* Theory of Computation *

→ Theory of Computation, Also Known as Automata Theory, is a field within Computer science and mathematics that focuses on studying abstract machines to understand the Capabilities and limitation of Computation by Analyzing mathematical models of how machines Can perform Calculations.

1) Symbol :-

→ A symbol (Also Called a character) is the Smallest building block, Which Can be Any Alphabet, letter or Picture.

Exa- A, B, Z, 1, 2, 3,

2) Alphabet :- (Σ)

→ An Alphabet is a finite non empty set of symbols, used to Construct strings and language.

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

$\Sigma = \{A, B, C, \ldots Z\}$

3) String :-

→ It is a finite sequence of symbols Which are the members of Alphabet set.

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

String = aabb, aa, b

) Language:- set of strings

⇒ Using the finite set of Alphabet We Can generate Infinite Words.

↳ So It is Not possible to list them, We have to use Some frameWork, Which Can Somehow represent the Same language.

There are mainly 2 methods to represent a language :

1) By a Grammer that generates a language (RG Generate RL)

2) By a m/c that Accepts a language (FA Accepts RL language)

⇒ $\Sigma = \{a, b\}$

$\Sigma^0 = \{\varepsilon\}$

$\Sigma^1 = \{a, b\}$

$\Sigma^2 = \Sigma \cdot \Sigma = \{a,b\} \cdot \{a,b\}$

$= \{aa, ab, ba, bb\}$

$\Sigma^K = \{w \mid |w| = K\}$

* Kleene Closure :-

→ if Σ is a set of symbols

$$\Sigma^* = \bigcup_{i=0}^{\infty} \{w \mid |w| = i\}$$

$$= \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots\ldots \cup \Sigma^\infty$$

* Positive Closure :-

$$\Sigma^+ = \bigcup_{i=1}^{\infty} \{w \mid |w| = i\}$$

$$= \Sigma^1 \cup \Sigma^2 \cup \dots\dots \cup \Sigma^n$$

* Automaton :-

→ An Automaton is defined as a self-operating system where Energy materials and information are transformed, transmitted and used for performing some functions without direct participation of man.

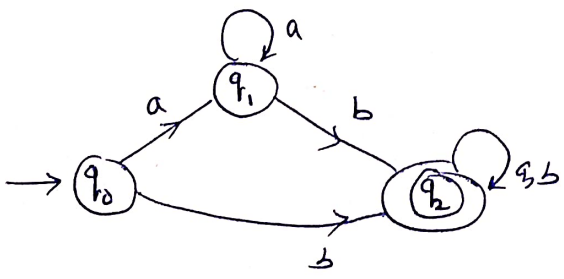
→ The Term is often used to describe a theoretical m/c that operates according to a set of rules and is capable of carrying out complex operations without human interventions.

Exa- Automatic packing m/c

• Finite Automata :-

→ This is a model that has finite set of states.

→ This is widely used in Computer Science for pattern matching, Lexical Analysis, and parsing among other, Applications.



• finite automata without output

1) Deterministic finite Automata
2) Non Deterministic
3) Non Deterministic finite automata with ε

• finite Automata with output

1) moore m/c
2) mealy m/c

1) Deterministic Finite Automata (DFA)

DFA is defined by 5 tuple:

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

Q : finite and Non Empty set of states

Σ : set of finite input Alphabet

δ : Transition fn

$\delta : Q \times \Sigma \to Q$

S : Initial State ($S \in Q$)

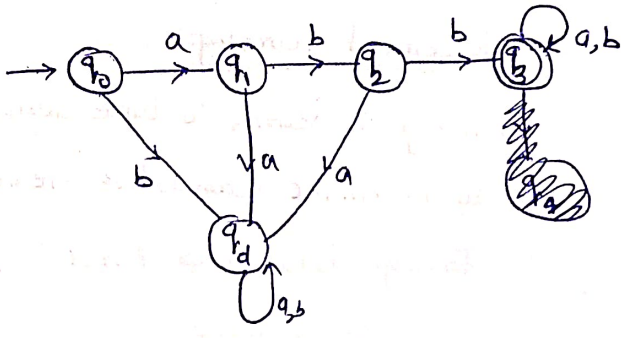
F : set of final states

($F \subseteq Q$)

$0 <= |F| <= N$

Q. Design a DFA, $\Sigma = \{a, b\}$ string starts with 'abb'

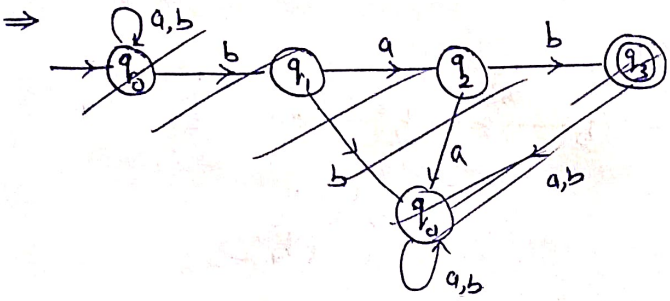
⇒

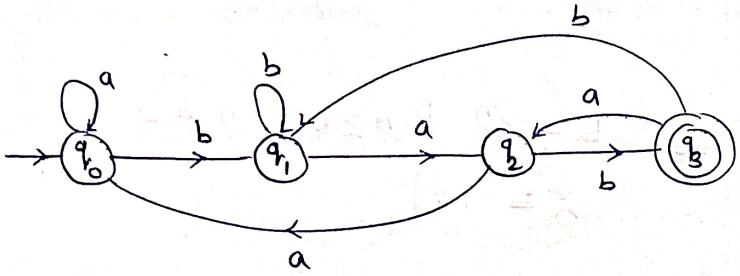


Q $\Sigma = \{a, b\}$

String Ends with bab

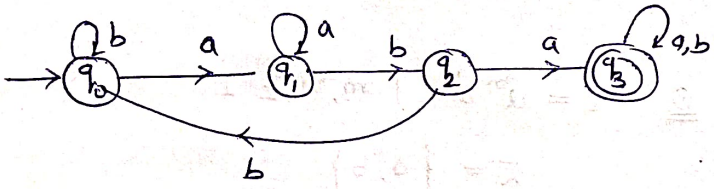
⇒





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

string Contain aba

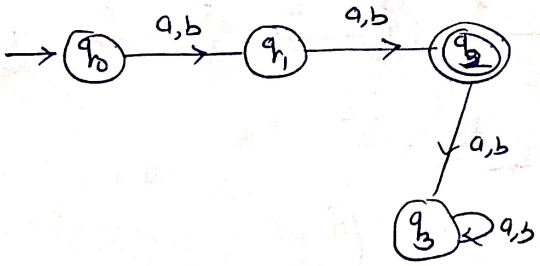


Transition Table

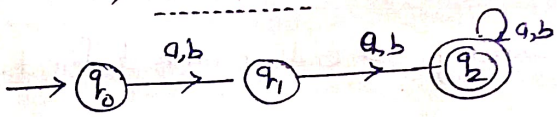
	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_3	q_0
q_3	q_3	q_3

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

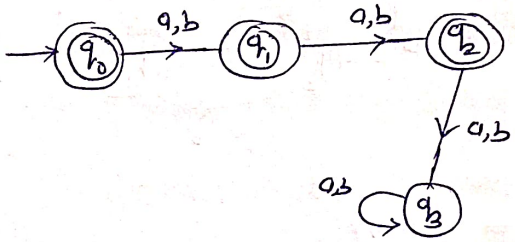
a) $|w| = 2$



b) $|w| \geq 2$



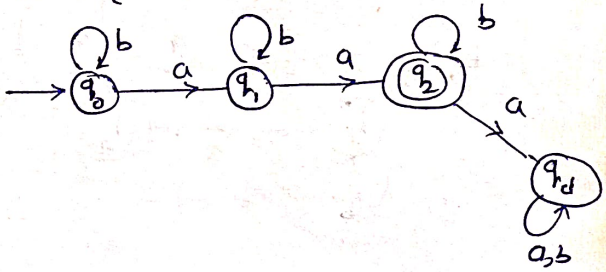
c) $|w| \leq 2$



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

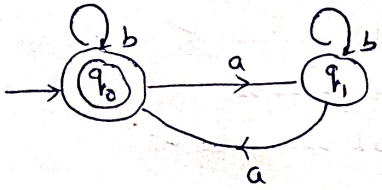
$n_a(w) = 2$

$L = \{$ ~~aa~~ aa, baa, aba, aab ----- $\}$

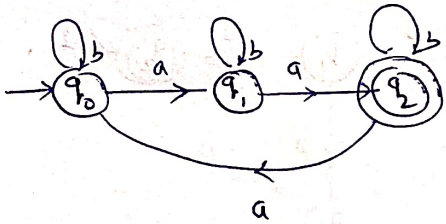


Q $\Sigma = \{a, b\}$

→ $n_a(w) \% 2 = 0$



→ $n_a(w) \bmod 3 = 2$



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

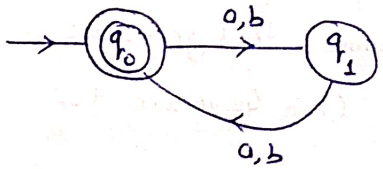
if $|w| \bmod n = x$

$x \in [0, n-1]$

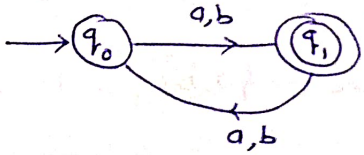
States = n $(q_0 \text{ ..... } q_{n-1})$

