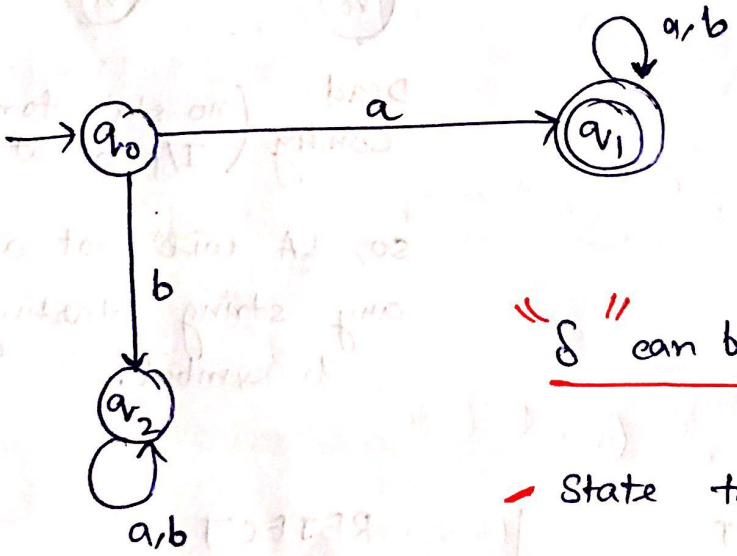
so, FA will not accept any string starting with b symbol.

	ACCEPT	REJECT
DFA	Reach F	Not Reach F
NFA	Reach F	Not Reach F + Dead End

	DFA	NFA
Choice	Not allowed	Allowed
Dead Config	Not allowed	Allowed
ϵ - move	Not allowed	Allowed

- $F = \emptyset$ Accept no string $L(M) = \emptyset$
- $F = Q$ Accept all string $L(M) = \Sigma^*$

(DFA)



"δ" can be represent by:-

- State transition Diagram
- State transition Table
- Function Representation

State transition table :-

δ_n	a	b
$\rightarrow q_0$	q_1	q_2
(q_1)	q_1	q_1
q_2	q_2	q_2

Function Representation :-

- $\delta(q_0, a) = q_1$
- $\delta(q_0, b) = q_2$
- $\delta(q_1, a) = q_1$
- $\delta(q_1, b) = q_1$
- $\delta(q_2, a) = q_2$
- $\delta(q_2, b) = q_2$

$[q_0 = q_0, \; F = q_1]$

Q. No. of δ func^n commands for 10 states DFA for ternary alphbets.

$\delta_{Q \times \Sigma} \rightarrow Q$

10 , 3

so, total $10 \times 3 = 30$ commands

But for NFA without ϵ and without choice $\delta \leq 30$ as Dead configuration is allowed

so,

DFA

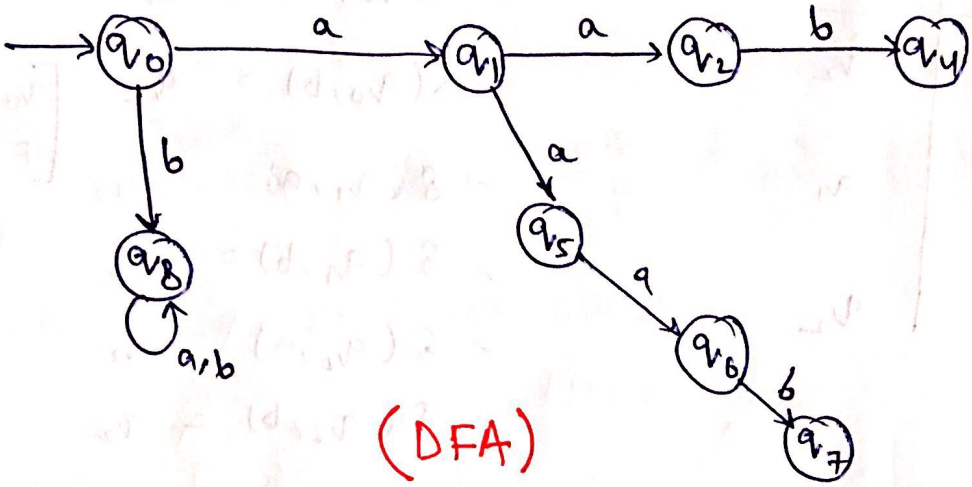
$|\delta| \rightarrow |Q \times \Sigma|$

NFA

without ϵ : without choice $\delta \leq Q \times \Sigma$

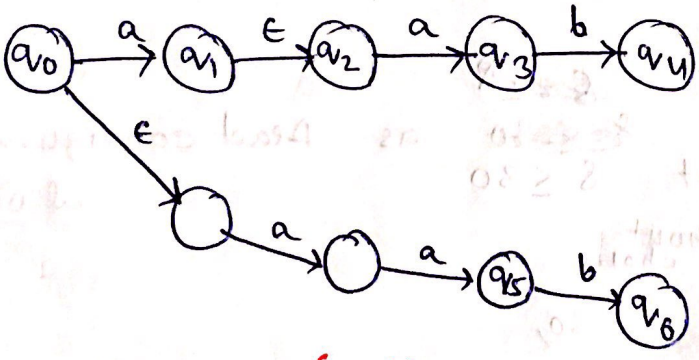
with ϵ : with choice $\delta \leq 2^Q \times \Sigma$; any value greater or less

= δ^* (extended transition func^n) :-



(DFA)

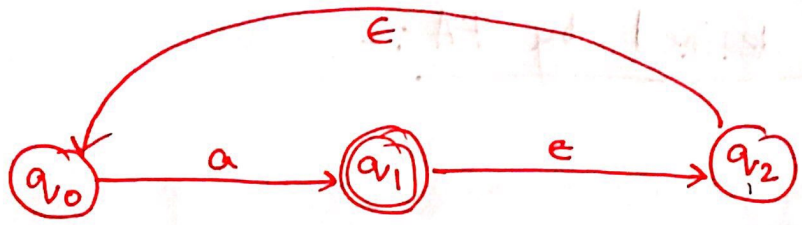
- $\delta^*(q_0, aab) = q_4$
- $\delta^*(q_0, aaab) = q_7$
- $\delta^*(q_0, bab) = q_8$



(NFA)

- $\delta^*(q_0, aab) = \{q_4, q_6\}$

Q.



$$\delta^*(q_0, a) = \{q_0, q_1, q_2\}$$

$$\delta^*(q_1, a) = \{q_0, q_1, q_2\}$$

$$\delta^*(q_2, a) = \{q_0, q_1 q_2\}$$

$$\delta^*(q_0, \epsilon) = \{q_0\}$$

$$\delta^*(q_1, \epsilon) = \{q_1, q_2, q_0\}$$

$$\delta^*(q_2, \epsilon) = \{q_2, q_0\}$$

Properties of δ^* :-

DFA

(i) $\delta^*(q, \epsilon) = q$

No, ϵ-moves

(ii) $\delta^*(q, xy) = \delta^*(\delta^*(q, x), y)$

(iii) $\delta^*(q, x) = \delta^*(q, y)$

$\Rightarrow \delta^*(q, xz) = \delta^*(q, yz)$

Right Invariance

NFA

(i) $q \in \delta^*(q, \epsilon)$

as NFA contains ϵ-moves

(ii) $\delta^*(q, x) = q_i \quad \forall\ i = 1, 2, 3 \ldots n$

$$\delta^*(q, xy) = \bigcup_{i=1}^{n} \delta^*(q_i, y)$$

(iii) same as DFA

Language defined by FA :-

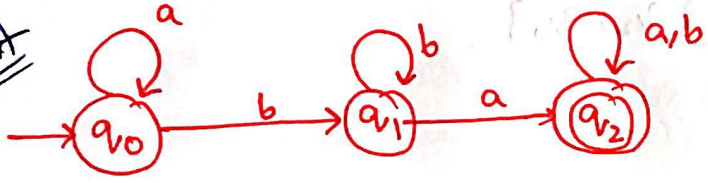
DFA $\longrightarrow$ $L(M) = \{ w \in \Sigma^* \mid \delta^*(q_0, w) \in F \}$

one of final state

NFA $\longrightarrow$ $L(M) = \{ w \in \Sigma^* \mid \delta^*(q_0, w) \cap F \neq \emptyset \}$

