Unit - 2

Regular Expressions

RE (mathematical Exp)
| Representation
Generator — Regular language — Acceptor

Regular grammar → Finite Automata

Regular grammar: Right linear grmr, Left linear grmr

Regular ~~set~~:- Any language accepted by finite automata is called regular set.

or

Finite set of group of strings or words called as regular set.

eg:- $\Sigma = \{a, b\}$ then $\{a, aa, bb, abb, \ldots\}$ are Regular sets accepted by finite automata

let 'R,S' are regular sets then

RS, $R \cup S$, R^*, S^* are also regular sets

Regular Expression:

An expression used to represent regular set is called 'regular expression'.

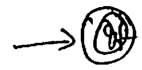
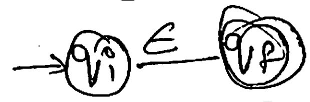
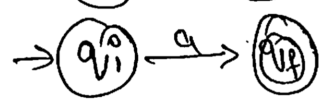
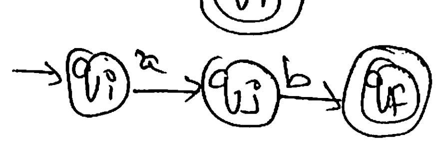
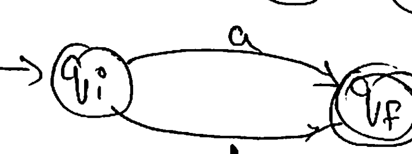
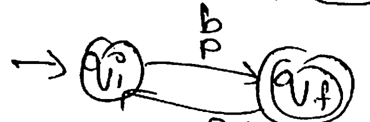
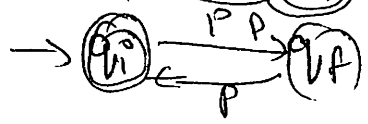
eg: $(0+1)^* (00+11)(0+1)^*$ — follows a seq of 0's & 1's and has a consecutive 0's & 1's followed by seq of 0's & 1's.

If $\phi \rightarrow$ regular Expression for empty set
$\epsilon \rightarrow$ RE for Empty string
$a \in \Sigma \rightarrow$ RE for set consisting of $\{a\}$
$\phi, \epsilon, a \rightarrow$ (Primitive RE)

RE contains three operations
1) Union (+)
2) Concatenation (·)
3) Kleene closure (*)

If r_1 & r_2 are RE then r_1+r_2, $r_1 \cdot r_2$, r_1^* are R.E

Eg 0^*0 – RE for regular set consisting of 0's.
$(0+1)^*$ – RE for regular set consisting of sequence of 0's & 1's.

Regular Expression	Regular set	Finite Automata
ϕ	$\{\ \}$	
ϵ	$\{\epsilon\}$	
a	$\{a\}$	
a^*	$\{\epsilon, a, aa, \ldots\}$	
$a^+ = a \cdot a^*$ $a^* \cdot a$	$\{a, aa, aaa, \ldots\}$	
$(a+b)^*$	$\{\epsilon, a, aa, b, bb, ab, \ldots\}$	
ab	$\{a\}\{b\} = \{ab\}$	
$a+b$	$\{a, b\}$ $\{a\} \cup \{b\}$	
$p(pp)^*$ $(pp)^*p$	$\{p, ppp, ppppp, \ldots\}$	
$(pp)^*$	$\{\epsilon, pp, pppp, \ldots\}$	

languages represented by regular expression is represented as L(s).

$rs \rightarrow L(rs) \rightarrow L(r)L(s)$ concatenation

$r+s \rightarrow L(r+s) \rightarrow L(r) \cup L(s)$

$r^* \rightarrow L(r^*) \Rightarrow [L(r)]^* = \epsilon \cup L(r) \cup (L(r))^2 \cup \dots$

$r^+ \rightarrow L(r^+) = [L(r)]^+ = L(r) \cup [L(r)]^2 \cup \dots$

Note:- r^* is called as Kleen closure

i.e $L(r^*) = L^* = \bigcup_{i=0}^{n} L^i$

r^+ is called Positive closure

i.e. $L(r^+) = L^+ = \bigcup_{i=1}^{n} L^i$

Eg: $\Sigma = \{a,b\}$, identify the RE for the language having length 2 i.e $|w| = 2$

$L_1 = \{aa, bb, ab, ba\}$

$aa + ab + ba + bb$

$= a(a+b) + b(a+b)$

$= (a+b)(a+b)$ — RE

eg:- Identify the RE for the lang of length atleast 2

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

Eg:- atmost '2' - 0, 1, 2

$\Rightarrow \{\epsilon, a, b, aa, ab, ba, bb\}$

$(a+b+\epsilon) \cdot (a+b+\epsilon)$

→ $\Sigma = \{a,b\}$ Even length strings

$L = \{\epsilon, aa, ab, ba, bb, ---\}$

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

→ $\Sigma = \{a,b\}$ odd length strings

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

→ divisible by '3'

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

→ $\Sigma = \{a,b\}$ Exactly 2 a's

$RE = b^*ab^*ab^*$

→ $\Sigma = \{a,b\}$ atleast 2 a's

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

→ $\Sigma = \{a,b\}$ atmost 2 a's

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

Language represented by RE

① $r = (a+b)^*(a+bb)$ signifies that it accepts sequence of a & b later followed by a or bb.

$L(r) = \{a, bb, aa, abb, ba, bbb, ---\}$

2) $r = (a+b)a^*$

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

$= L(a^*) \cdot L(a) \cup L(b)$

$= \{\epsilon, a, aa, --\} (L(a) \cup L(b))$

$= \{\epsilon, a, aa, --\} \{a, b\}$

$= \{a, aa, aaa, b, ab, aab, ----\}$

~~Inden~~

Identity rules for regular Expression: (or) Algebraic laws for RE

1) $\phi + r = r$
2) $\phi \cdot r = r \cdot \phi = \phi$
3) $\epsilon \cdot r = r \cdot \epsilon = r$
4) $\epsilon^* = \epsilon$ or $\phi^* = \epsilon$
5) $r + r = r$
6) $r^* \cdot r^* = r^*$
7) $r \cdot r^* = r^* \cdot r$
8) $(r^*)^* = r^*$
9) $\epsilon + rr^* = \epsilon + r^* r = r^*$
10) $(pq)^* p = p(qp)^*$
11) $(p+q)^* = (p^* q^*)^* = (p^* + q^*)^*$
12) $(p+q)r = pr + qr$

$p(q+r) = pq + pr$

∴ Arden's 1 $\quad p \neq \epsilon \quad r = q + rp \quad r = qp^*$

① $r = \epsilon + 1^*(011)^*\left(1^*(011)^*\right)^* = (1+011)^*$

$= \epsilon + \underbrace{1^*(011)^*}_{r}\left(\underbrace{1^*(011)^*}_{r}\right)^* \quad \therefore \epsilon + rr^* = r^*$

$= \left(\underbrace{1}_{P}^*\underbrace{(011)}_{q}^*\right)^* \quad \therefore (P^*q^*)^* = (P+q)^*$

$\Rightarrow (1+(011))^*$

② $(1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1) = 0^*1(0+10^*1)^*$

$\Rightarrow (1+00^*1)\left(\epsilon + \underbrace{(0+10^*1)^*}_{r^*}\underbrace{(0+10^*1)}_{r}\right) \quad \epsilon + r^*r = r^*$

$\Rightarrow (1+00^*1)(0+10^*1)^*$

$\Rightarrow 1(\epsilon+00^*)(0+10^*1)^*$

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

Additional Examples:-

1) $L = \{1^{2n+1} \mid n \geq 0\}$

$= \{1, 111, 11111, ---\} = 1(11)^*$ or $(11)^*1$

2) strings having any no. of 0's & any no. of 1's.