final state = q_x

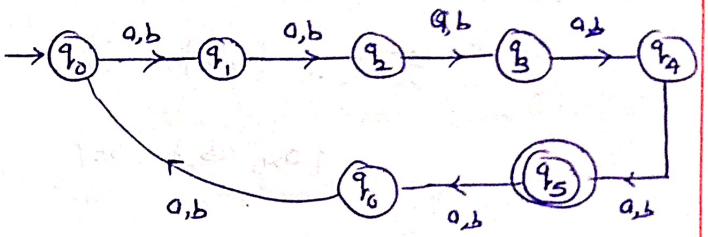
i) $|w| \bmod 2 = 0$



ii) $|w| \bmod 2 = 1$



iii) $|w| \bmod 7 = 5$

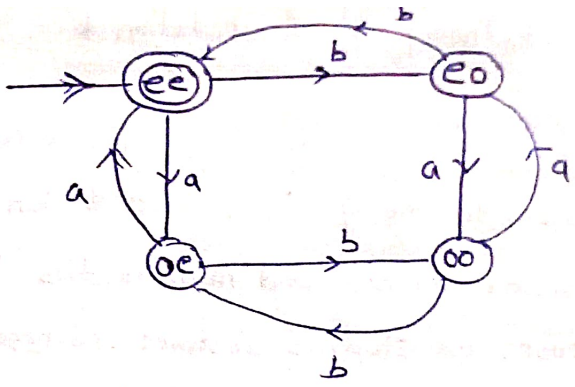


Q $\Sigma = \{a, b\}$

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

e: Even
o: odd

a	b
e	e
e	o
o	e
o	o



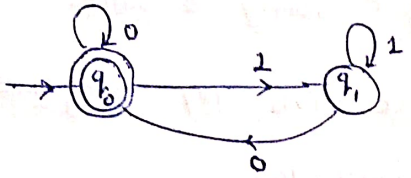
Q $\Sigma = \{a, b\}$

$n_a(w) = 0 \bmod 3$ && $n_b(w) \bmod 2 = 0$

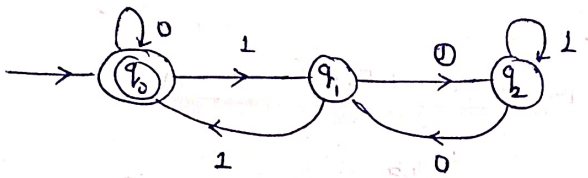
Q. $\Sigma = \{0,1\}$

A) String is interpreted as binary representation of an integer is divisible by 2.

Ans:-



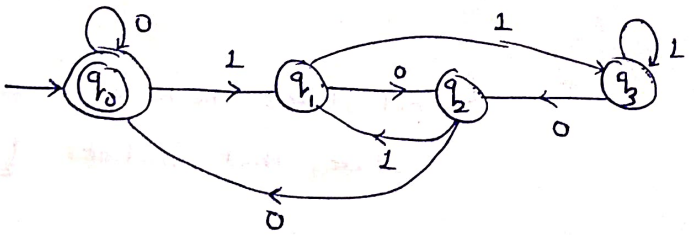
B) binary representation of an Integer is divisible by 3.



	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

c) divisible by 4

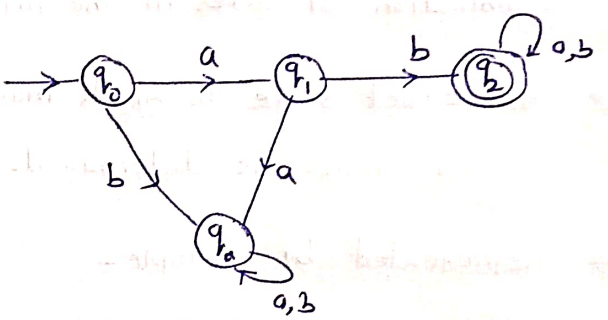
	0	1
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_0	q_1
q_3	q_2	q_3



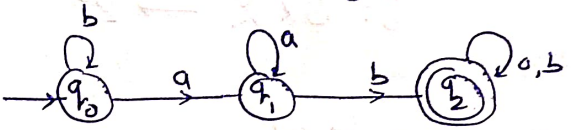
Q. $\Sigma = \{a,b\}$

→ All strings start with ab

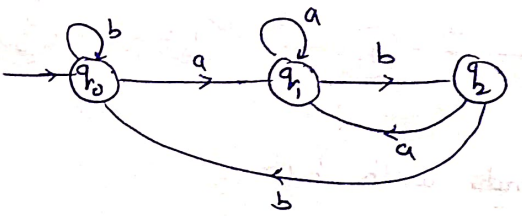
Ans - $L = \{ab, aba, abba, abb \text{----}\}$



→ Contains substring ab



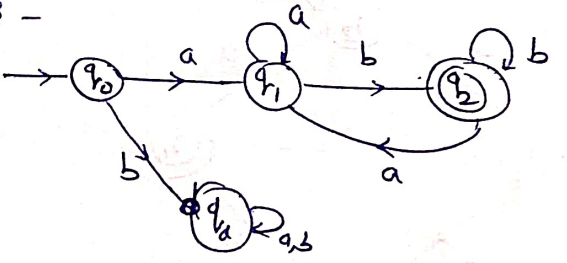
→ End with ab



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

→ Starting with a and Ending with b

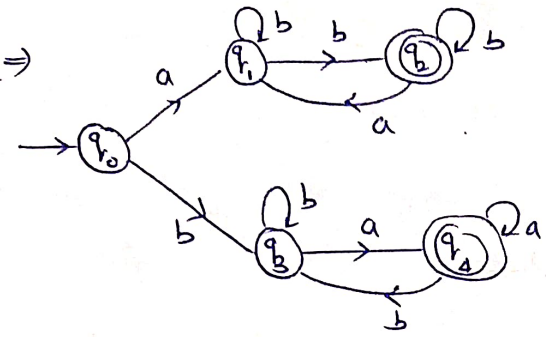
Ans -



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

→ Starting and Ending with different symbol

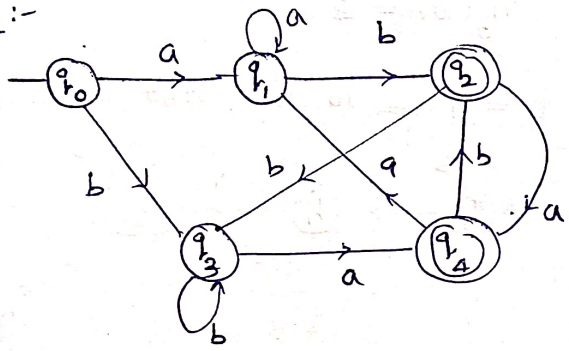
Ans ⇒



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

→ Ending with 'ab' or 'ba'

Ans :-



Q $\Sigma = \{0, 1\}$

→ Starts ~~and~~ or Ends with 01

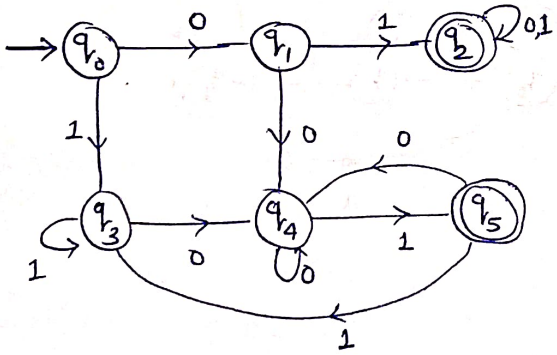
Ans -

01 ----- ✓

------ 01 ✓

01 ------ 01 ✓

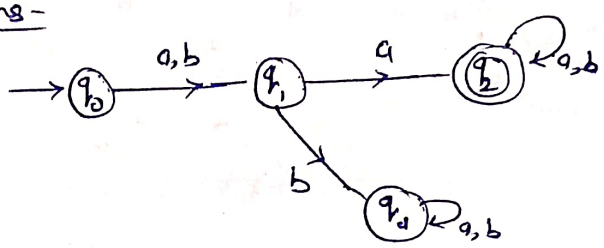
$L = \{01, 001, 010, 0101 \text{ ------}\}$



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

→ Second symbol from LHS is 'a'

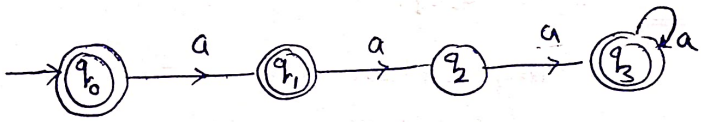
Ans -



Q $L = a^n \mid n \geq 0, n \neq 2$

$\Sigma = \{a\}$

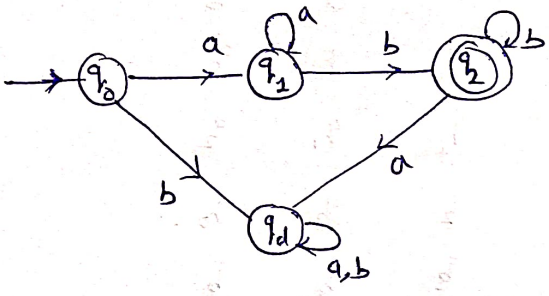
Ans- $L = \{\varepsilon, a, \overset{\times}{aa}, aaa, \text{-----}\}$