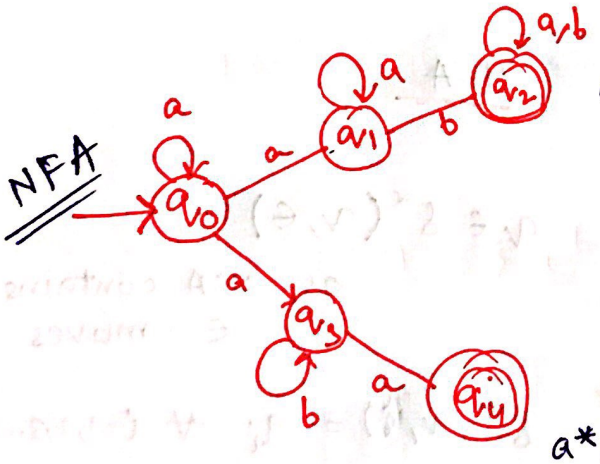
set contain atleast one final state

DFA



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

NFA



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

a^*ab^*a

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

Theorems :-

$p \to q$ ✓
$\sim q \to \sim p$ ✓
$q \to p$ ✗
$\sim p \to \sim q$ ✗

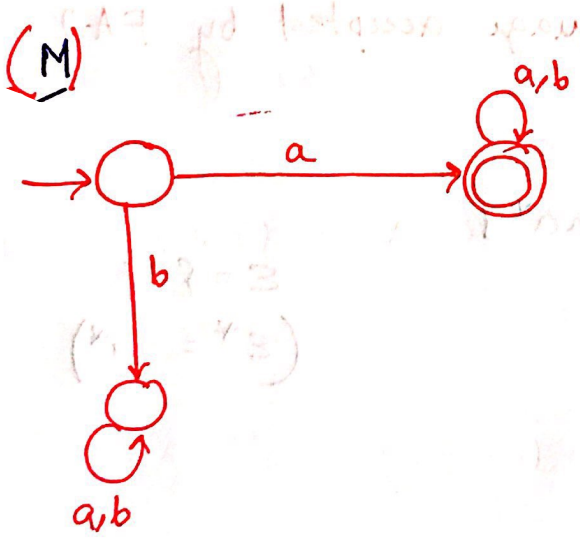
1. $L(\overline{M}) = \overline{L(M)}$ (only in DFA)

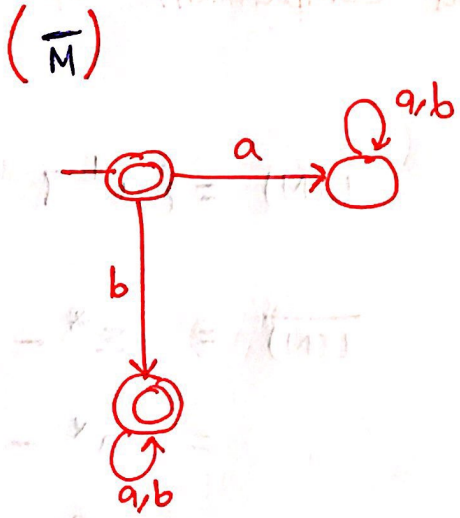
2. Finite ⇒ Regular i.e every finite language is REG ~~then~~ vice versa not true.

3. Pumping Lemma :- L is REG ⇒ L satisfies the pumping lemma for REG lang

4. Myhill-nerode theorem :- L is REG ⇔ No. of myhill-nerode equivalent classes is Finite

5. Kleen's theorem :- L is REG ⇔ L is acceptable by FA which accept L





$L(M) = a(a+b)^*$

$L(M)$ = starting with a

$\overline{L(M)} = \epsilon + b(a+b)^*$

(Not start with a)

$L(\overline{M}) = \epsilon + b(a+b)^*$

$(L(\overline{M}) = \overline{L(M)})$

Q. Which is False ?

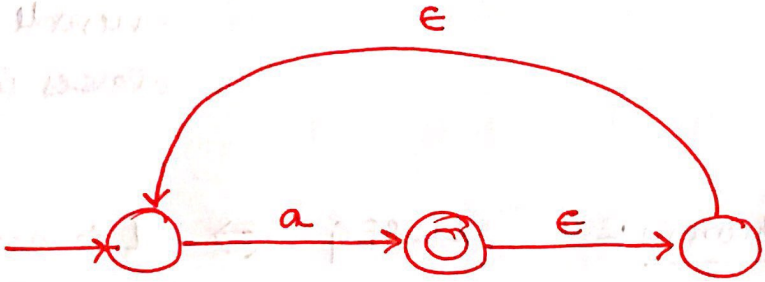
(a) Every finite lang is REG

(b) Every non-regular lang is infinite

(c) ✓ Every infinite lang is non-regular

(d) None of these.

Q.



Find compliment of Language accepted by FA?

$L(M) = \{a^+\} = \{aa^*\}$

$\Sigma = \{a\}$

$(\Sigma^* = a^*)$

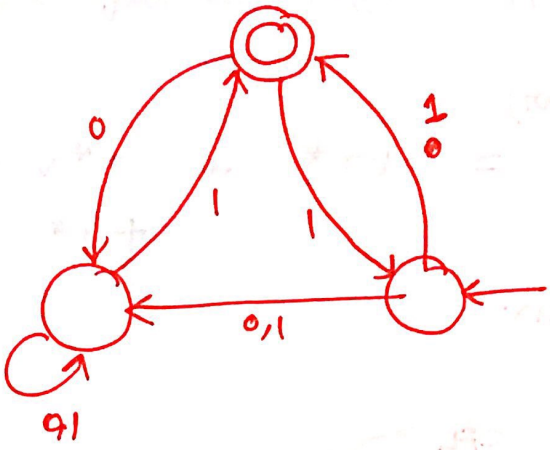
$\overline{L(M)} = \Sigma^* - L(M)$

$= a^* - a^+$

$= \epsilon$

NOTE:- Since FA is NFA so $(L(\bar{M}) \neq \overline{L(M)})$

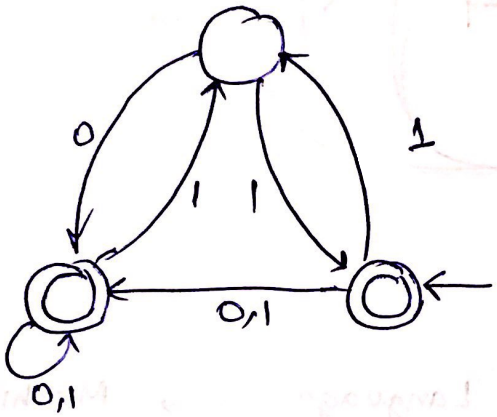
Q.



Find Language accepted by complement M/c.

- It is NFA :- $L(\bar{M}) \neq \overline{L(M)}$

so, $(\bar{M})$



so, $(L(\bar{M}) = (0+1)^*)$

⇒ Kleen's theorem :-

L is Regular ⇔ ∃ a FA which accepts L

⇔ ∃ a DFA which accept L

⇔ ∃ a NFA which have single final state

⇔ ∃ a Reg Exp for L.

⇔ ∃ a Regular grammer for L

⇔ ∃ a Right linear grammer for L

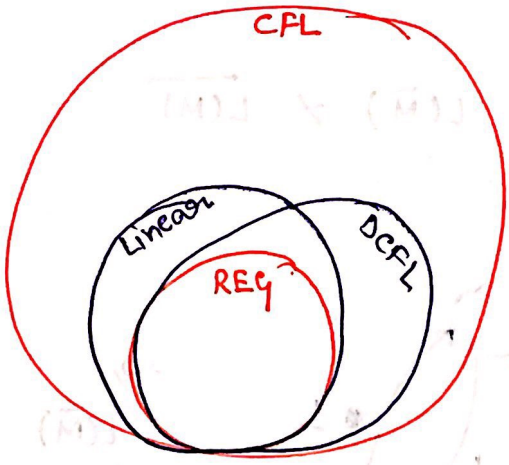
⇔ ∃ a Left linear grammer for L

* coz, can convert Right to left linear & vice versa

(Not Regular)

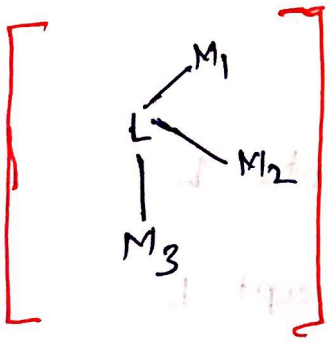
(Linear Grammer = VT^* + T^*VT^* + T^*V + T^*)

combination of Left & Right linear



⇒ FA design :-

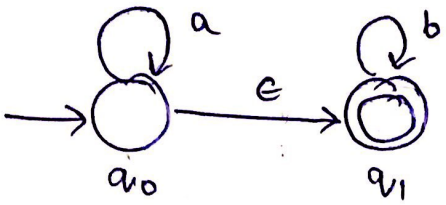
Language ⟶ Machine (FA)



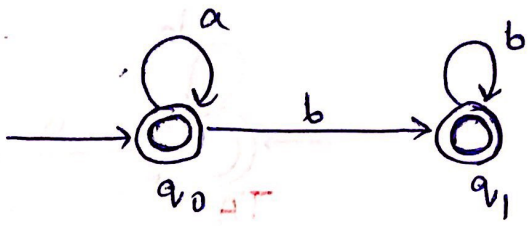
- A Language can have more than one DFA or NFA.
- A regular lang have a unique minimal DFA