$(0+1)^*$

3) strings ends with 00 over the $\{0,1\}$

$(0+1)^*00$

4) strings begining with 0 & ending with 11

$0(0+1)^*11$

5) 4th char from the rightend is always 'a'

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

6) P.T $(0+11^*0) + (0+11^*0)(10+10^*1)^*(10+10^*1) = 1^*0(10+10^*1)^*$

$\Rightarrow (0+11^*0)\left(\epsilon + \underbrace{(10+10^*1)^*}_{r^*}\underbrace{(10+10^*1)}_{r}\right)$

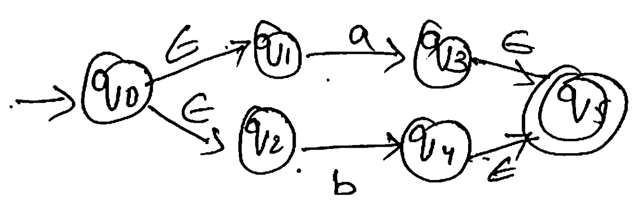
$\Rightarrow (0+11^*0)(10+10^*1)^*$

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

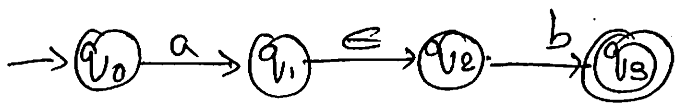
$\Rightarrow 01^*(10+10^*1)^*$ //

Equivalance of RE & FA using 'ϵ' :-

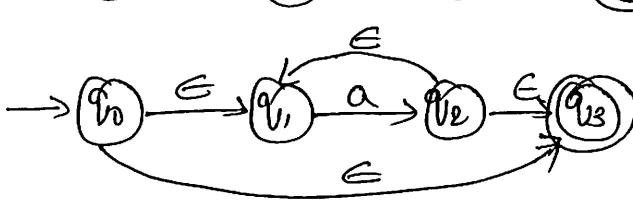
a+b



a·b



a^*



Theorem: Every language defined by a regular expression is also defined by a finite automata

Proof: Suppose $L = L(R)$ for a regular expression R. We show that $L = L(E)$ for some ϵ-NFA with

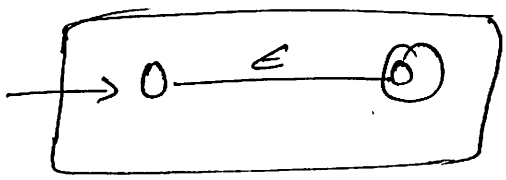
1. Exactly one accepting state
2. No arcs into the initial state
3. No arcs out of the accepting state

Basis:- There are three parts to the basis

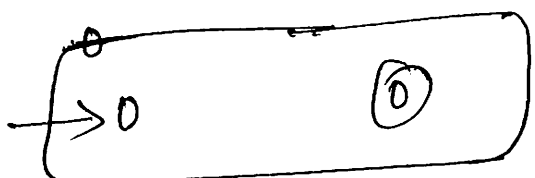
a) How to handle the expression ϵ. Since the only path from the start state to an accepting state is labeled 'ϵ'.

b) Shows the construction for $\emptyset$. Clearly there are no paths from start state to accepting state. So $\emptyset$ is the language to this automation.

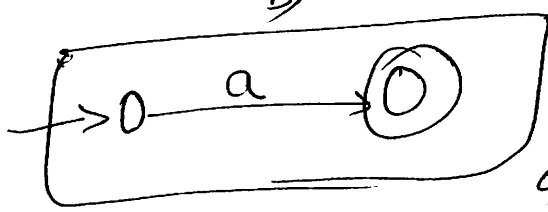
c) Gives the automation for a regular expression 'a'. The language consists of the one string a, which is also $L(a)$.



a)



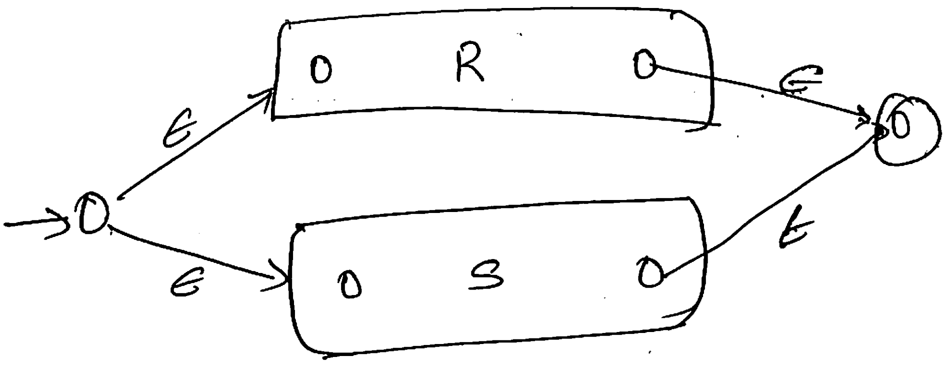
b)



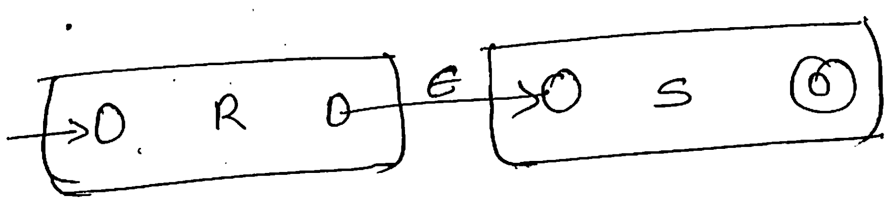
c)

Induction: The languages of these subexpressions are also the languages of ϵ-NFA's with a single accepting state. The (~~four~~ Three) cases are:

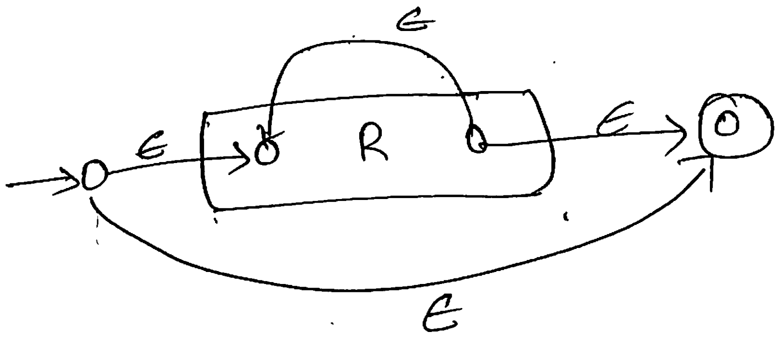
1) For the expression 'R+S', the language of the automaton is $L(R) \cup L(S)$.



2) for the expression 'RS', the language of the automaton is $L(R)L(S)$.



3) for the expression R^*

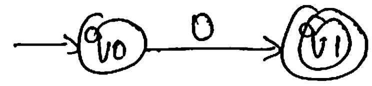


Constructe-NFA for the following RE.