Q $L = a^m b^n \mid m, n \geq 1$

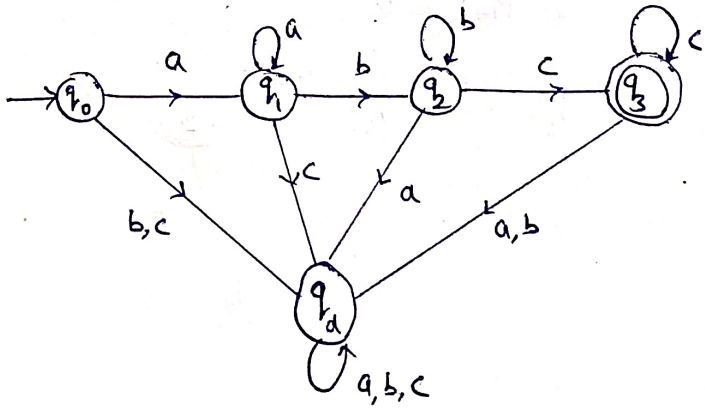
$\Sigma = \{a, b\}$

Ans- $L = \{ab, abb, aab, aaab, \text{-----}\}$



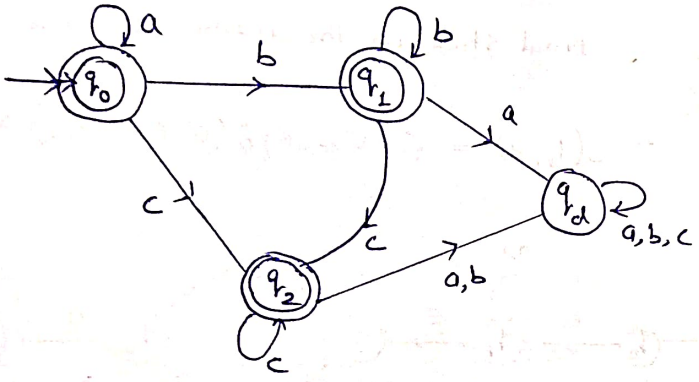
Q $L = a^m b^n c^p, \quad m, n, p \geq 1$

$L = \{a, b, c, aabc, abbc, abcc, \text{-----}\}$



Q $L = a^m b^n c^p$, $\forall$ m, n, P ≥ 0

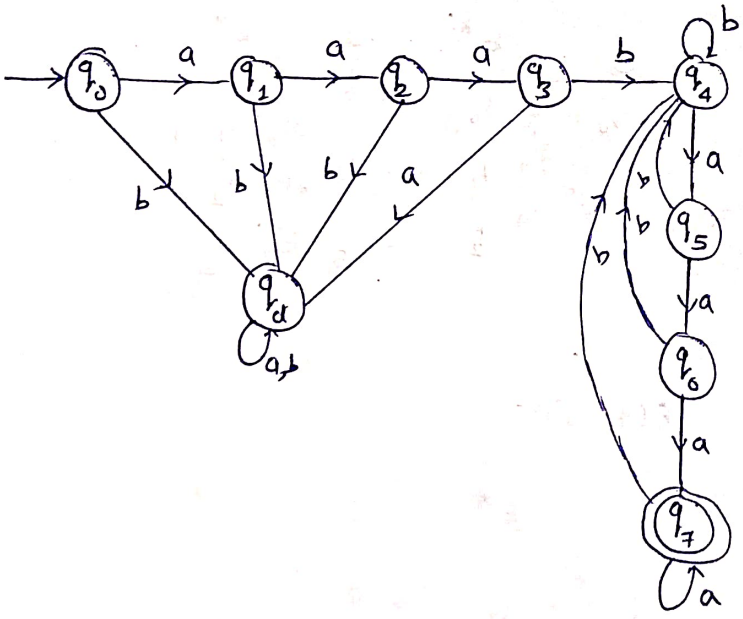
a	b	c	S
0	0	0	$\in$
0	0	z	c^z
0	y	0	b^y
x	0	0	a^x
0	y	z	$b^y c^z$
x	0	z	$a^x c^z$
x	y	0	$a^x b^y$
x	y	z	$a^x b^y c^z$



Q $L = a^3 b \omega a^3$, ~~$\omega \neq (ab)^*$~~

A) $\omega = \Sigma^*$

$L = \{ aaabaaa, aaabaaaa \text{ ----- } \}$

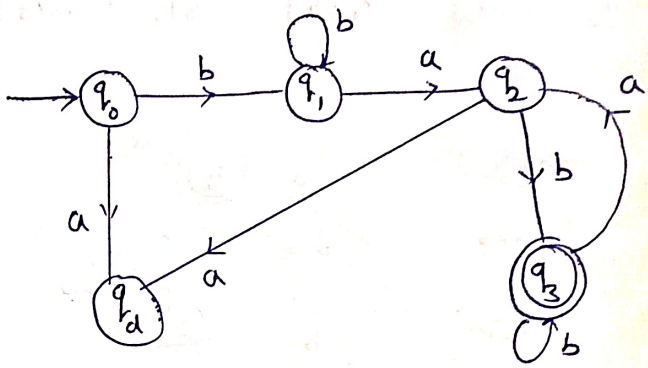


Q $\Sigma = \{a, b\}$

$L = \{ \omega \in \{a,b\}^* \mid$ Every a in ω is immediately and preceded and followed by b $\}$

Ans –

$L = \{ bab, babab \text{ ---- } \}$

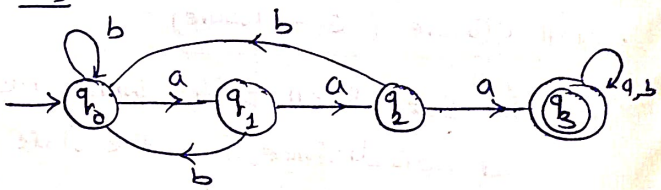


* Compliment of DFA:-

Q – $\Sigma = \{a, b\}$

. string must Not Contain a substring 'aaa'.

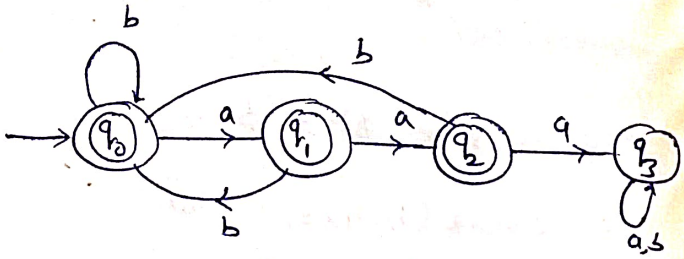
Ans :-



↓ Compliment

make – finale → Non-finale

. NF → finale



* Non-Deterministic finite Automata :

→ In NFA, for a particular input symbol, the m/c can move to any combination of states in the machine.

→ The exact state to which machine moves cannot be determined.

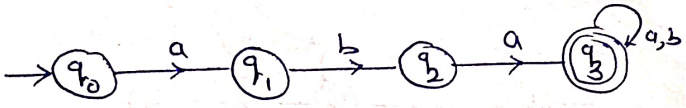
→ Represented by 5 tuple –

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

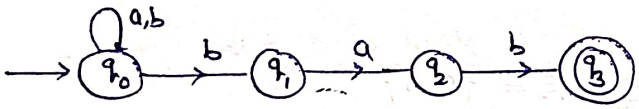
$$\delta : Q \times \Sigma \longrightarrow 2^Q$$

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

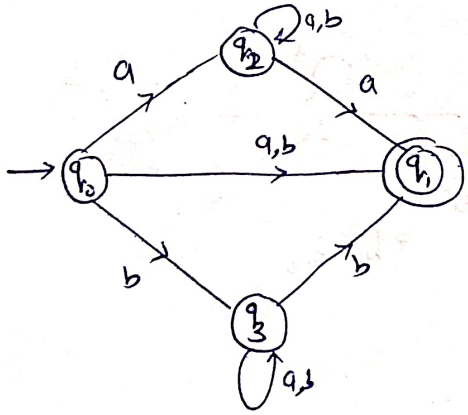
A) ⇒ String starts with 'aba'



B) Ends with bab



c) Start and End with same symbol



→ Every DFA is also an NFA.

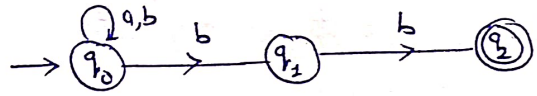
→ Every NFA can be translated to an equivalent DFA, so their language accepting capability is same.

→ NFAs like DFAs only recognize Regular Languages.

→ A Null Transition is also possible for NFA.

* NFA to DFA Conversion

NFA :

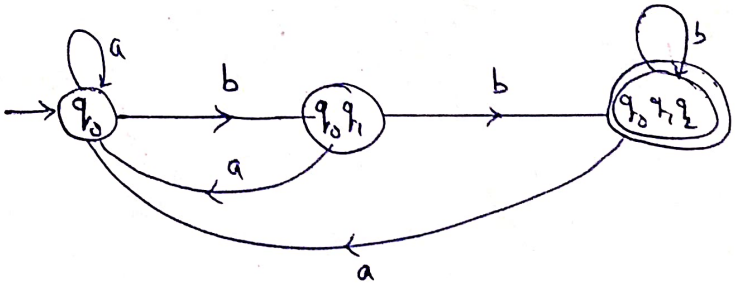


1) Transition Table –

	a	b
→ q_0	q_0	$\{q_0, q_1\}$
q_1	ϕ	q_2
* q_2	ϕ	ϕ

	a	b
→ q_0	q_0	$[q_0, q_1]$
$[q_0 q_1]$	q_0	$[q_0 q_1 q_2]$
* $[q_0 q_1 q_2]$	q_0	$[q_0 q_1 q_2]$

↳ All states should be finale states that includes q_2

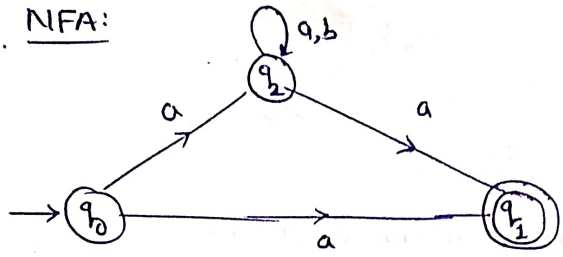


⇒ If NFA have n states, which is Converted into DFA with m states then

$$1 \leq m \leq 2^n$$

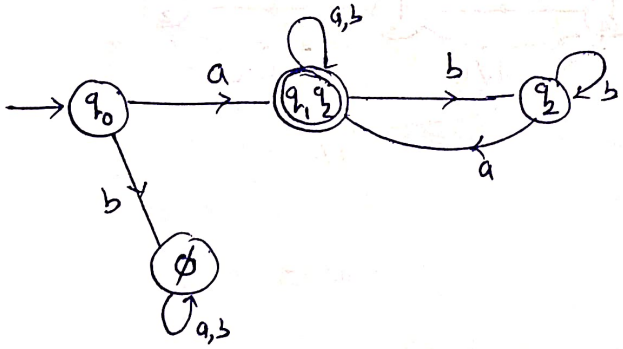
Q

NFA:

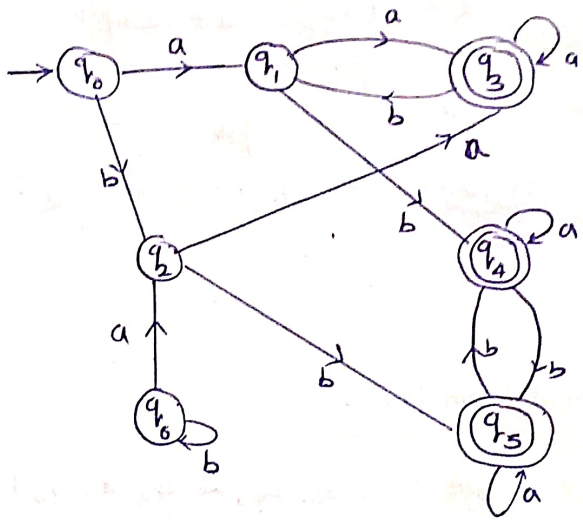


	a	b
→ q_0	$\{q_1, q_2\}$	ϕ
* q_1	ϕ	ϕ
q_2	$\{q_1, q_2\}$	q_2

	a	b
→ q_0	$[q_1 q_2]$	$[\phi]$
* $[q_1 q_2]$	$[q_1 q_2]$	q_2
$[\phi]$	$[\phi]$	$[\phi]$
q_2	$[q_1 q_2]$	q_2



* minimization of DFA:



Step-1 : Check Reachability

if state is not reachable then eliminate that

↳ q_6 will Eliminate

Step-2 : Make Transition Table

	a	b
→ q_0	q_1	q_2
q_1	q_3	q_4
q_2	q_3	q_5
q_3	q_3	q_1
q_4	q_4	q_5
q_5	q_5	q_4

Step-3: Split into classes

1) Accepting (final)

2) Non Accepting (Non final)

then split by K^{th} Equivalent state group

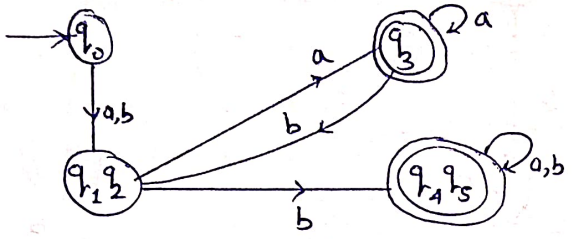
$\pi_0 = \{q_0, q_1, q_2\} \{q_3, q_4, q_5\}$

$\because q_0 \xrightarrow{a} q_1$

$q_1 \quad q_3$

$\pi_1 = \{q_0\}, \{q_1, q_2\}, \{q_3\}, \{q_4, q_5\}$

$\pi_2 = \{q_0\}, \overline{\{q_1, q_2\}}, \{q_3\}, \overline{\{q_4, q_5\}}$



* ∈- NFA :-

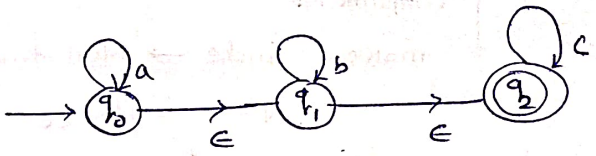
→ An Automata that Consists of Null transitions is Called a ∈-NFA.

→ Represent by 5 tuple- $(Q, \Sigma, \delta, S, F)$

$$\delta: (Q \times \{\Sigma \cup \varepsilon\}) \longrightarrow 2^Q$$

• Null Closure (∈ - Closure) :

⇒ The set of all states which are at zero distance from the state q_i is called ∈ - closure.



$\in$ -closure $(\phi) = \phi$

$\in$ -closure $(q_0, q_1, q_2 \ldots\ldots, q_n)$

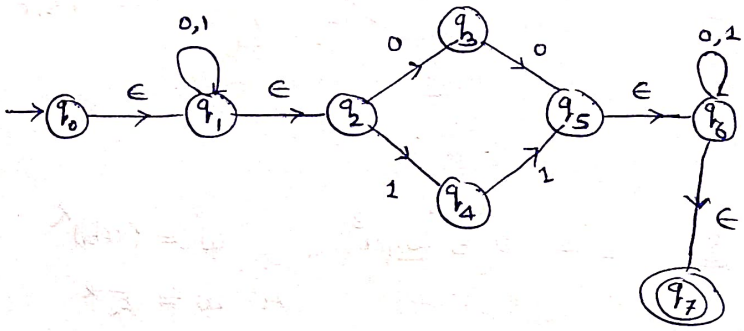
$$= \bigcup_{i=0}^{n} \delta(\in\text{-closure}(q_i))$$

1/18

Equivalence b/w ∈ NFA to NFA

1) There will be no change in the initial state.
2) No change in total No. of states
3) may be change in the Number of final states.
4) All the states will get the status of the final state in the resulting NFA whose ∈ Closure Contains at least 1 final state in the initial ∈-NFA

$\because \delta(q_0, a) = \in\text{-Closure}[\delta(\in\text{-closure}(q_0), a)]$



$\delta(q_0, 0)$:

ε^* 0 ε^*

$q_0 \xrightarrow{\varepsilon^0} q_0 \longrightarrow \phi$

$q_0 \xrightarrow{\varepsilon^1} q_1 \longrightarrow q_1 \xrightarrow{\varepsilon^0} q_1$, $q_1 \xrightarrow{\varepsilon^1} q_2$

$q_0 \xrightarrow{\varepsilon^2} q_2 \longrightarrow q_3 \xrightarrow{\varepsilon^0} q_3$

$\delta(q_0, 1)$:

ε^* 1 ε^*

$q_0 \xrightarrow{\varepsilon^0} q_0 \longrightarrow \phi$

$q_0 \xrightarrow{\varepsilon^1} q_1 \longrightarrow q_1 \xrightarrow{\varepsilon^0} q_1$, $q_1 \xrightarrow{\varepsilon^1} q_2$

$q_0 \xrightarrow{\varepsilon^2} q_2 \longrightarrow q_4 \xrightarrow{\varepsilon^0} q_4$

	0	1
$\rightarrow q_0$	$q_1 q_2 q_3$	$q_1 q_2 q_4$
q_1	$q_1 q_2 q_3$	$q_1 q_2 q_4$
q_2	q_3	q_4
q_3	$q_5 q_6 q_7$	ϕ
q_4	ϕ	$q_5 q_6 q_7$
$*q_5$	$q_6 q_7$	$q_6 q_7$
$*q_6$	$q_6 q_7$	$q_6 q_7$
$*q_7$	ϕ	ϕ

* Moore and mealy Machine :-

- Both moore and mealy m/c are special Case of DFA.
- Both acts like o/p Producers rather than language Acceptors.
- No Need to define final states.
- No Concepts of dead states.
- Moore and mealy m/c are equivalent in Power.

* Moore Machine :

six - tuple : $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

Q: finite set of states

Σ: input Alphabet

Δ : output Alphabet

δ : Transition fn , $\delta : Q \times \Sigma \rightarrow Q$

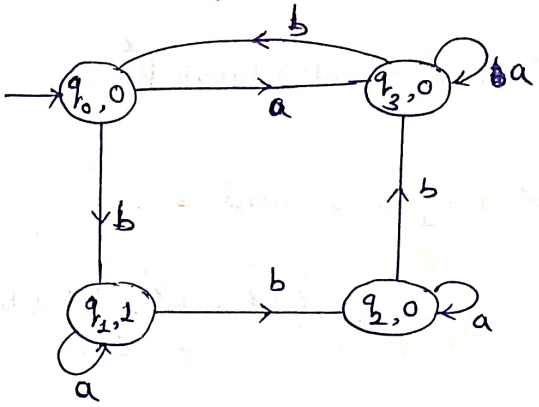
λ : output fn mapping Q into Δ

q_0 : initial state

Exq

Transition Table -

	a	b	λ
$\rightarrow q_0$	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0



Input Str - a b b b a

	q_0	a	b	b	b	a
	↓	↓	↓	↓	↓	↓
o/p		0	0	0	1	0	0

- Input str length = n

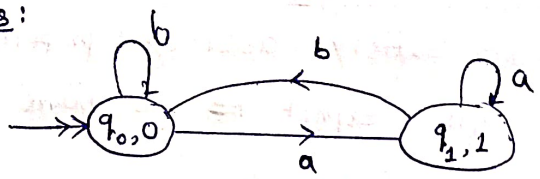
  output str length = n+1

- Also Respond with ϵ

Q Construct a moore m/c , $\Sigma = \{a, b\}$

Count the No. of a's in p i/p string in terms of 1 , $\Delta = \{0, 1\}$

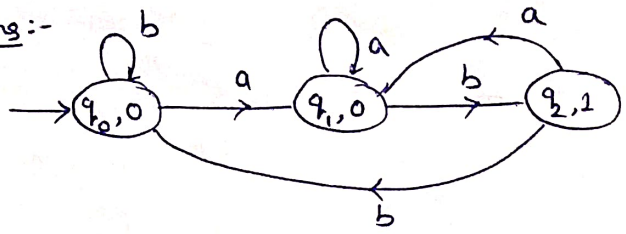
Ans :



→ Q: Count the occurence of ab

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

Ans:-



* Mealy machine :-

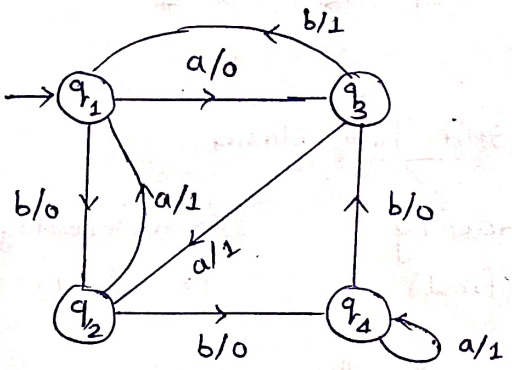
Six-tuple : $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$

$\lambda : Q \times \Sigma \longrightarrow \Delta$

- Length of Input = Length of o/p
- do not Respond with ϵ

Eg

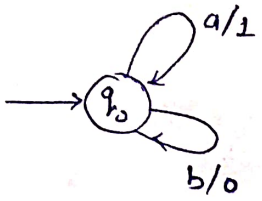
	a		b	
	State	o/p	State	o/p
→ q_{01}	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0



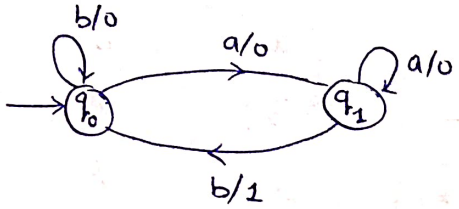
Q $\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$

Count No. of a's.