- Two different REG lang can't have same unique minimal DFA, but number of states in DFA can be same..
- A REG lang can't have a unique minimal NFA. But number of states in that NFA will be same.

eg:-

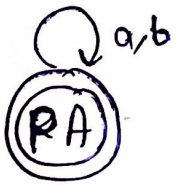


$a^* b^*$



Design Guidelines:-

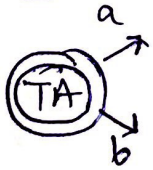
- Be stingy with states (Try reusing every state)
- $|\omega_{min}| = n \Rightarrow n(DFA_{min}) \geq n+1$
  - $|\omega_{min}|$: min string in DFA
  - $n(DFA_{min})$: no. of states
- Fill up the missing arcs
- Only 4 types of states in minimal DFA: $\Sigma = \{a, b\}$



Permanent Accept

eg:- checks substring present

Permanent Reject

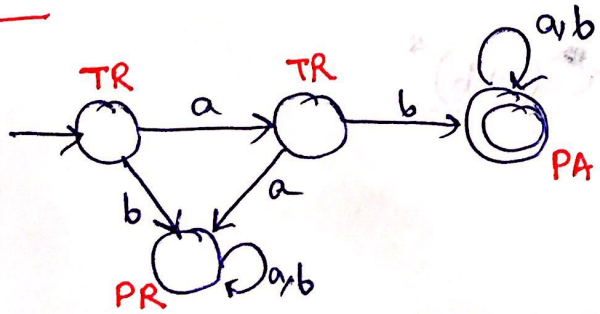


Temporary Accept

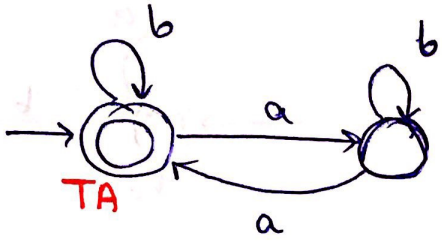
eg:- MOD func, or EVEN / ODD

Temporary Reject

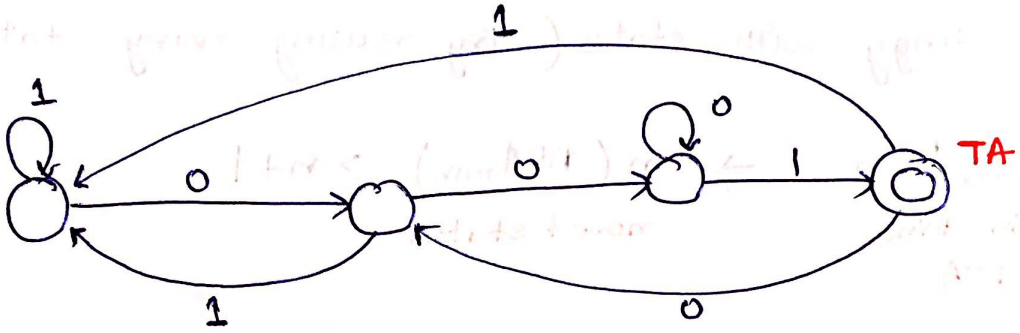
$L = ab(a+b)^*$



L = {even no. of a}



L = {(0+1)* 001}



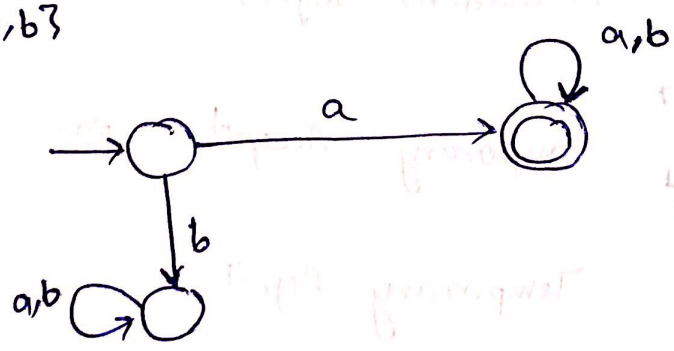
Design of some common FA's :-

TYPE - 1 (pattern matching)

1. L = {starting with "a"}

Σ = {a,b}

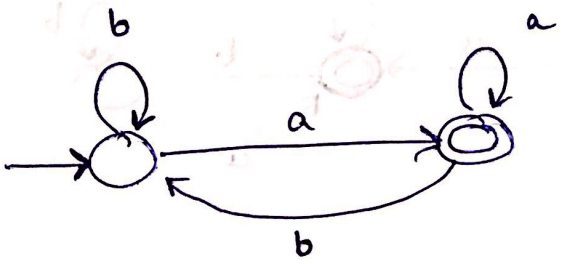
(DFA :-)



RE :- a (a+b)*

2. L = { ends with 'a' }

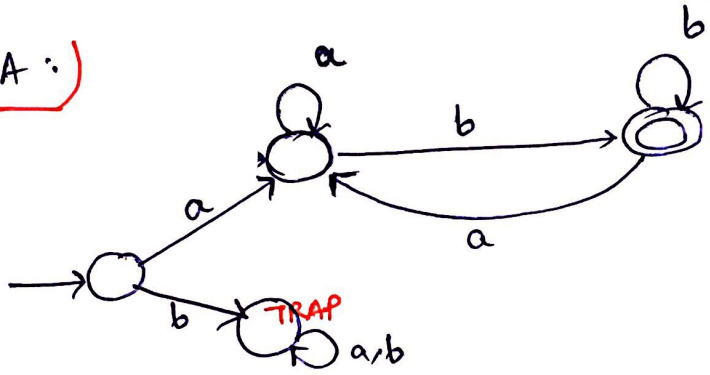
(DFA:)



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

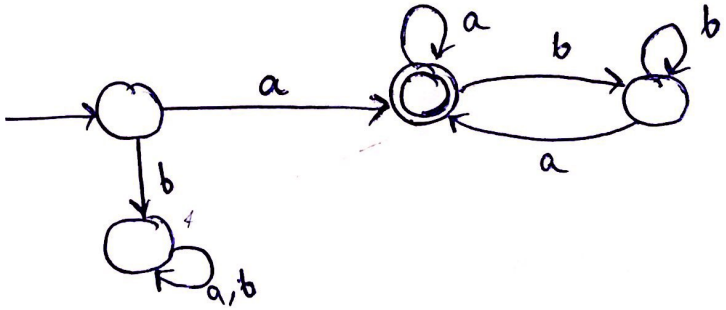
3. L = { starts with a and ends with b }

(DFA:)



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

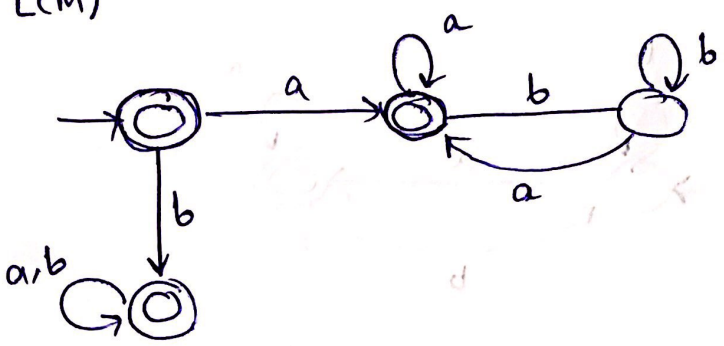
4. L = { start with a and ends with a }



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

5. L = { Not starting with a & not ending with b }

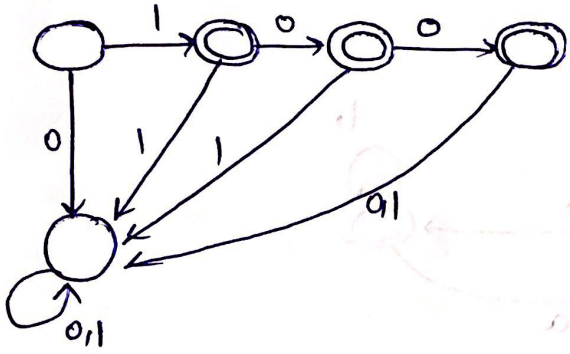
$\overline{L(M)} = L(\overline{M})$



Make M/c (FA) for $\overline{L}$ and then $\overline{M}$ we will get L(M).