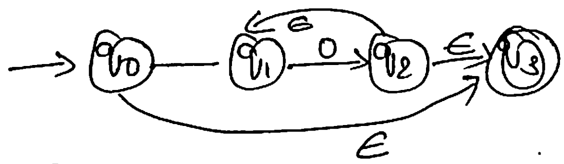
1) $R = (0+1)0^*$

$r_1 = 0$
$r_2 = r_1^*$
$r_3 = 1$
$r_4 = r_1 + r_3$
$r_5 = r_4 \cdot r_2$

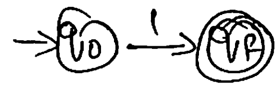
$r_1 = 0$



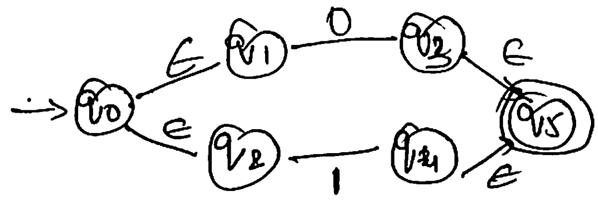
$r_2 = r_1^*$



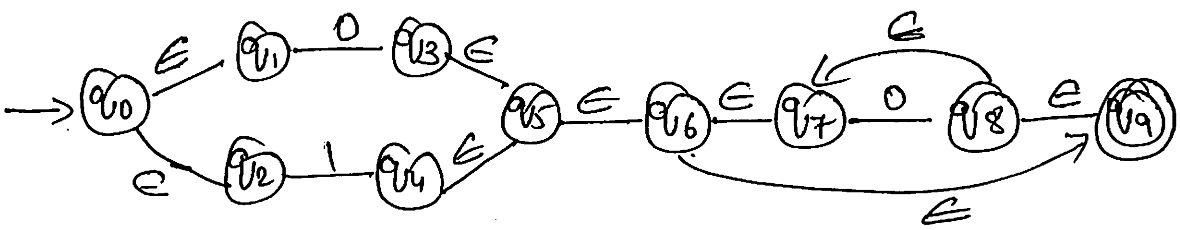
$r_3 = 1$



$r_4 = r_1 + r_3$

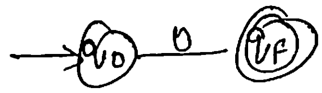


$r_5 = r_4 \cdot r_2$

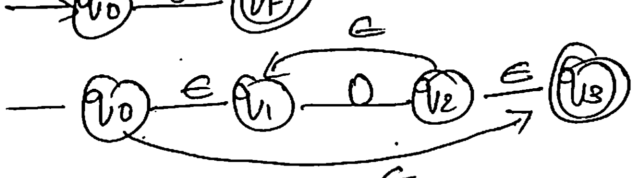


2) $R = (10^*)^*$

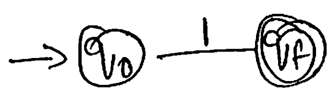
$r_1 = 0$



$r_2 = r_1^*$



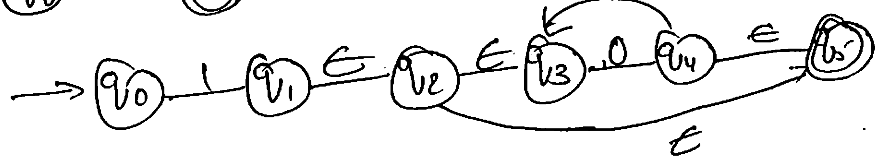
$r_3 = 1$



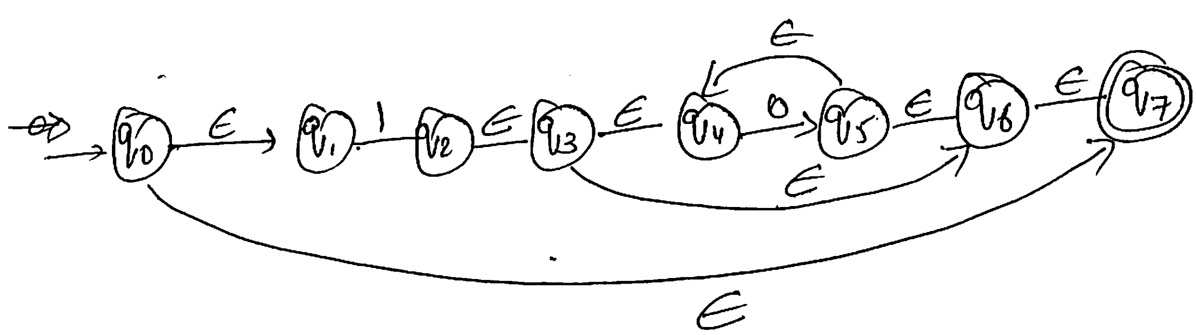
$r_4 = r_3 \cdot r_2$

$r_5 = r_4^*$

$r_4 =$

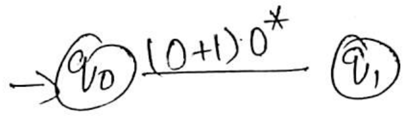


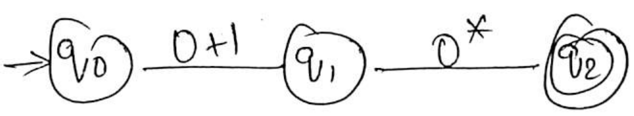
$r_5 =$



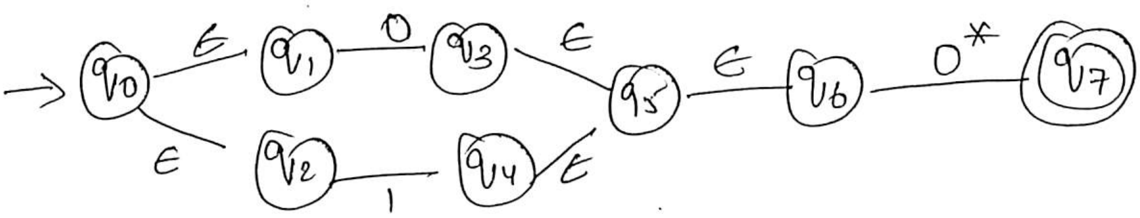