Ans- mealy m/c -

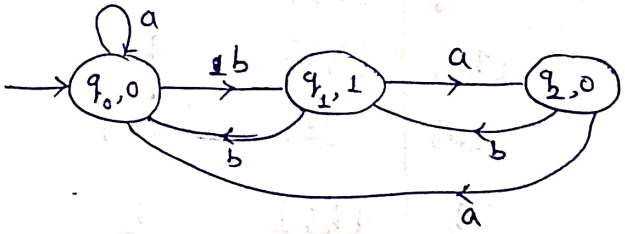


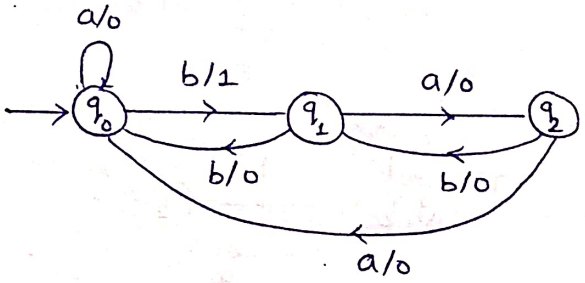
Count occurance of ab



⇒ Conversion of moore to mealy

only attact destination output for mealy.





⇒ Conversion of mealy to moore

* Regular Expressions :-

Let R be a Regular Expression over Alphabet Σ if R is :

1) $\in$, is a RE denoting the set $\{\in\}$
2) ϕ is a RE denoting Empty set $\{\}$
3) for Each symbol $a \in \Sigma$, a is RE denoting set $\{a\}$
4) Union of two RE is also RE.
5) Concatenation is also RE
6) Kleene closure of RE is also Regular.
7) If R is Regular, (R) is also Regular.
8) Nothing Else, Repeat 1 to 7 Recursively.

Primitive RE — $\{\in\}, \{\}, \{a\}$

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

RE for Length 1 : $(a+b)$

RE for Length 2 : $(aa+ab+ba+bb)$

$= (a+b)^2$

• Equivalent Regular Expressions :-

→ Two Regular Expressions P and Q are equivalent if P and Q Represent the Same set of string.

Exa - $r_1 = a^*$

$r_2 = a^* + (aa)^*$

Q design a Regular Expression

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

A) String W starts With substring 'abb'

Ans - $R = abb(a+b)^*$

B) String W Contains 'aba'

Ans- $R = (a+b)^* aba (a+b)^*$

c) Starting and Ending with a

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

d) Starting and Ending with Same Symbol

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

e) W is like- $w = sx$

$s = aaa / bbb$

$\Rightarrow (aaa+bbb)(a+b)^*$

f) $|W| = 3$

$R = (a+b)^3$

$|W| \geq 3$

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

$|W| \leq 3$

$R = \in + (a+b) + (a+b)^2 + (a+b)^3$

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

g) 2nd symbol from left end is always b.

$\Rightarrow$ $R = (a+b)\,b\,(a+b)^*$

h) ✓ $|w| = 0 \mod 3$

$R = [(a+b)^3]^*$

✓ $|w| = 3 \mod 4$

$R = (a+b)^3 [(a+b)^4]^*$

✓ $|w|_a = 0 \mod 3$

$R = b^* (b^* a\, b^* a\, b^* a\, b^*)^*$

✓ $|w|_b = 2 \mod 3$

$R = a^* b\, a^* b\, a^* (a^* b\, a^* b\, a^* b a^*)^*$

- Algebic properties of RE :-

1) Closure property :-

if R_1 and R_2 are RE, then the following ~~or~~ will also be RE

$R_1 + R_2$, $R_1 \cdot R_2$

R_1^* , R_1^+

2) Associative property :-

→ RE satisfy associative property with respect ~~to~~ to union and Intersection.

$r_1 + (r_2 + r_3) = (r_1 + r_2) + r_3$

$(r_1 \cdot r_2) \cdot r_3 = r_1 \cdot (r_2 \cdot r_3)$

3) Identity Property :-

$r \cdot \in = r$

$r + \phi = r$

4) Commutative Property :-

↳ Commutative w.r.t union not with Concatenation

$r_1 + r_2 = r_2 + r_1$

$r_1 \cdot r_2 \neq r_2 \cdot r_1$

5) Distributive property :-

$r_1 (r_2 + r_3) = r_1 r_2 + r_1 r_3$

$(r_1 + r_2) r_3 = r_1 r_3 + r_2 r_3$

$r_1 + (r_2 \cdot r_3) \neq (r_1 + r_2)(r_1 + r_3)$

$(r_1 \cdot r_2) + r_3 \neq (r_1 + r_3) \cdot (r_2 + r_3)$

6) Idempotent property

$r_1 + r_1 = r_1$

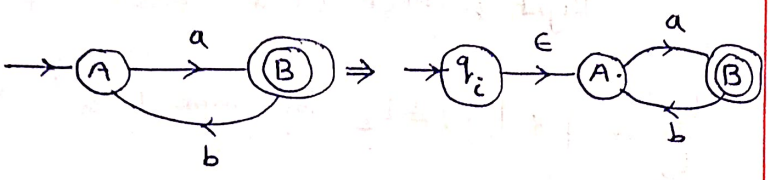
$r_1 \cdot r_1 = r_1$

*

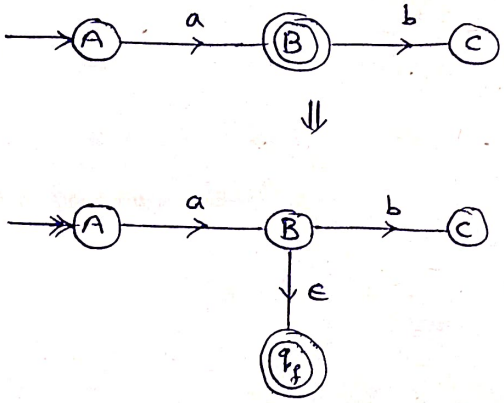
* Conversion from FA to Regular Expression:

1) State Elimination method:

A) If the Initial state is either an accepting state or has incoming transitions, introduce a new non-Accepting ~~start~~ Start state, then add an $\in$ transition from the new start state to original start state.

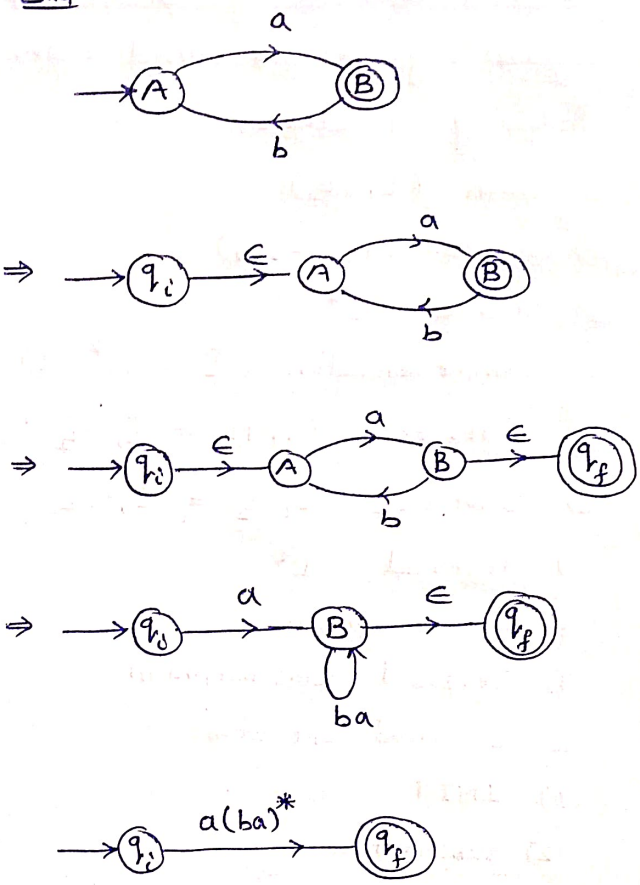


B) If there are outgoing Transition from Accepting state then create a new accepting state, change all Other Accepting states to Non-Accepting state, Add $\in$ transition.