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

6. L = {100, 10, 1}

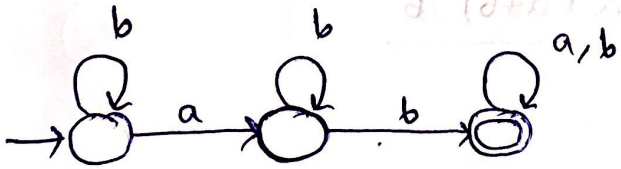


RE = (100 + 10 + 1)

= 1(00 + 0 + ϵ)

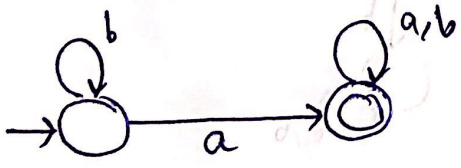
= 1(0(0+ϵ)+ϵ)

7. L = {containing substring 'ab'}



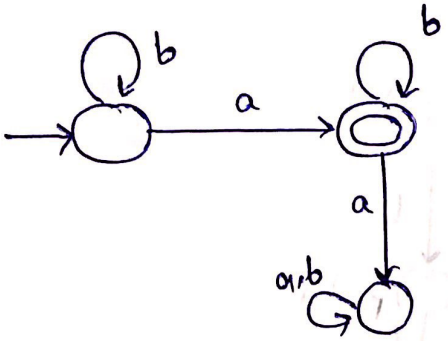
RE = $(a+b)^*ab(a+b)^*$

8. L = { contain a ≥ 1 }



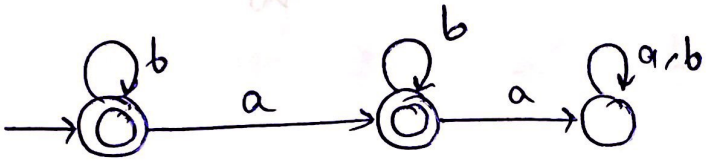
RE = $(a+b)^*a(a+b)^*$

9. L = { contain exact one 'a' }



RE = $b^* a b^*$

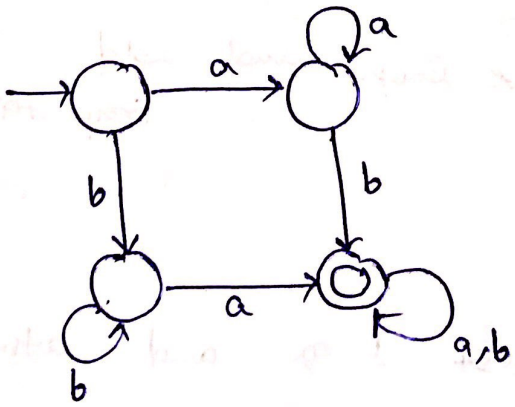
10. L = { contain atmost one 'a' }



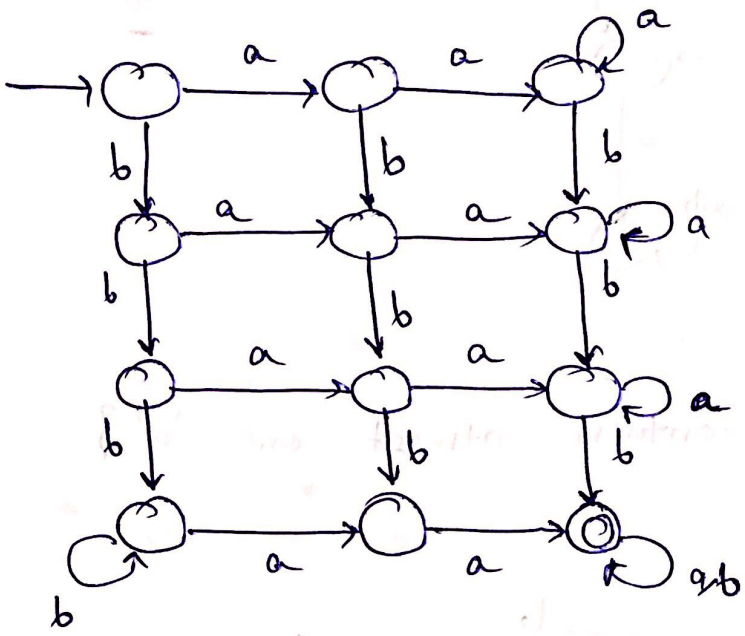
RE = $(b^* + b^* a b^*)$

TYPE-2 (Product automata) :— conditions are applied to both a and b of Σ.

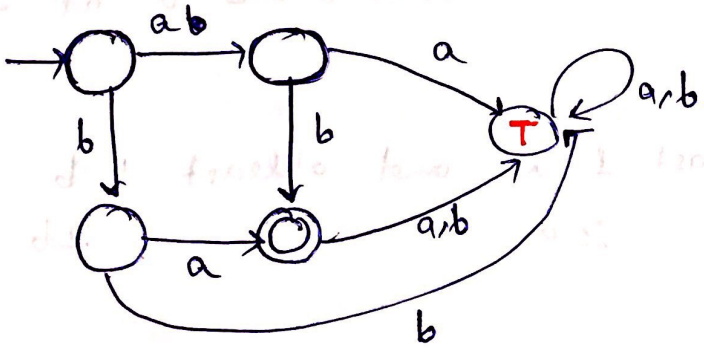
11. L = { atleast 1 a and atleast 1 b
≥ 1 a $\quad\quad$ ≥ 1 b }



L = { atleast 2 a and atleast 3 b }

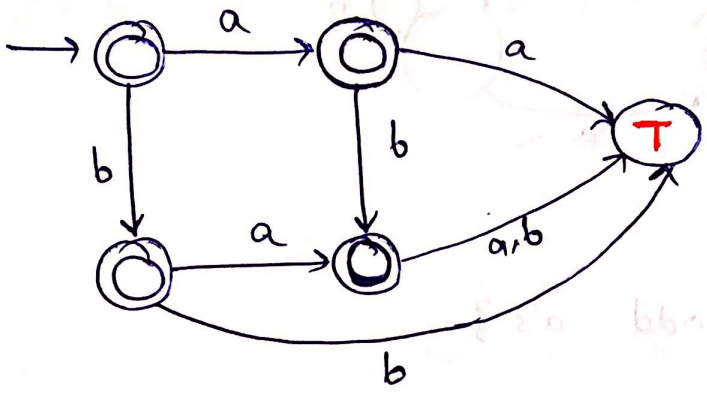


12. L = { 1' a & 1' b }



Remove loops and add Trap state

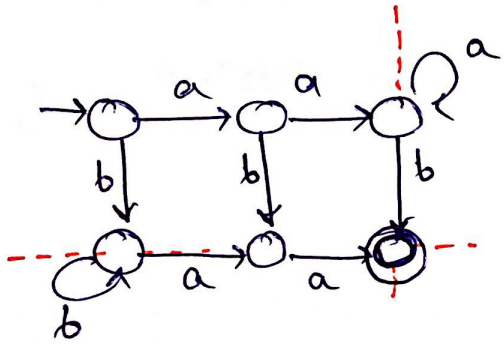
13. L = { atmost 1 a and atmost 1 b }



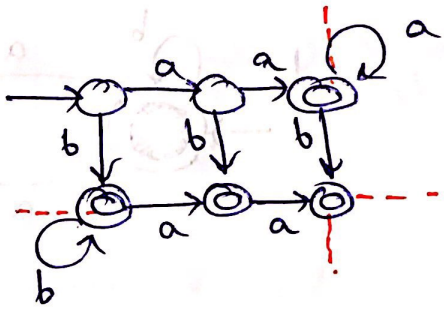
all state becomes final

- (OR & AND) b/w a and b conditions:

$L_1 = \{ \geq 2a \text{ and } \geq 1b \}$; $L_2 = \{ \geq 2a \text{ or } \geq 1b \}$



single final state



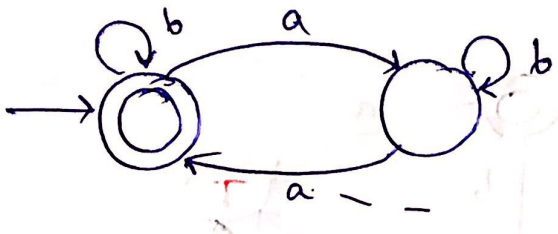
all intersecting states are final

TYPE-3 (Mod Machines):- No (PA) state, (TA) is used

14. $L = \{ \text{Even a's} \}$

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

exact 2a

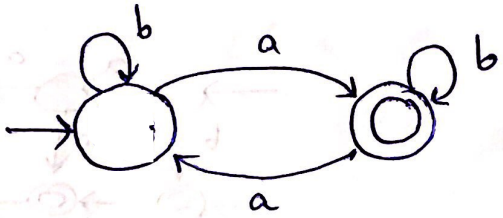


15. $L = \{ \text{odd } a\text{'s} \}$

$$RE = \left(\underbrace{(b^* a b^* a b^*)^*}_{\text{even a's}} + b^* \right) \underbrace{a b^*}_{\text{add 1 a}}$$