Alternative method for RE to FA:-

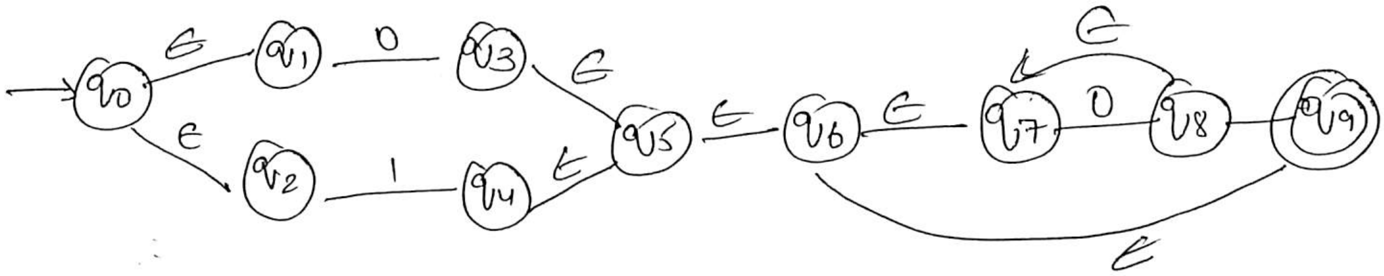
$(0+1)0^*$

①


②


③


④


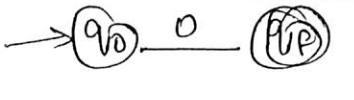
⑤


3) $R = 0 + 01^*$

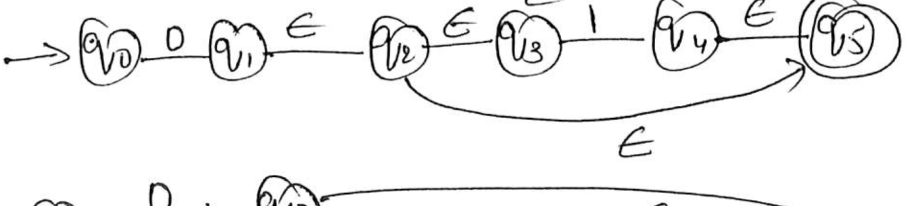
$r_1 = 1$

$r_2 = r_1^*$

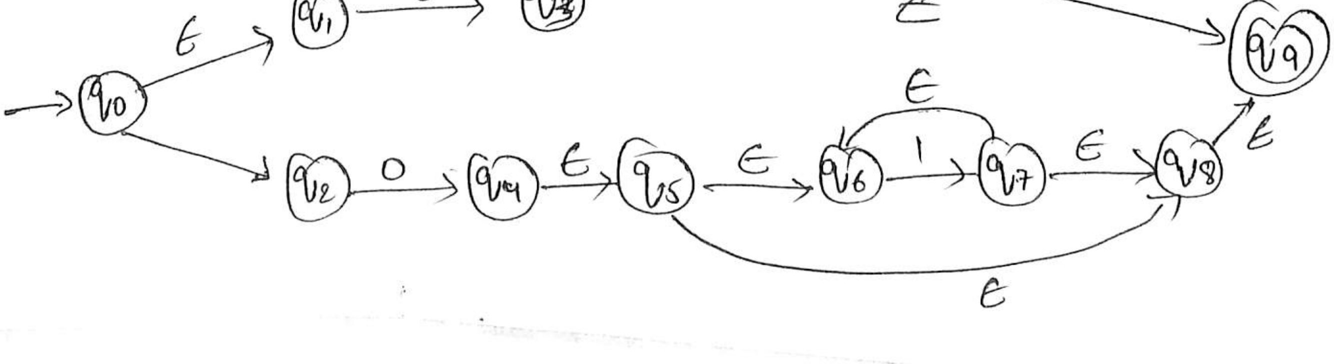
$r_3 = 0$



$r_4 = r_3 \cdot r_2$

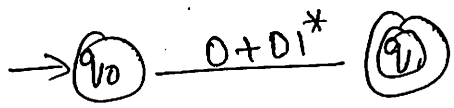


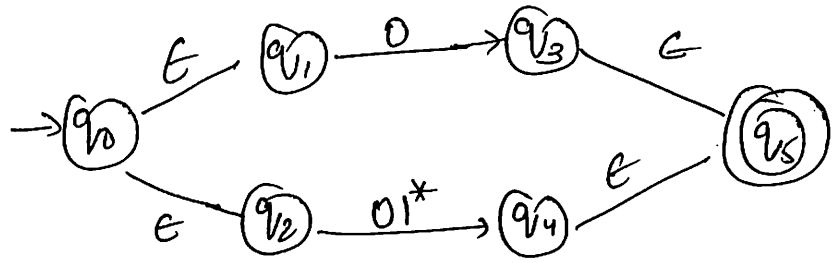
$r_5 = r_3 + r_4$

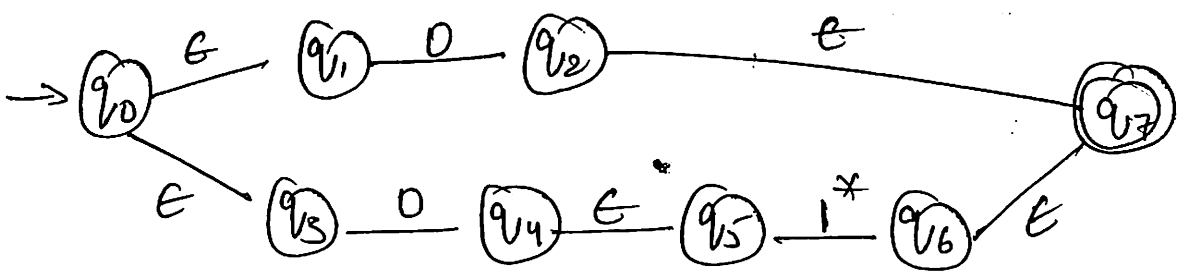


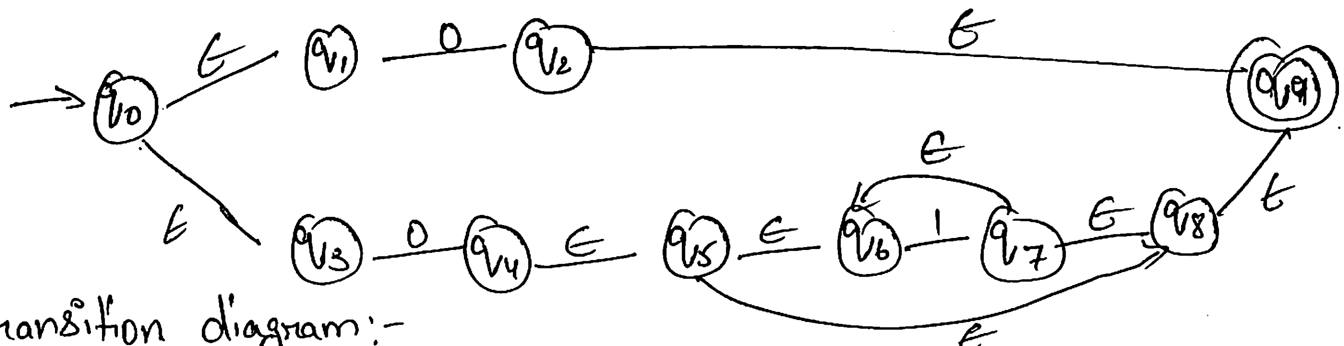
alternative method:

$R = 0 + 01^*$

① 

② 

③ 

④ 

Transition diagram:-

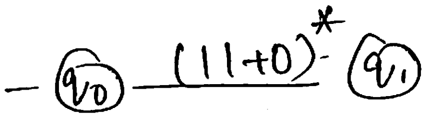
	0	1	ε
$\rightarrow q_0$	—	—	$\{q_1, q_3\}$
q_1	$\{q_2\}$	—	—
q_2	—	—	$\{q_9\}$
q_3	$\{q_4\}$	—	—
q_4	—	—	$\{q_5\}$
q_5	—	—	$\{q_6, q_8\}$
q_6	—	$\{q_7\}$	—
q_7	—	—	$\{q_6, q_8\}$
q_8	—	—	$\{q_9\}$
$* q_9$	-	—	—