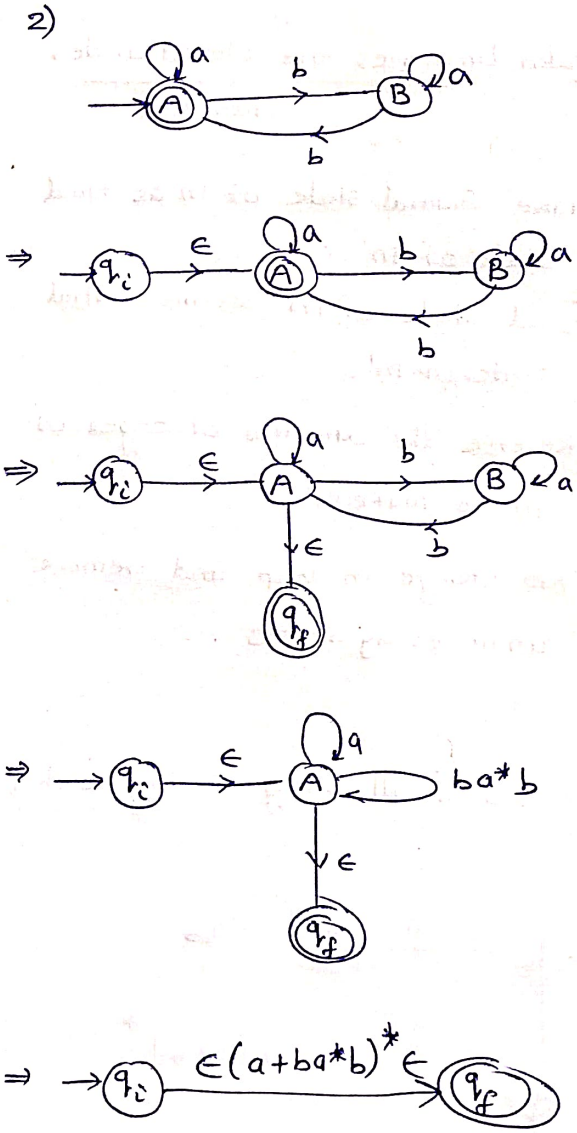
C) Eliminate all states one by one except Initial and final state.

Eg –

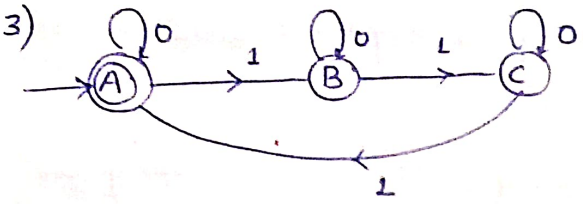


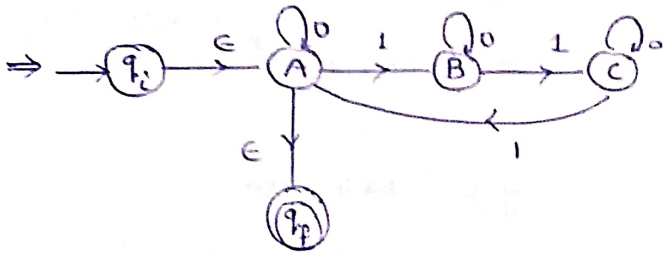
---

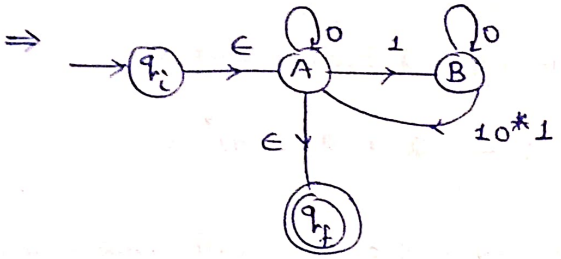
2)

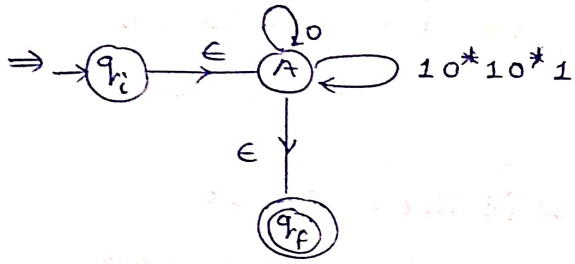


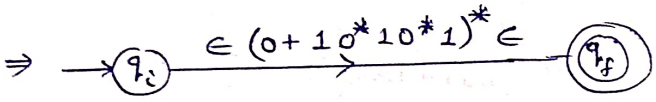
3)


0
1
0
1
0
A
B
C
1


⇒
∈
0
1
0
1
0
A
B
C
1
∈
q_i
q_f

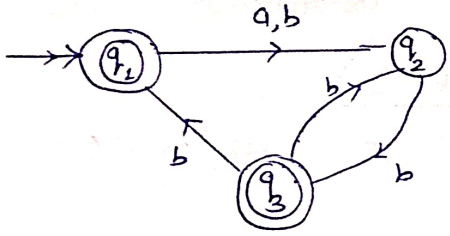

⇒
∈
0
1
0
A
B
10*1
∈
q_i
q_f

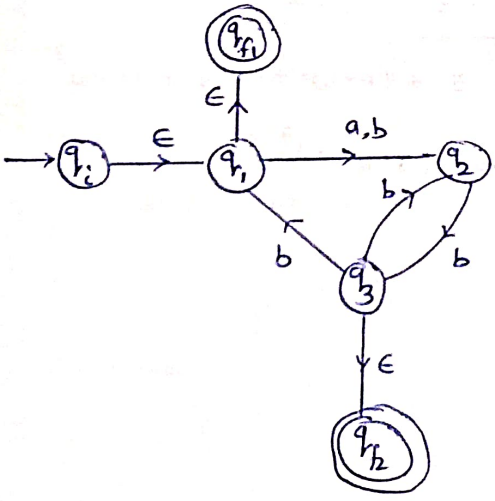

⇒
∈
0
A
10*10*1
∈
q_i
q_f

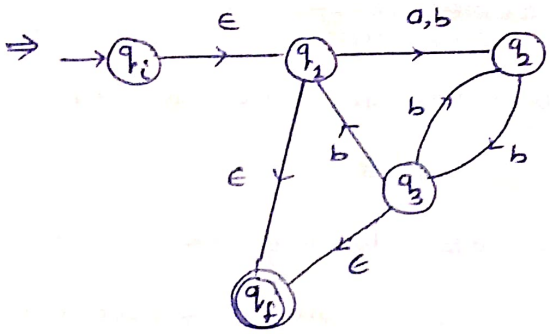

⇒
∈ (0+10*10*1)* ∈
q_i
q_f

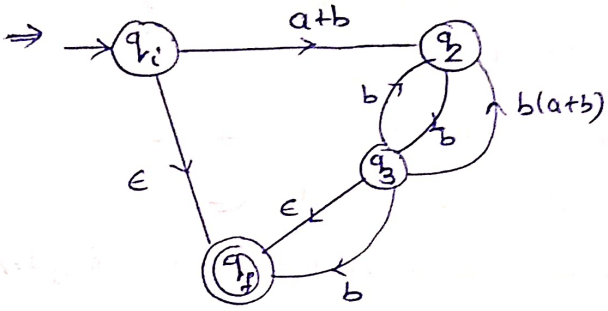
RE = $(0 + 10^*10^*1)^*$

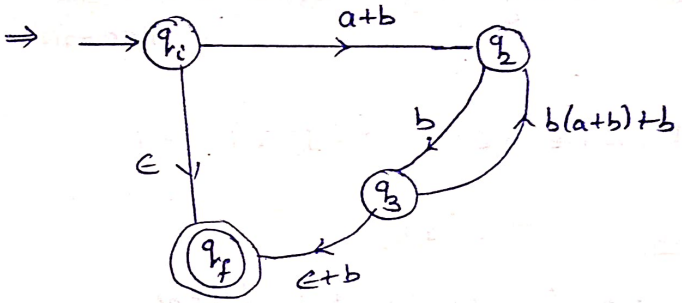
4)

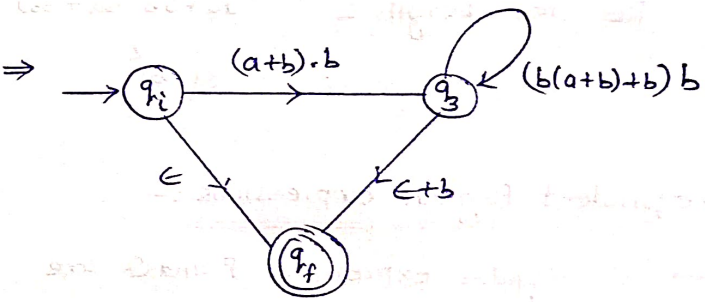

a,b
b
b
b
q_1
q_2
q_3

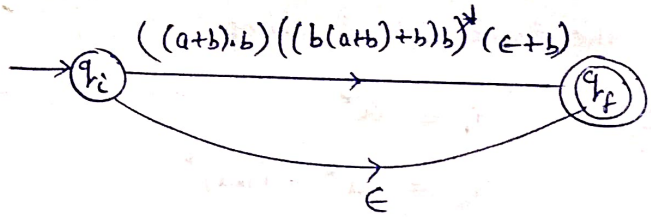

q_{f1}
∈
∈
a,b
b
b
b
∈
q_i
q_1
q_2
q_3
q_{f2}


⇒
∈
a,b
b
b
b
∈
∈
q_i
q_1
q_2
q_3
q_f


⇒
a+b
b
b
b(a+b)
∈
∈
b
q_i
q_2
q_3
q_f


⇒
a+b
b
b(a+b)+b
∈
∈+b
q_i
q_2
q_3
q_f

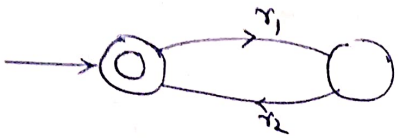

⇒
(a+b).b
(b(a+b)+b)b
∈
∈+b
q_i
q_3
q_f


((a+b).b)((b(a+b)+b)b)*(∈+b)
∈
q_i
q_f

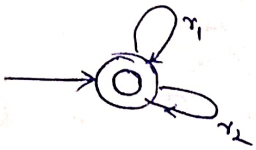
* Conversion of Regular Expression to FA

1) R^*

2) $(R_1 \cdot R_2)^*$

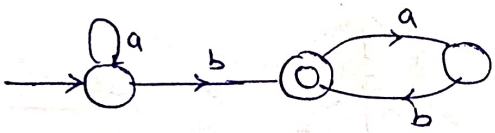


3) $(R_1 + R_2)^*$

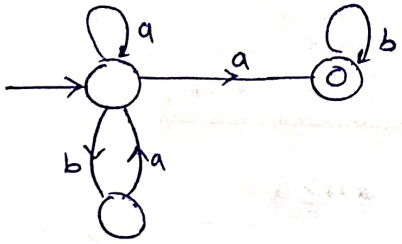


4) $R_1^* R_2 R_3^*$

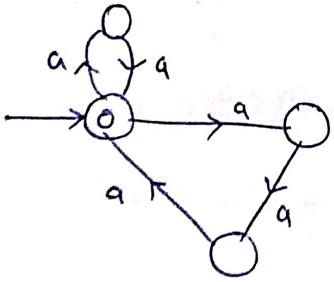
5) $a^* b (ab)^*$



6) $(a+ba)^* ab^*$



7) $(aa + aaa)^*$



* Pumping Lemma :-

→ method to check whether a language is regular or not.