OR

$$= b^* a \left((b^* a b^* a b^*)^* + b^* \right)$$



Q. (i) $L = \{ a \bmod 3 = 0 \}$

(ii) $L = \{ a \bmod 3 = 1 \}$

(iii) $L = \{ a \bmod 3 = 2 \}$

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

(ii) $\left((b^* a b^* a b^* a b^*)^* + b^* \right) a b^*$

(iii) $((b^*ab^*ab^*ab^*)^* + b^*)ab^*ab^*$

NOTE:-

$a \bmod k = n \quad (n<k)$

$a \bmod k \geq n$

$a \bmod k \leq n$

no. of states in DFA = k

eg:-

$a \bmod 7 = 5$

states = 7

Final states = 1

$a \bmod 7 \geq 5$

states = 7

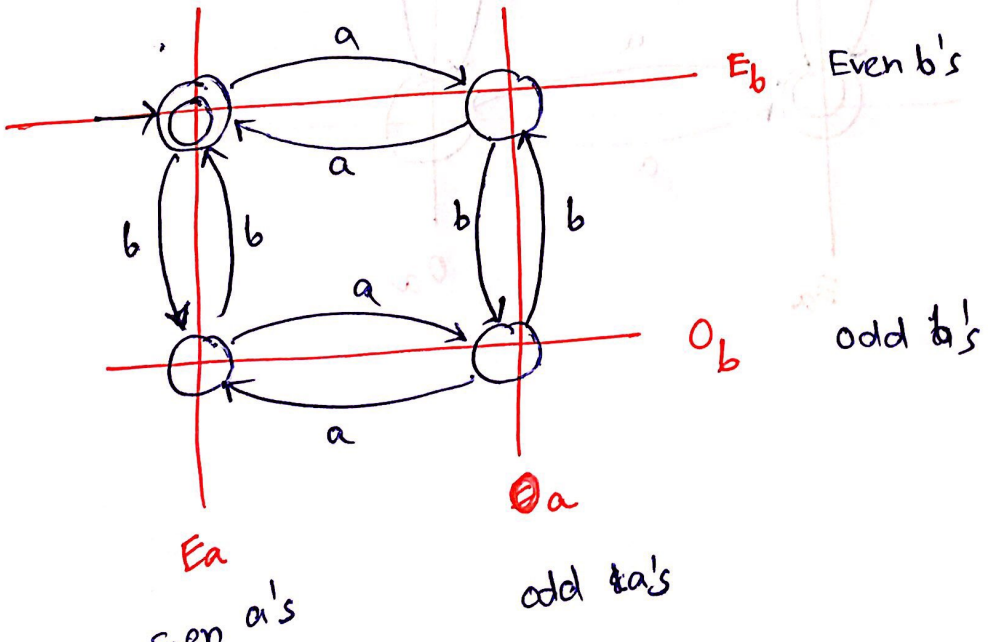
Final states = {5,6}
= 2

$a \bmod 7 \leq 5$

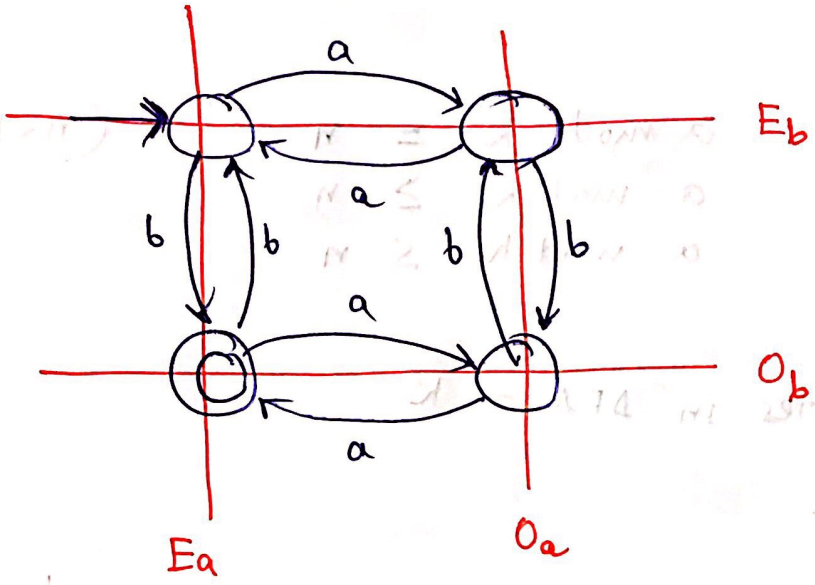
states = 7

Final states = {0-5}
= 6

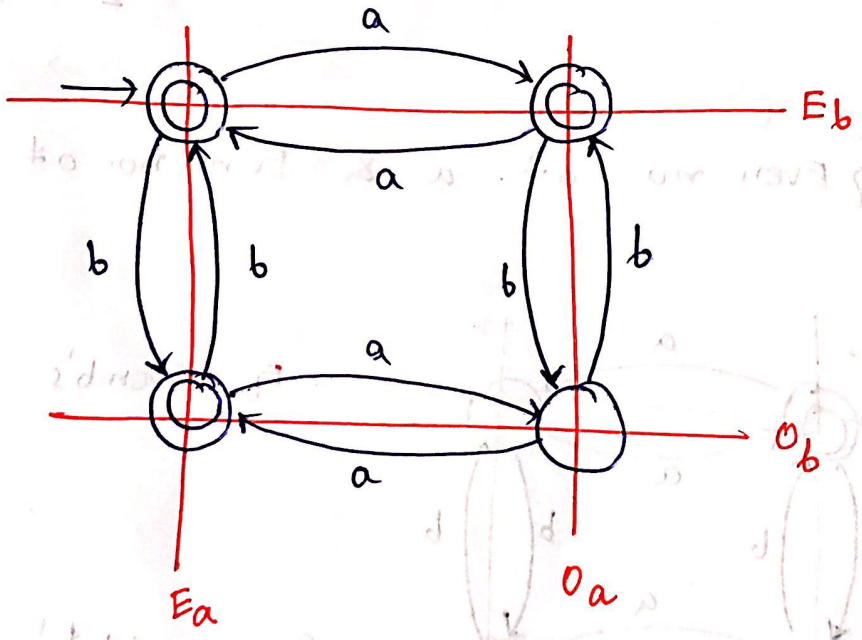
16. L = { Even no. of a & Even no. of b }