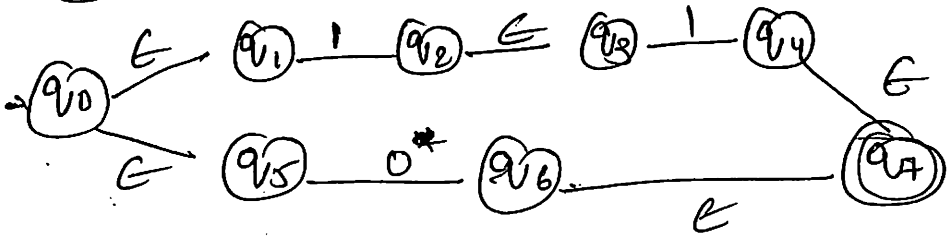
→ Given $r = (11+0)^*(00+1)^*$ construct NFA with ε-moves by using alternative method

let $r = r_1 \cdot r_2$

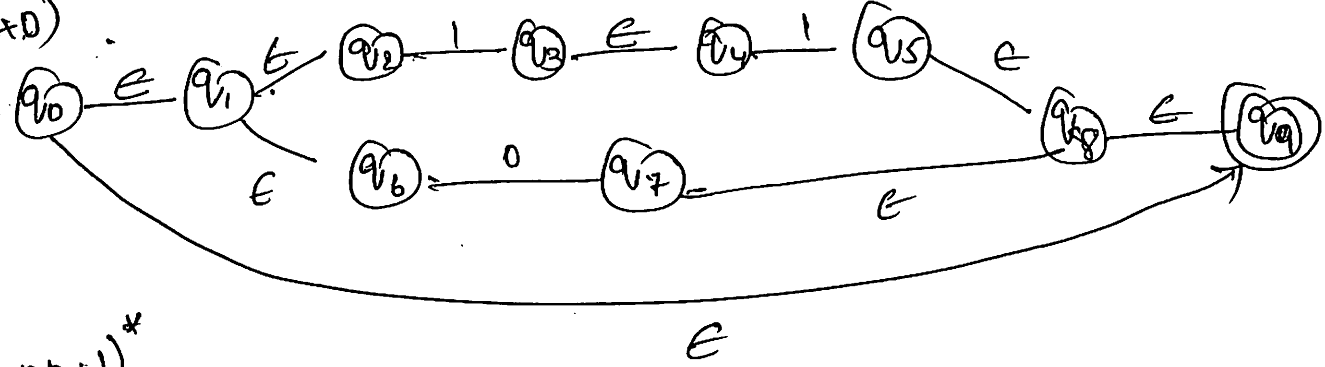
$r_1 = (11+0)^*$

$r_2 = (00+1)^*$

$r_1 =$ 

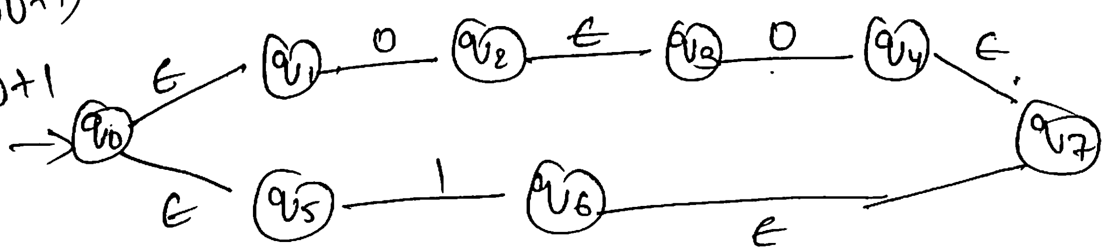
$r_3 = 11+0 \Rightarrow$ 

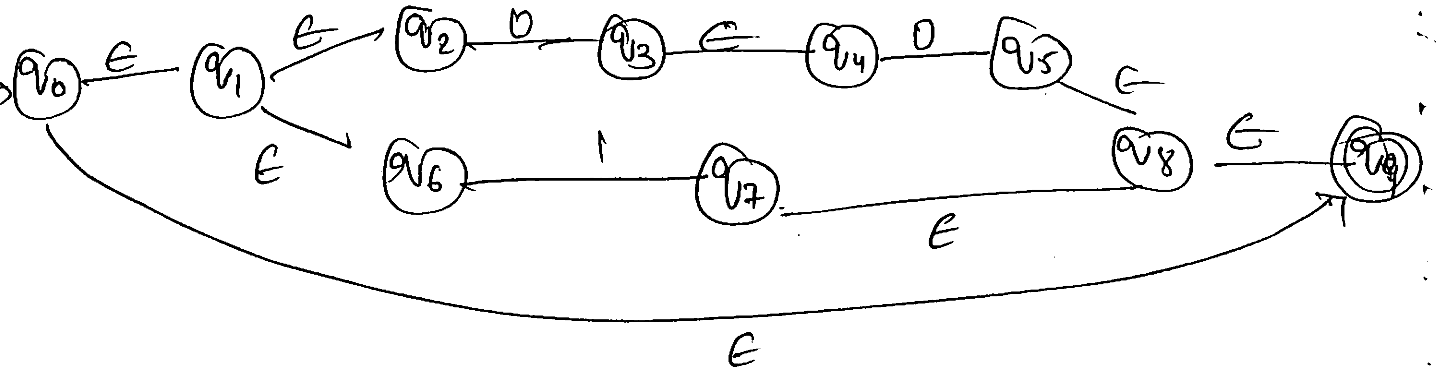
$(11+0)^*$



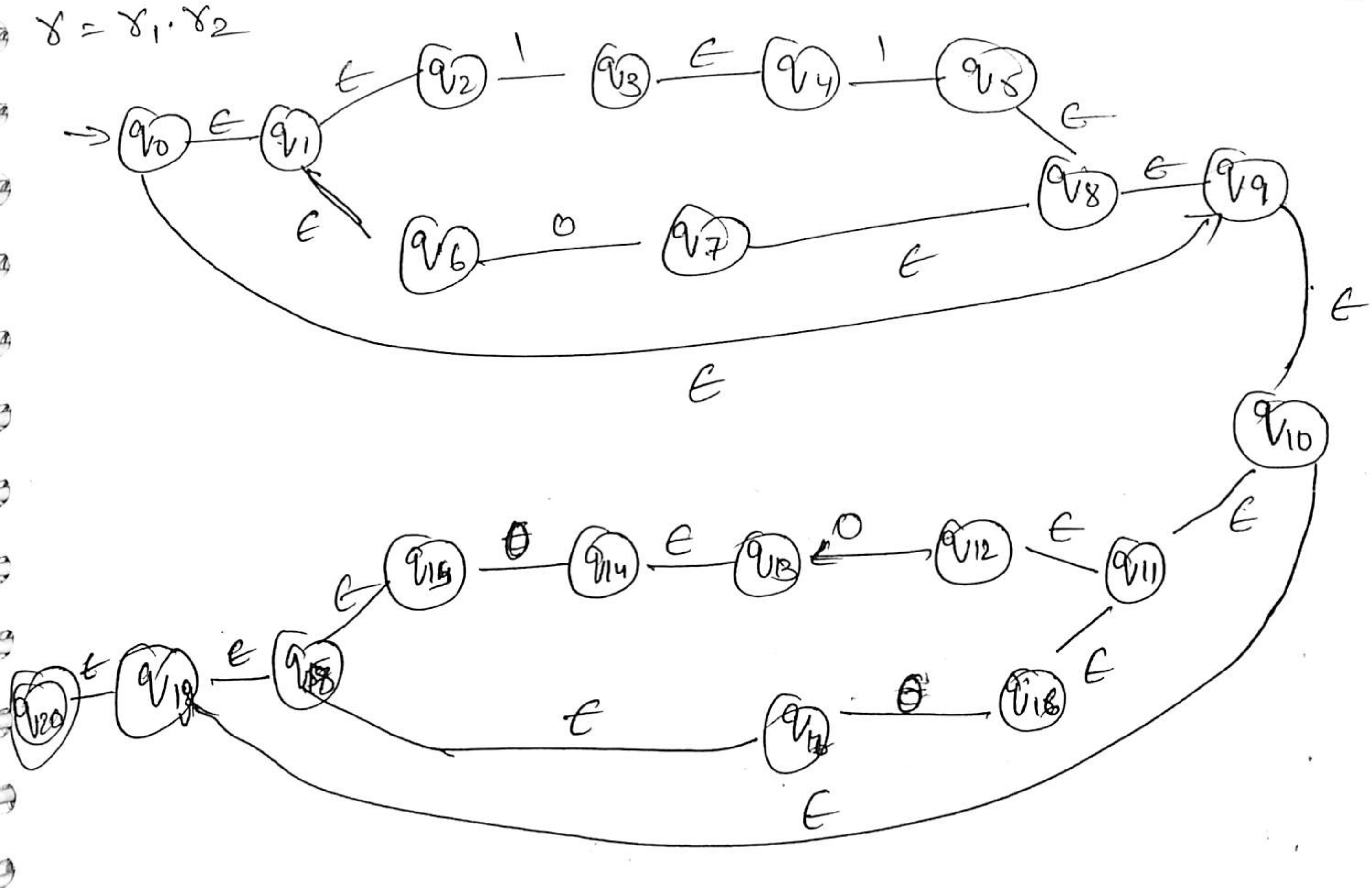
$r_2 = (00+1)^*$

$r_4 = 00+1$



$r_2 =$ 

$r = r_1 \cdot r_2$



More Examples:

→ $R.E = 1(11)^*$

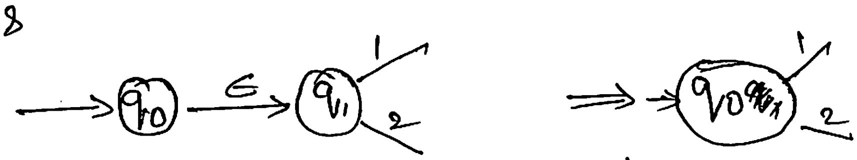
→ $R.E = (01+2^*)0$

→ $R.E = a^* + b^* + c^*$

→ $(a+b)^* aa (a+b)^*$

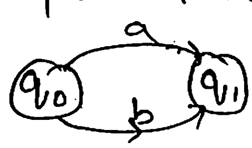
→ $r = (1+01+001)^* (\epsilon+0+00)$

→ Alternative method for constructing NFA-∈ to NFA is

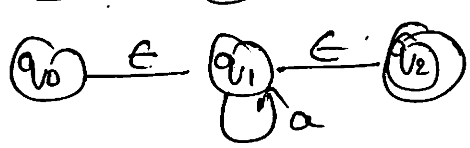
if a) 

b) 

In alternate method for finite automata we can use following representation:

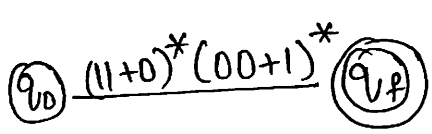
$a+b$ 

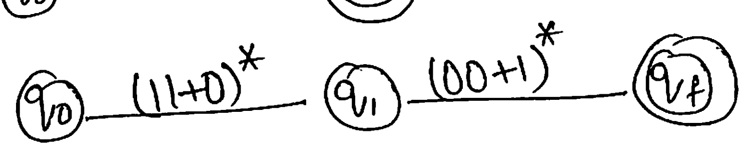
$a \cdot b$ 

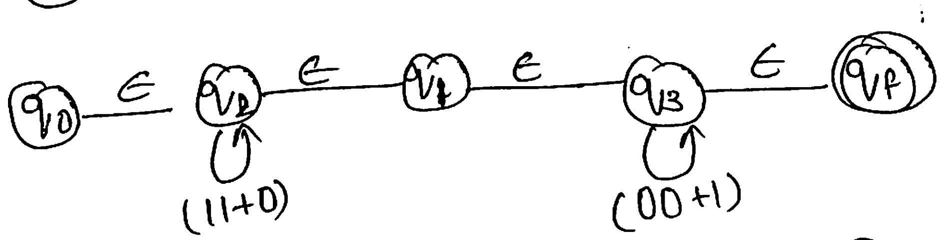
a^* 

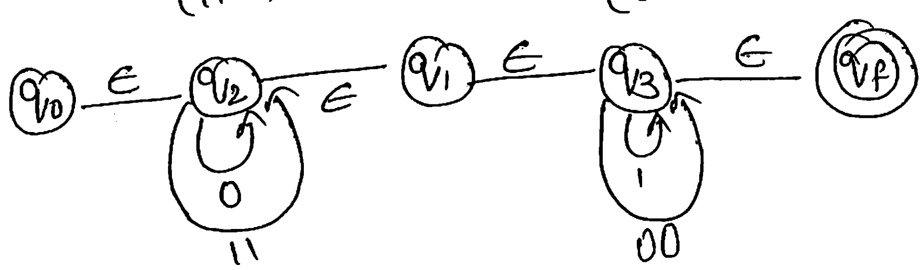
Ex.:

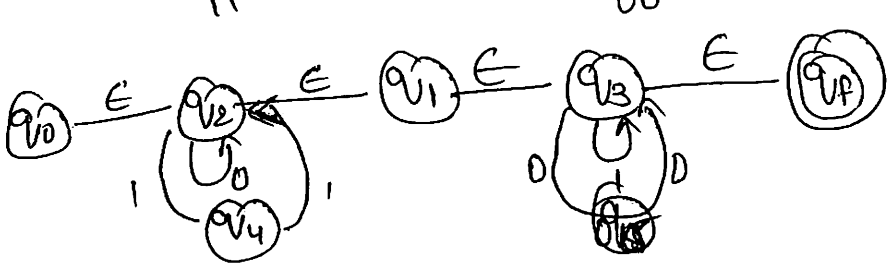
$r = (11+0)^*(00+1)^*$











Transition table of NFA with ϵ-moves

	0	1	ϵ
q_0	—	—	$\{q_2\}$
q_1	—	—	$\{q_3\}$
q_2	$\{q_2\}$	$\{q_4\}$	$\{q_1\}$
q_3	$\{q_5\}$	$\{q_3\}$	$\{q_f\}$
q_4	—	$\{q_2\}$	—
q_5	$\{q_3\}$	—	—
$*q_f$	—	—	—

ϵ-NFA to NFA:

ϵ-closure of $(q_0) = \{q_0, q_2, q_1, q_3, q_f\}$
ϵ-closure $(q_1) = \{q_1, q_3, q_f\}$
ϵ-closure $(q_2) = \{q_2, q_1, q_3, q_f\}$
ϵ-closure $(q_3) = \{q_3, q_f\}$
ϵ-closure $(q_4) = \{q_4\}$
ϵ-closure $(q_5) = \{q_5\}$
ϵ-closure $(q_6) = \{q_6\}$

by using extended function

$\delta^*(q_0, 0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0, \epsilon), 0))$
$= \epsilon\text{-closure}(\delta(q_0, q_1, q_2, q_3, q_f), 0)$
$= \epsilon\text{-closure}(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0) \cup \delta(q_3, 0) \cup \delta(q_f, 0))$
$\Rightarrow \epsilon\text{-closure}(\emptyset \cup \emptyset \cup \{q_2\} \cup \{q_5\} \cup \{\cancel{q}_3\})$
$= \epsilon\text{-closure}(q_2, q_5)$
$= \{q_5\} \cup \{q_2, q_1, q_3, q_f\}$
$= \{q_1, q_2, q_3, q_5, q_f\}$ //

$\delta'(q_0,1) = \epsilon\text{-closure}(q_4, q_3)$
$= \{q_3, q_4, q_f\}$

$\delta'(q_1,0) = \epsilon\text{-closure}(q_5)$
$= \{q_5\}$

$\delta'(q_1,1) = \epsilon\text{-closure}(q_3)$
$= \{q_3, q_f\}$

$\delta'(q_2,0) = \epsilon\text{-closure}(q_2, q_5)$
$= \{q_1, q_2, q_3, q_5, q_f\}$