→ If L is an Infinite language then there exists some positive integers 'n' (pumping length) such that any string $w \in L$ has length greater than and equal to n i.e. $|w| \geq n$ then w can be divided into three parts, $w = xyz$ satisfy following Condⁿ.

i) for each $i \geq 0$, $xy^iz \in L$

ii) $|y| > 0$ and

iii) $|xy| \leq n$

Exa :- $a^n b^{2n}$, $n \geq 0$

$w = aabbbb \in L$

$\underset{x}{aa}\ \underset{y}{bb}\ \underset{z}{bb}$

xy^2z : $\underset{x}{aa}\ \underset{y^2}{bbbb}\ \underset{z}{bb} \notin L$

⇒ Ir-regular Language

* Closure properties of Regular Language

→ Regular language are closed under following operation –

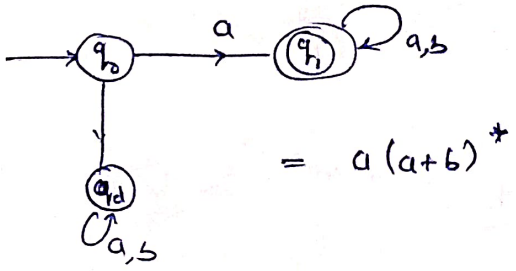
1) Union ($L_1 \cup L_2$)
2) Concatenation ($L_1 \cdot L_2$)
3) Closure (L^*)
4) Complementation ($\overline{L} = \Sigma^* - L$)
5) Intersection ($L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$)
6) Difference ($L_1 - L_2 = (L_1 \cap \overline{L_2})$)
7) Reversal L^R
8) Homomorphism
9) Reversal Homomorphism
10) Quotient operation
11) INIT
12) Substitution
13) Infinite union X (Not Close in this)

- Regular Languages are closed under Reversal

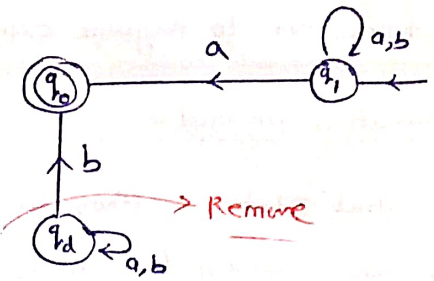
1) make Initial state of M as final state of M'
2) final state of M become Initial state of M'.
3) Reverse the direction of Edges of M to make M'
4) No Change in loop and remove unnecessary states.

Eg

L_1 : Set of all string starting with a



$= a(a+b)^*$



- Quotient operation

Left quotient (cut prefix)
Right quotient (cut suffix)

$$\frac{L_1}{L_2} = \{x \mid xy \in L_1 \text{ for some } y \in L_2\} \quad (R)$$

$$\frac{L_1}{L_2} = \{y \mid xy \in L_1 \text{ for some } x \in L_2\} \quad (L)$$

~~* Context - free Gram~~

* Context free Grammer (CFG) :-

CFG : 4 tuple (V, T, P, S)

V - finite set of Variables (Non terminal)

T - finite set of Terminals ($V \cap T = \phi$)

P - Production Rules

This is a finite set whose elements are $\alpha \longrightarrow \beta$, where α and β are strings on $V \cup T$. α has atleast one symbol from V.

S - Start variable

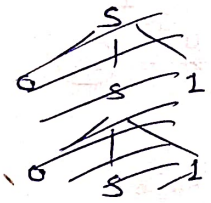
$$\underbrace{\alpha}_{\text{only one variable}} \longrightarrow \underbrace{\beta}_{(V \cup T)^*}$$

Exa-
$$\begin{cases} S \longrightarrow 0A \\ A \longrightarrow A0 \,/\, 0 \end{cases}$$

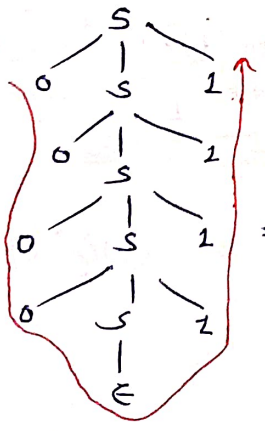
$V = \{S, A\}$, $T = \{0\}$

$S = S$

$P = \{S \longrightarrow 0A,\ A \longrightarrow A0,\ A \longrightarrow 0\}$



Exa $S \longrightarrow 0S1 \,/\, \in$



= 00001111

$L(G) = \{0^n 1^n \mid n \geq 0\}$

$\Rightarrow$ if $G = (V, T, P, S)$

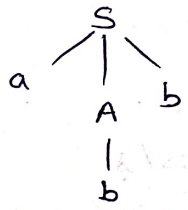
$L(G) = \{w \mid w \in T^*, S \xrightarrow{*} w\}$

($\xrightarrow{*}$: multiple steps)

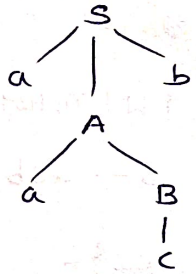
If G_1 and G_2 are Equivalent

$L(G_1) = L(G_2)$

Q
$$\begin{cases} S \longrightarrow aAb \\ A \longrightarrow aB/b \\ B \longrightarrow c \end{cases}$$



$\Rightarrow$ abb

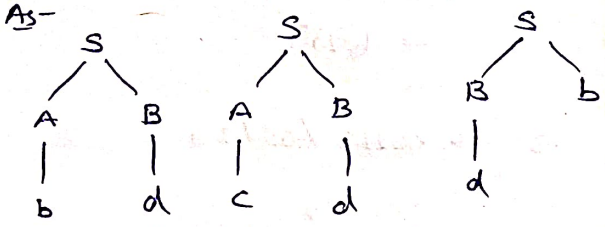


$\Rightarrow$ aacb

$L(G) = \{abb, aacb\}$

Q
$$\begin{cases} S \longrightarrow AB \,/\, Bb \\ A \longrightarrow b/c \\ B \longrightarrow d \end{cases}$$

Ans-

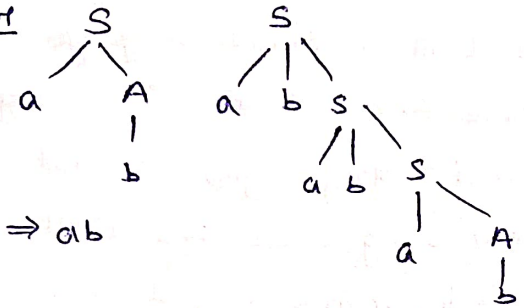


$\Rightarrow$ bd, cd, db

Q $S \longrightarrow aA \mid abS$

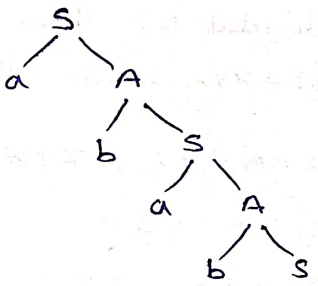
$A \longrightarrow bs \mid b$

Ans



$\Rightarrow ab$

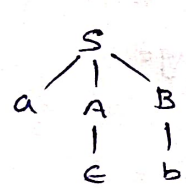
$abab\,ab$



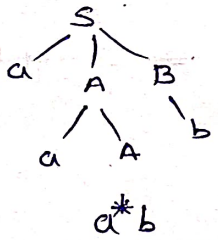
$(ab)^*$

$L(G) = \{ (ab)^n \mid n \geq 1 \}$

Q
$$\begin{cases} S \longrightarrow aAB \\ A \longrightarrow aA \mid \in \\ B \longrightarrow b \end{cases}$$