L = { even a's & odd b's }

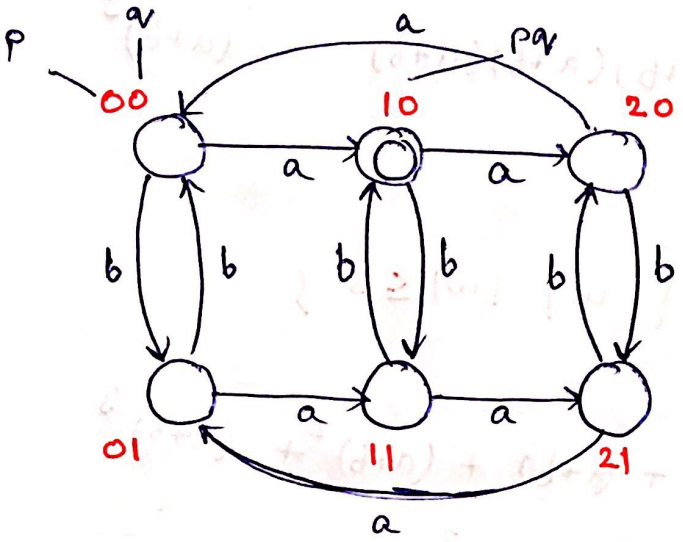


L = { even a's or even b's }



Q. $L = \{ w \mid n_a(w) \bmod 3 = 1 \ \& \ n_b(w) \bmod 2 = 0 \}$

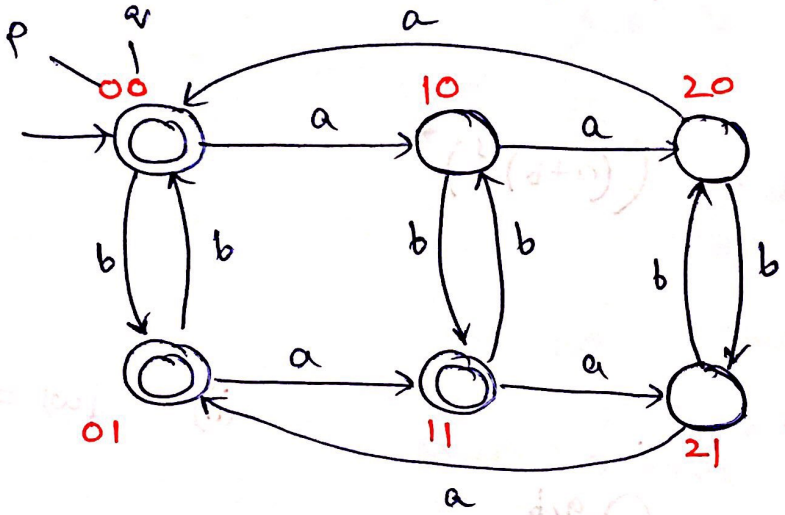
$p = 1$, $q = 0$



$p = \{ \bmod 3 \}$
$= 0, 1, 2$

$q = \{ \bmod 2 \}$
$= 0, 1$

Q. $L = \{ w \mid n_a(w) \bmod 3 \leq n_b(w) \bmod 2 \}$



$p = \{0, 1, 2\}$

$q = \{0, 1\}$

for $p \leq q$

$\{00, 01, 11\}$

Q. (i) $L = \{ w \mid |w| \geq 3 \}$

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

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

(ii) $L = \{ w \mid |w| = 3 \}$

$$RE = (a+b)(a+b)(a+b) = (a+b)^3$$

(iii) $L = \{ w \mid |w| \leq 3 \}$

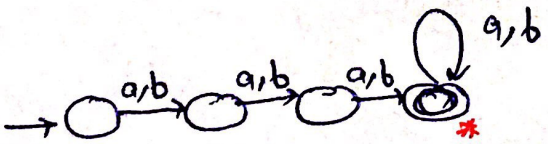
$$RE = \epsilon + (a+b) + (a+b)^2 + (a+b)^3$$

$$= (\epsilon + (a+b))^3$$

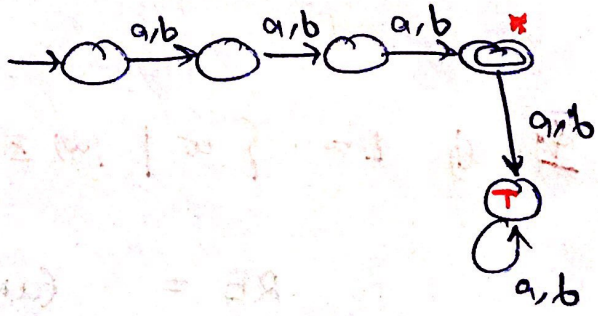
(iv) $L = \{ w \mid |w| \bmod 3 = 0 \}$

$$RE = ((a+b)^3)^*$$

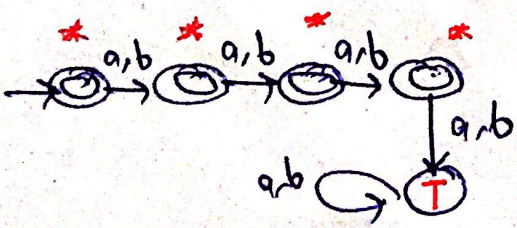
(i) $|w| \geq 3$



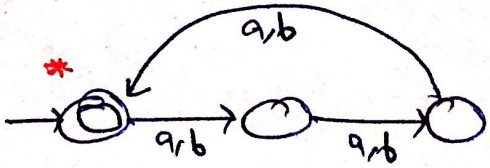
(ii) $|w| = 3$



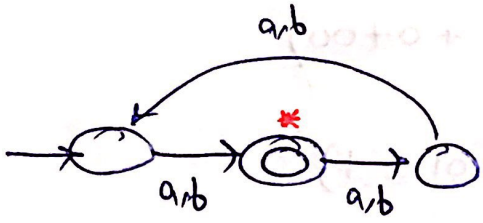
(iii) $|w| \leq 3$



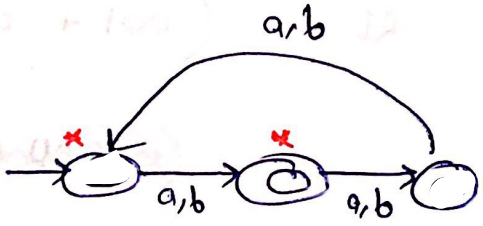
(iv) $|w| \bmod 3 = 0$



(v) $|w| \bmod 3 = 1$



(vi) $|w| \bmod 3 \leq 1$



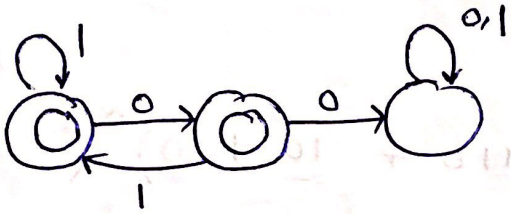
Q. $L = \{$ no two consecutive zeroes $\}$

$$RE = (01+1)^*(\epsilon+0)$$

OR

$$= (\epsilon+0)(01+1)^*$$

Take complimentary M/c for consecutive two 0's

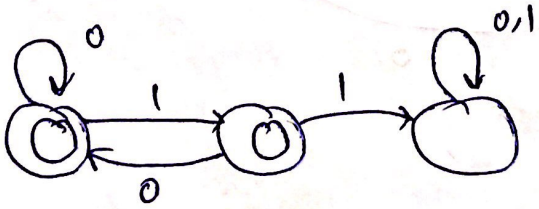


Q. $L = \{$ no two consecutive ones $\}$

$$RE = (\epsilon+1)(0+10)^*$$

OR

$$(0+10)^*(\epsilon+1)$$

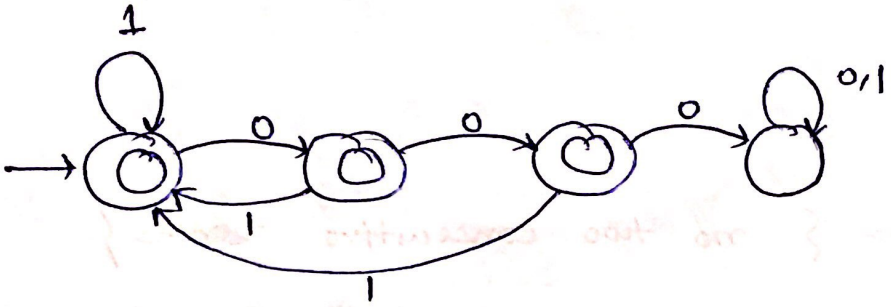


Q. $L = \{$ no three consecutive 0's $\}$

$$RE = (001 + 01 + 1)^*(\epsilon + 0 + 00)$$

OR

$$= (\epsilon + 0 + 00)(001 + 01 + 1)^*$$

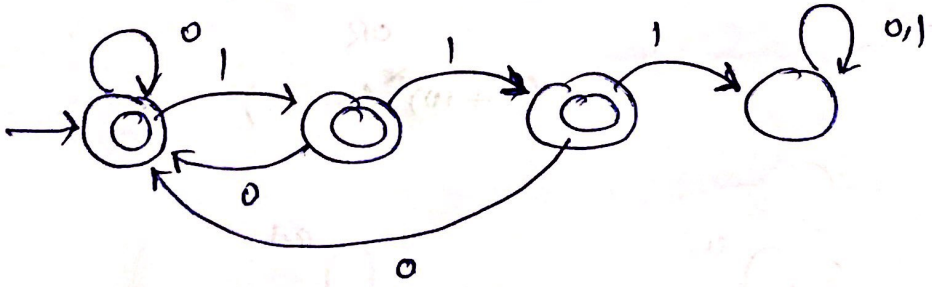


Q. $L = \{$ no three consecutive 1's $\}$

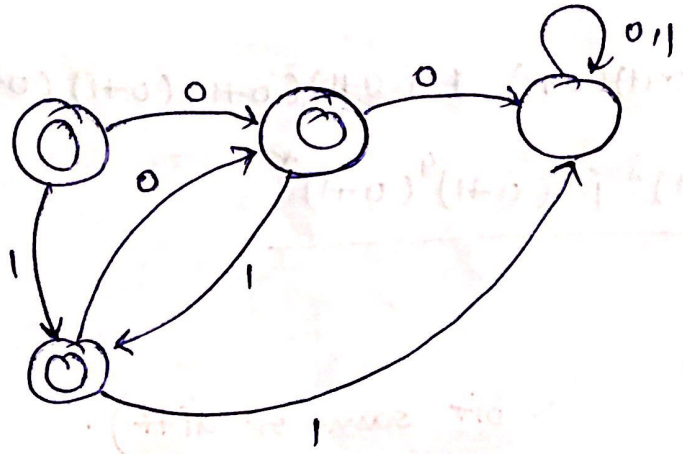
$$RE = (110 + 10 + 0)^*(\epsilon + 1 + 11)$$