ab



a^*b

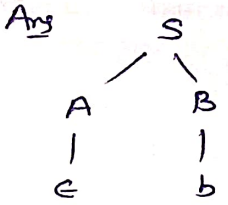
$L(G) = \{ a^n b \mid n \geq 1 \}$

Q $S \longrightarrow AB$

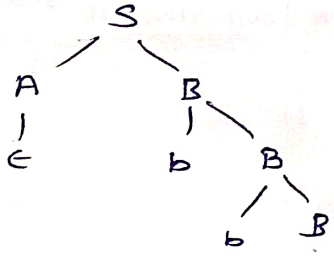
$A \longrightarrow aA \mid \in$

$B \longrightarrow bB \mid b$

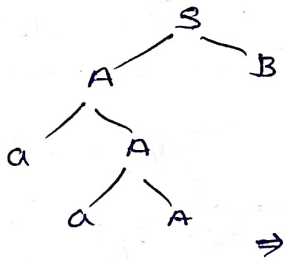
Ans



$\Rightarrow b$



$\Rightarrow b^*$

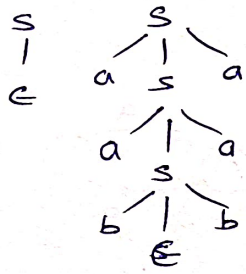


$\Rightarrow a^n b^m$

$L(G) = \{ a^n b^m \mid n \geq 0, m \geq 1 \}$

Q $S \longrightarrow asa \mid bsb \mid \in$

Ans



$\Rightarrow aabbaa$

$L(G) = ww^R$, $|w| \geq 2$

* CFL $\longrightarrow$ CFG

1) $a^n b^n$, $n \geq 0$

$S \longrightarrow asb \mid \in$

or

$asb \mid \lambda$

2) $a^n b^{n+2}$, $n \geq 0$

$S \longrightarrow asb \mid bb$

3) $a^{2n} b^n$, $n \geq 0$

$S \longrightarrow aasb \mid \lambda$

4) $a^{2n+3} b^n \quad n \geq 0$

$S \rightarrow aasb \mid aaa$

5) $a^m b^n$, ~~$m > n$~~ $m > n$, $n \geq 0$

~~$S_1 \rightarrow aS_1b$~~

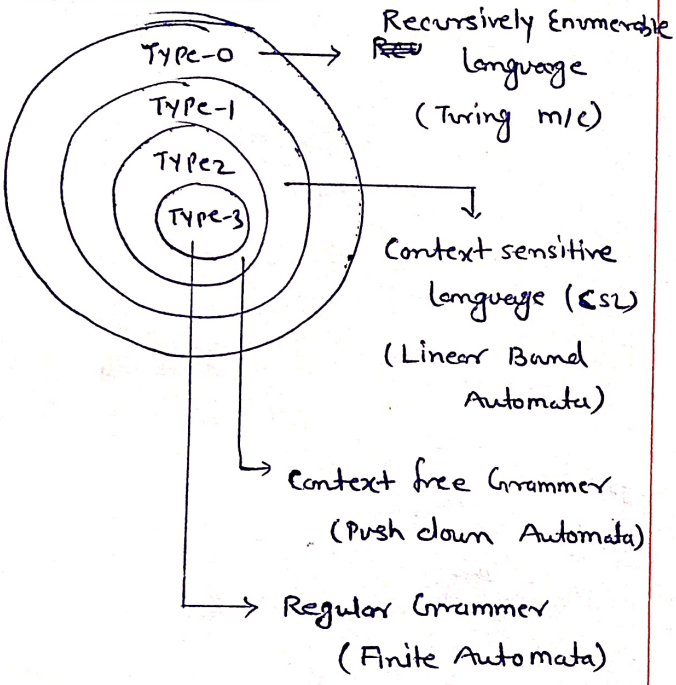
$S \rightarrow AB$

$A \rightarrow aA \mid a$

$B \rightarrow aBb \mid \lambda$

6) $\{w \mid n_a(w) = n_b(w)\}$

* Chomsky Hierarchy :-



Type-0 Grammer :-

- Also called as Unrestricted Grammer
  Phase structured Grammer
  Recursively Enumerable Grammer
- A type-0 Grammer is without restriction.

Type-1 Grammer

- Also known as Case sensitive Grammer, Length Increasing Grammer, Non Contracting Grammer

$\alpha \rightarrow \beta$

$\alpha \in \{\Sigma \cup V_n\}^* V_n \{\Sigma \cup V_n\}^*$

$\in \{T \cup V\}^* V \{T \cup V\}^*$

$\beta \in \{T \cup V\}^+$

$|\alpha| <= |\beta|$

Type-2 Grammer

$\alpha \rightarrow \beta$

$\alpha \in V \quad , \quad |\alpha| = 1$

$\beta \in \{T \cup V\}^*$

Type-3 Grammer :

Used to Generate Regular language

- Regular Grammer Can be of 2 types Either left linear or right linear.
- Left linear support 2 types of production :-

$A \rightarrow a \mid Ba$

$A, B \in V \quad , \quad |A| = |B| = 1$

$a \in T^*$

- Right Linear grammer -

$A \rightarrow a \mid aB$

$A, B \in V \quad , \quad |A| = |B| = 1$

$a \in T^*$

- If the left and right linear Grammer are Combined, the lang need not longer be regular.

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

A) design the grammer in which string w starts with 'abb'

Ans:

$$\begin{cases} S \longrightarrow AB \\ A \longrightarrow abb \\ B \longrightarrow aB \,/\, bB \,/\, \in \end{cases}$$

B) Start and End with a

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

$S \longrightarrow A \,/\, ABA$

$A \longrightarrow a$

$B \longrightarrow aB \,/\, bB \,/\, \in$

Prev Q

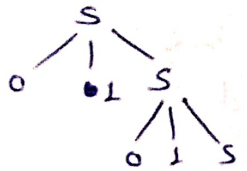
Q $\{w \mid n_a(w) = n_b(w)\}$

$S \rightarrow asb \,/\, bsa \,/\, ss \,/\, \lambda$

ww^R –

$S \longrightarrow asa \,/\, bsb \,/\, \in$

Q $S \longrightarrow 01S \,/\, 01$



$= (01)^+$

Q $S \longrightarrow 0011S \,/\, 01 \,/\, 10$

$= (0011)^* (01 + 10)$

Q $S \longrightarrow 01A \,/\, B11$

$A \longrightarrow 011A \,/\, 01$

$B \longrightarrow 101B \,/\, 11$

Ans $A = (011)^* 01$

$B = (101)^* 11$

$S \longrightarrow 01(011)^* 01 \,/\, (101)^* 1111$

$= 01(011)^* 01 + (101)^* 1111$

Q $\begin{cases} S \rightarrow 011A \,/\, 101B \\ A \rightarrow 110A \,/\, 00 \\ B \rightarrow 11B \,/\, s \end{cases}$

$A = (110)^* 00$

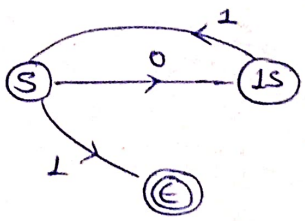
$B = (11)^* s$

$S \longrightarrow 011(11)^* 00 \,/\, 101(11)^* s$

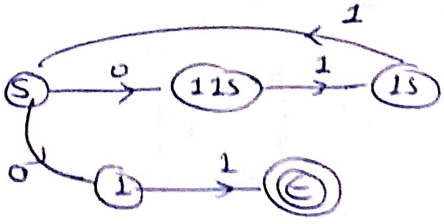
$(101(11)^*)^* (011(110)^* 00)$

* Regular Grammer to FA

1) $S \longrightarrow 01S \,/\, 1$



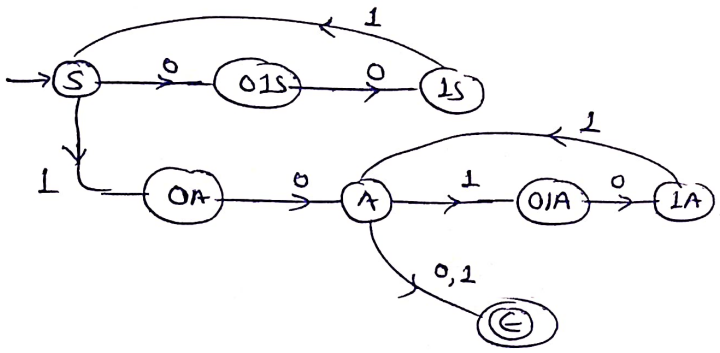
2) $S \longrightarrow 011S \,/\, 01$



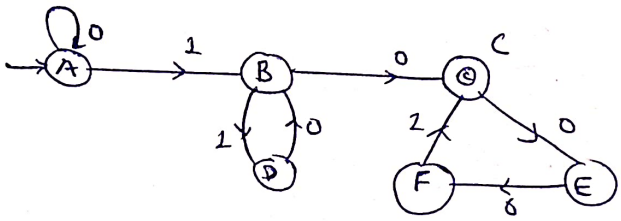
$= (011)^* 01$

⑤ $S \longrightarrow 001S \mid 10A$

$A \longrightarrow 101A \mid 0 \mid 1$



FA → Grammer



$A \longrightarrow 0A \mid 1B$

$B \longrightarrow 1D \mid 0C$

$C \longrightarrow 0E \mid \epsilon$

$D \longrightarrow 0B$

$E \longrightarrow 0F$

$F \longrightarrow 1C$

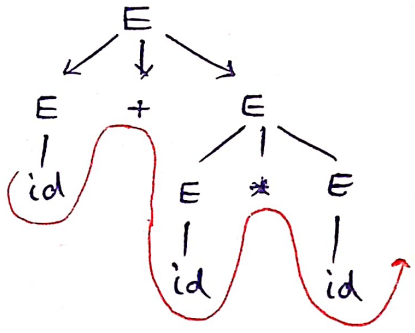
* Derivation:

- The process of deriving a string is known as ~~deri~~ derivation.
- The graphical representation of derivation is known as derivation Tree.
  or
  Syntax Tree
  or
  Parse Tree

∵ $E \rightarrow E+E \mid E*E \mid E=E \mid id$

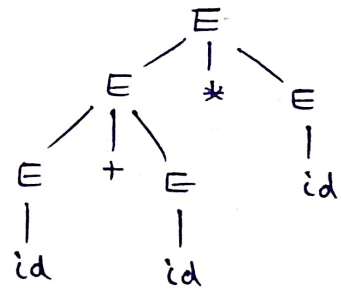
$W = id+id*id$

$E = E+E*E$



- Left most Derivation: (LMD)

Process of Constructing Parse Tree by expanding the left most non Terminal is known as LMD and the graphical representation of LMD is known as LMDT.



LMD

E

E * E

E+E*E

id+E*E

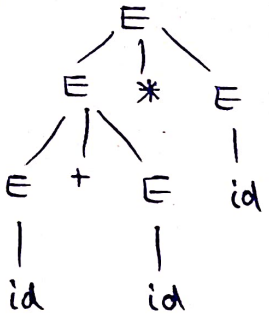
id+id*E

id+id*id

LMD

~~E → E+E~~

- Right most derivation: (RMD)



RMD

E

E + E

E + E * E

E + E * id

E + id * id

id + id * id