OR

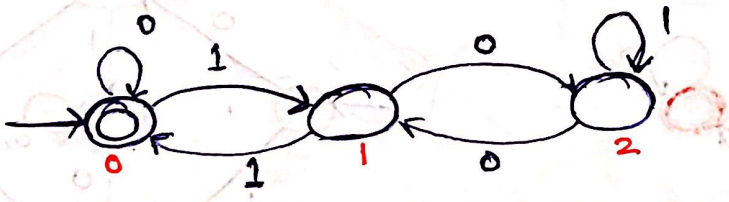
$$= (\epsilon + 1 + 11)(110 + 10 + 0)^*$$



Q. L = { Neither consecutive 2 0's nor 2 consecutive 1's }



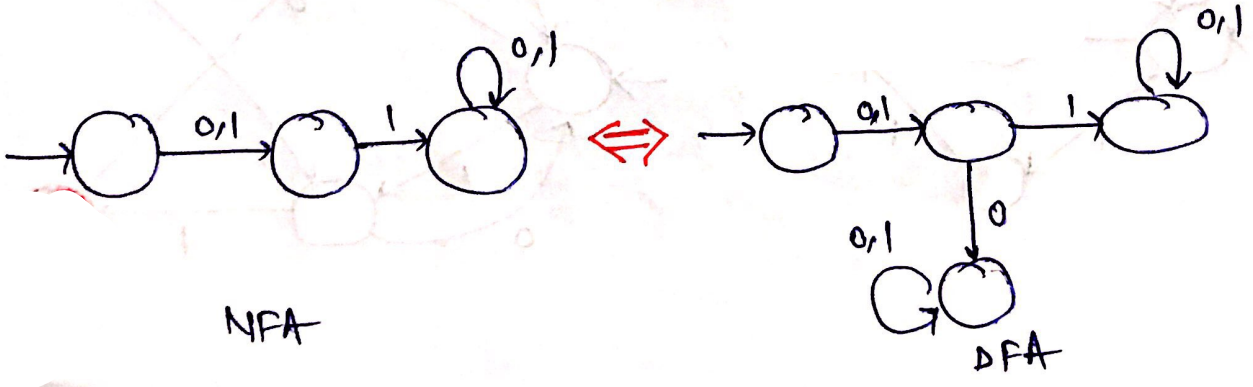
Q. $L = \{ w \mid d(w) \bmod 3 = 0 \}$



TYPE-4 : (Bit setting problem)

Q. L = { second bit is 1 }

RE = $(0+1)\,1\,(0+1)^*$ (Left bit set) DFA possible



Q. $L = \{ \omega_1 1 \omega_2 \mid |\omega_1| = 3 , |\omega_2| \geq 4 \}$

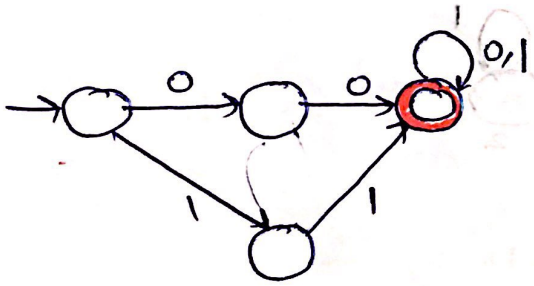
$$RE = (0+1)(0+1)(0+1)\,1\,(0+1)(0+1)(0+1)(0+1)(0+1)^*$$

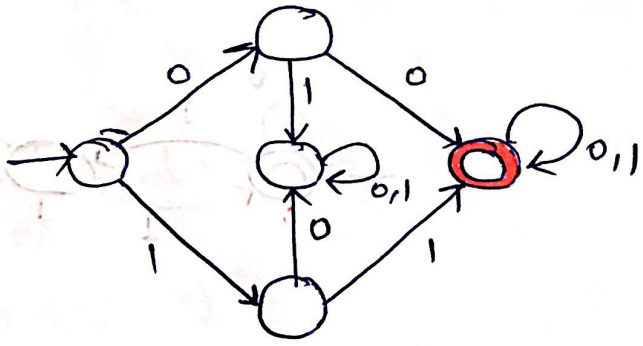
$$= \underline{(0+1)^3\,1\,(0+1)^4(0+1)^*}$$

TYPE -5: (bit same or diff).

Q. $L = \{$ first two symbols are same $\}$

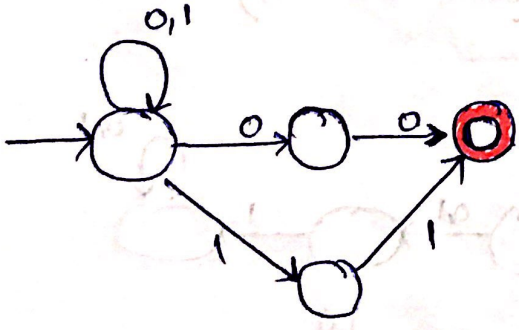
$$RE = (00+11)(0+1)^*$$

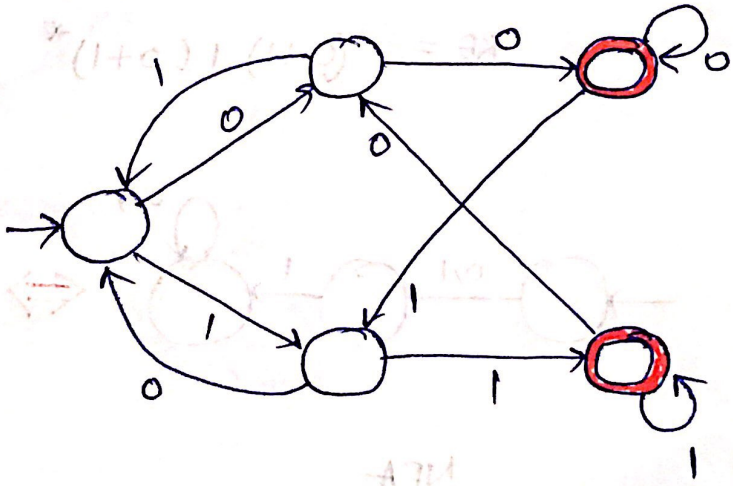




Q. $L = \{$ last two symbols are same $\}$

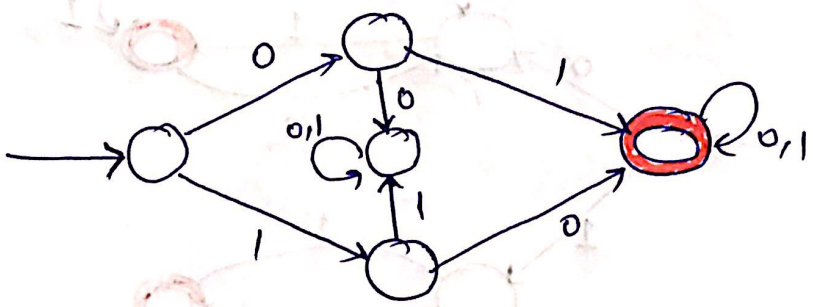
$$RE = (0+1)^*(00+11)$$





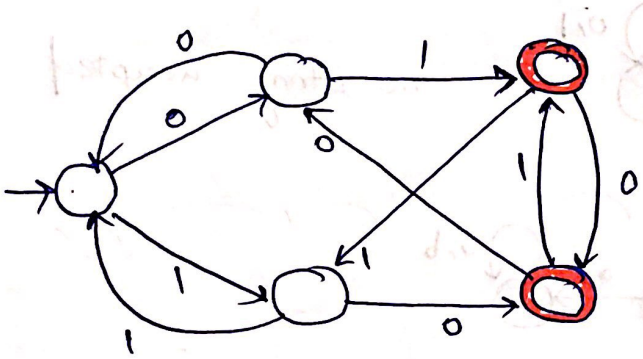
Q. L = { first two symbols are different }

RE = $(01+10)(0+1)^*$



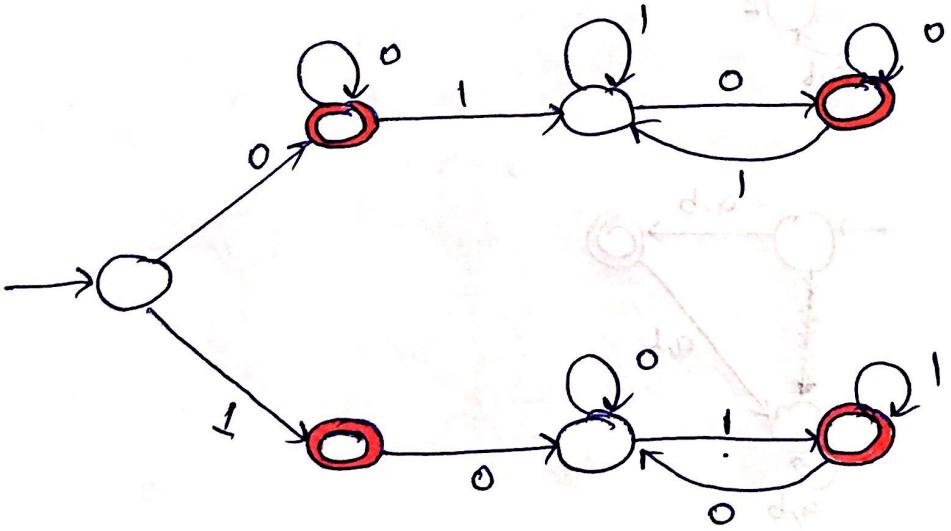
Q. L = { last two symbols are different }

RE = $(0+1)^*(01+10)$



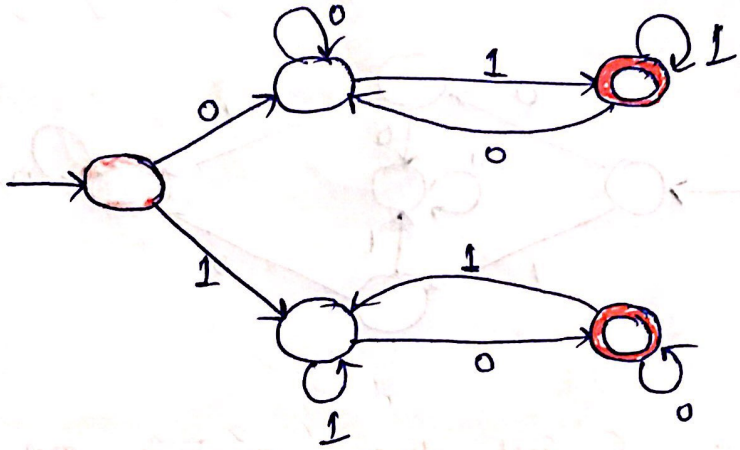
Q. L = { First and last symbols are same }

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



Q. L = { first and last symbol are different }

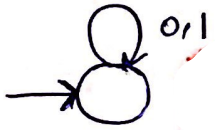
$RE = 0(0+1)^*1 + 1(0+1)^*0$



Design FA from Regular Expression :-

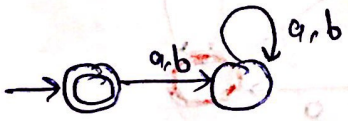
$\Sigma = \{a, b\}$

1. $\emptyset$

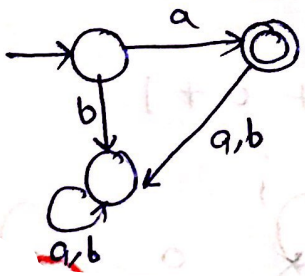


no string accepted

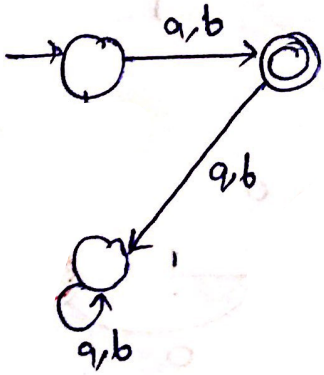
2. ϵ



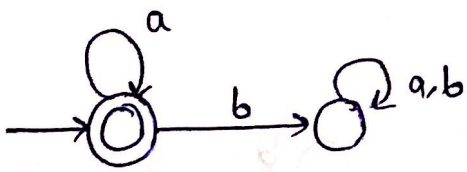
3. a



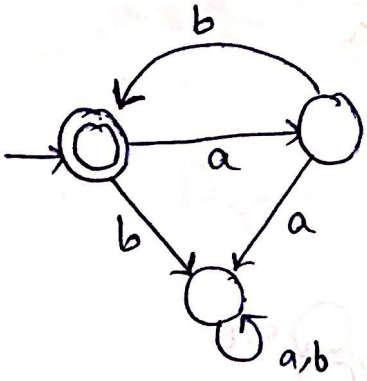
4. a+b



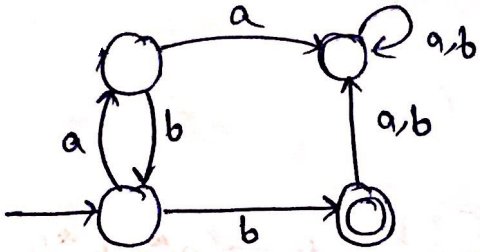
5. a^*



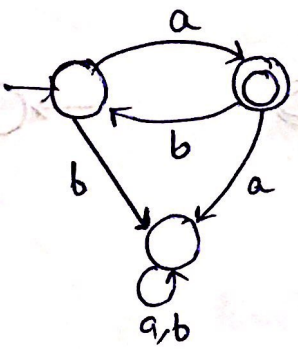
6. $(ab)^*$



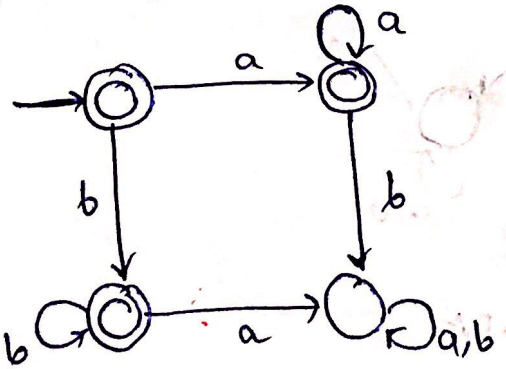
7. $(ab)^*b$



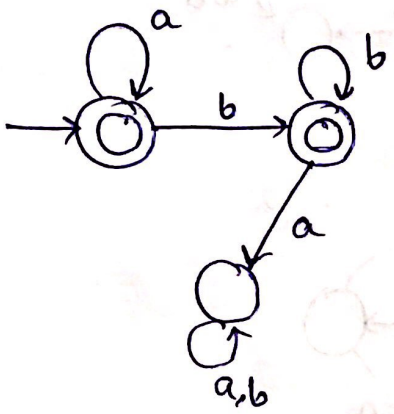
8. $(ab)^*a$



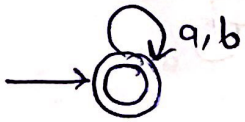
9. $a^* + b^*$



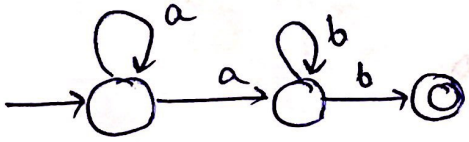
10. $a^* b^*$



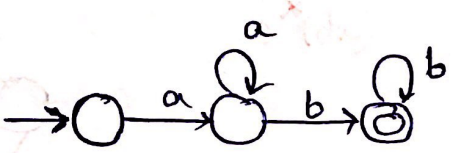
11. $(a+b)^*$



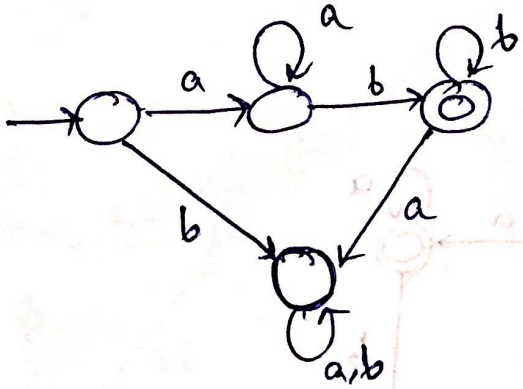
Q. Make DFA for $L = a^* a b^* b$



choice NFA



Dead NFA



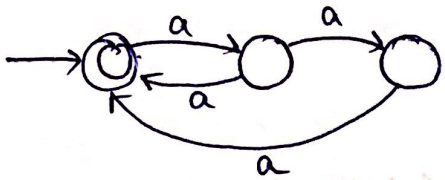
DFA

$\Sigma = \{a\}$

$(aaa)^*a \longrightarrow$ Multiple of 3 + 1 $= \{a^n \mid n = 3x+1\}$

$aaaa^* \longrightarrow a \geq 3 = \{a^n \mid n \geq 3\}$

$(aa + aaa)^* \longrightarrow \{a^n \mid n = 2x + 3y\}$



choice NFA

$(aa)^* + (aaa)^* \longrightarrow \{a^n \mid n = 2x \text{ or } n = 3y\}$

NOTE:-

$a^{k_1 n + k_2}$; k_1 and k_2 are integers

Linear func^n, FA is possible

eg:-

$\{a^{3n+2}\}$; $\longrightarrow$ mod 3 (no. of states), residual = 2

no. of states $= k_1$ $(k_1 > k_2)$

$= k_2 + 1$ $(k_2 > k_1)$

$n \geq 0$

if $n \geq 1$ then

eg $\{ a^{3n+2} \}$

no. of states $= (3(1)+2) + 1$

$= \underline{6}$

if $n \geq 2$ then

$\{ a^{3n+2} \}$

no. of states $= (3(2)+2) + 1$

$= \underline{9}$

4 Aug 2019