$\delta'(q_2,1) = \epsilon\text{-closure}(q_4, q_3)$
$= \{q_3, q_4, q_f\}$

$\delta'(q_3,0) = \epsilon\text{-closure}(q_5)$
$= \{q_5\}$

$\delta'(q_3,1) = \epsilon\text{-closure}(q_3)$
$= \{q_3, q_f\}$

$\delta'(q_4,0) = \epsilon\text{-closure}(\emptyset)$
$= \emptyset$

$\delta'(q_4,1) = \epsilon\text{-closure}(q_2)$
$= \{q_1, q_2, q_3, q_f\}$

$\delta'(q_5,0) = \epsilon\text{-closure}(q_3)$
$= \{q_3, q_f\}$

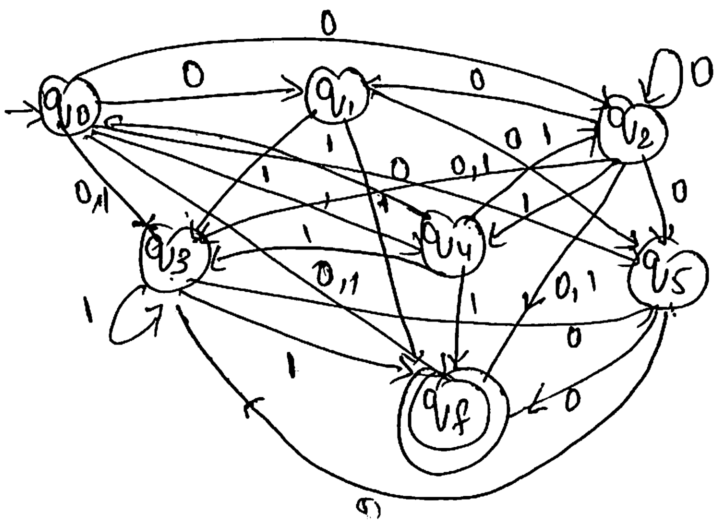
$\delta'(q_5,1) = \epsilon\text{-closure}(\emptyset)$
$= \emptyset$

$\delta'(q_f,0) = \epsilon\text{-closure}(\emptyset)$
$= \emptyset$

$\delta'(q_f,1) = \epsilon\text{-closure}(\emptyset)$
$= \emptyset$

Transition table of NFA

δ'	0	1
q_0	$\{q_1, q_2, q_3, q_5, q_f\}$	$\{q_3, q_4, q_f\}$
q_1	$\{q_5\}$	$\{q_3, q_f\}$
q_2	$\{q_1, q_2, q_3, q_5, q_f\}$	$\{q_3, q_4, q_f\}$
q_3	$\{q_5\}$	$\{q_3, q_f\}$
q_4	$\emptyset$	$\{q_1, q_2, q_3, q_f\}$
q_5	$\{q_3, q_f\}$	$\emptyset$
q_f	$\emptyset$	$\emptyset$



→ Construction of DFA equivalent to a regular Exp $(0+1)^*(00+11)(0+1)^*$ & also find the reduced DFA.

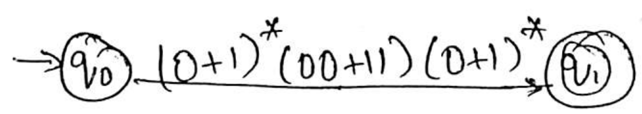
given R.E is $(0+1)^*(00+11)(0+1)^*$

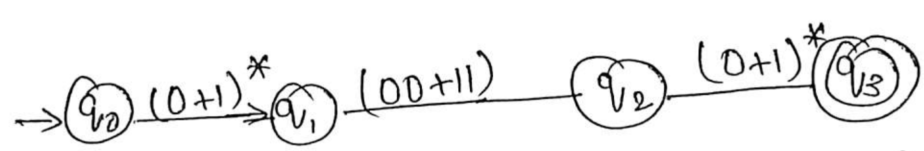
→ First construct the transition graph with ϵ using construction rules.

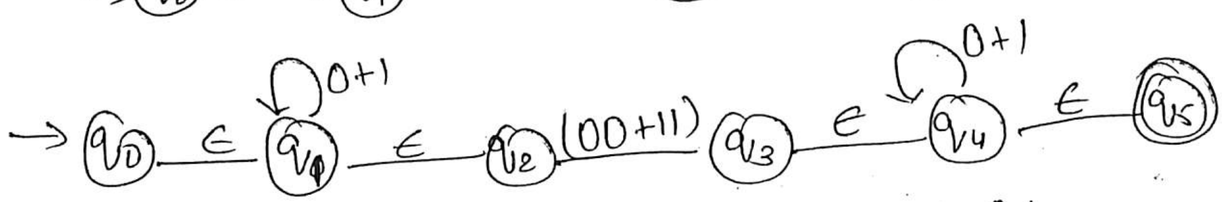
→ Construction of DFA equivalent to a regular Expression $(0+1)^*(00+11)(0+1)^*$ & also find the reduced DFA.

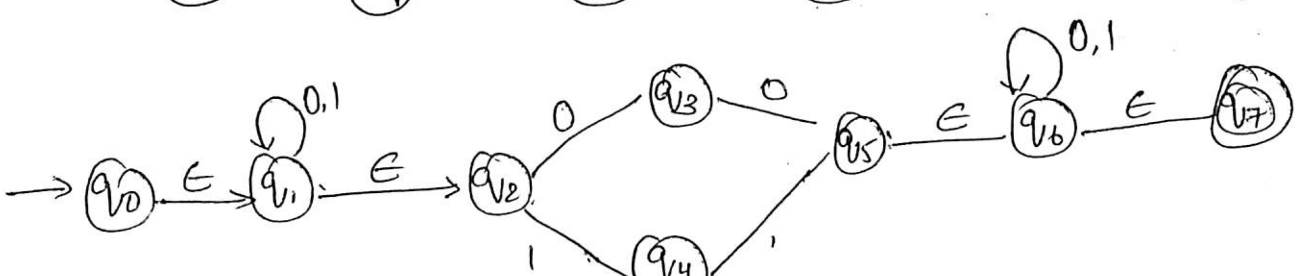
Sol: Given R.E is $(0+1)^*(00+11)(0+1)^*$

First construct transition graph with 'ε' using the

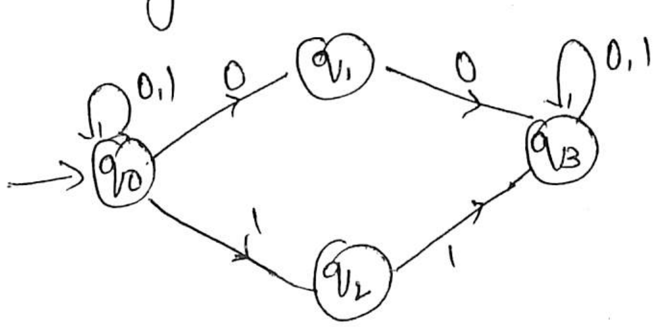








Transition graph without ε-moves:



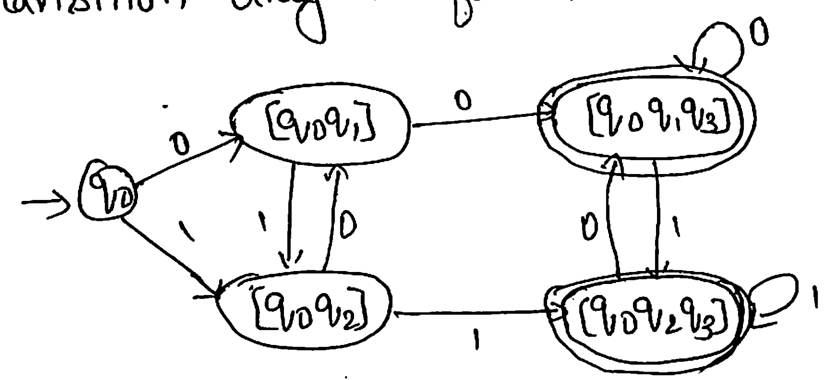
NFA without ε-moves

	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_0, q_2\}$
q_1	$\{q_3\}$	—
q_2	—	$\{q_3\}$
* q_3	$\{q_3\}$	$\{q_3\}$

Construct DFA for NFA

states	0	1
→ [q_0]	[$q_0 q_1$]	[$q_0 q_2$]
[$q_0 q_1$]	[$q_0 q_1 q_3$]	[$q_0 q_2$]
[$q_0 q_2$]	[$q_0 q_1$]	[$q_0 q_2 q_3$]
*[$q_0 q_1 q_3$]	[q_0, q_1, q_3]	[$q_0 q_2 q_3$]
*[$q_0 q_2 q_3$]	[$q_0 q_1 q_3$]	[$q_0 q_2 q_3$]

Transition diagram for DFA is :

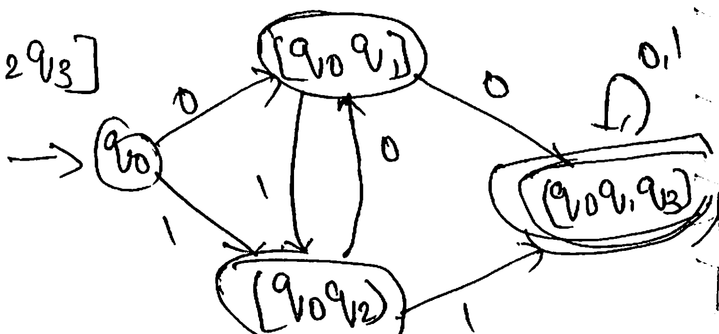


Reduce the number of states of above DFA

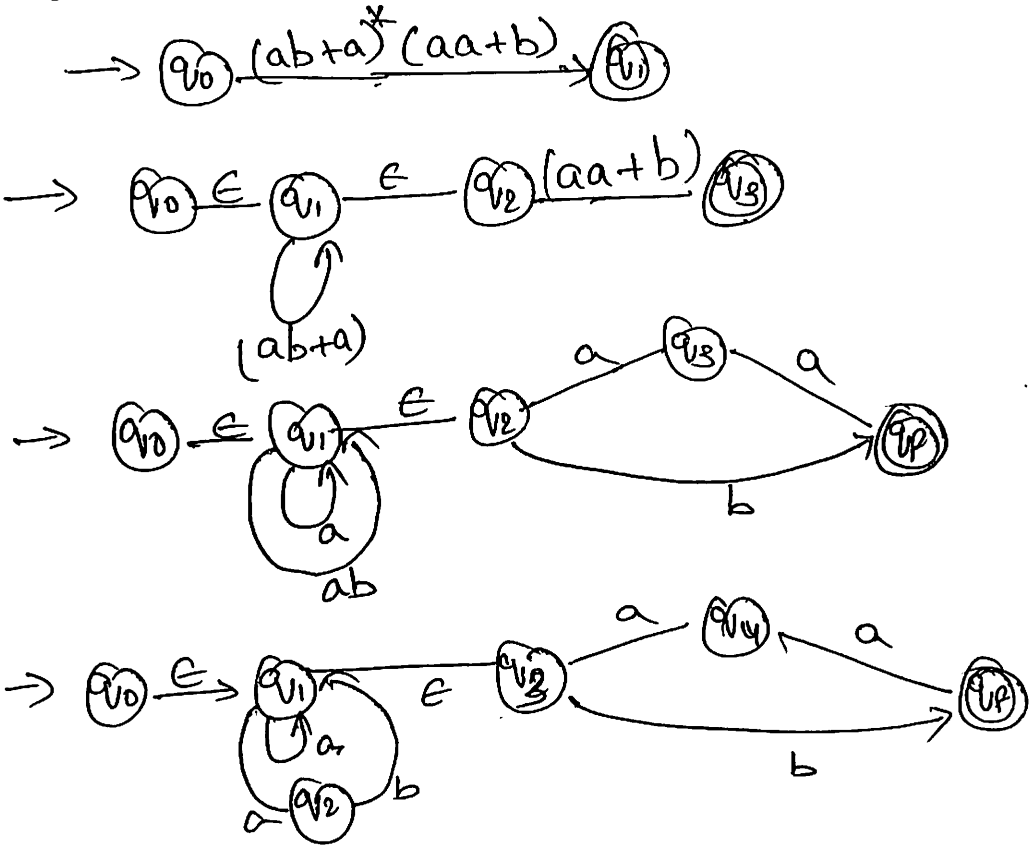
As the rows corresponding to [$q_0, q_1 q_3$] & [$q_0 q_2 q_3$] are identical & delete the last row [$q_0 q_2 q_3$]

∴

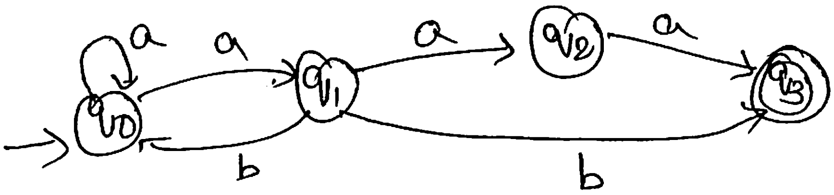
δ' states	0	1
→ q_0	[$q_0 q_1$]	[$q_0 q_2$]
[$q_0 q_1$]	[$q_0 q_1 q_3$]	[$q_0 q_2$]
[$q_0 q_2$]	[$q_0 q_1$]	[$q_0 q_2 q_3$]
* [$q_0 q_1 q_3$]	[$q_0 q_1 q_3$]	[$q_0 q_2 q_3$]



→ Construct transition systems equivalent to the regular expression $(ab+a)^*(aa+b)$.



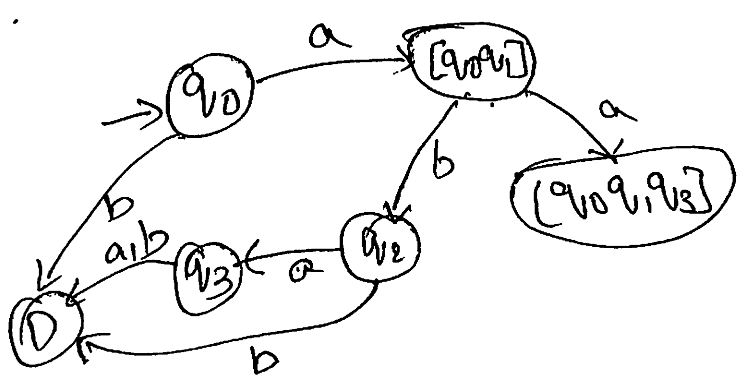
NFA without 'ε'



	a	b
$\rightarrow q_0$	$\{q_0, q_1\}$	—
q_1	$\{q_3\}$	$\{q_2\}$
q_2	$\{q_3\}$	—
$*q_3$	—	—

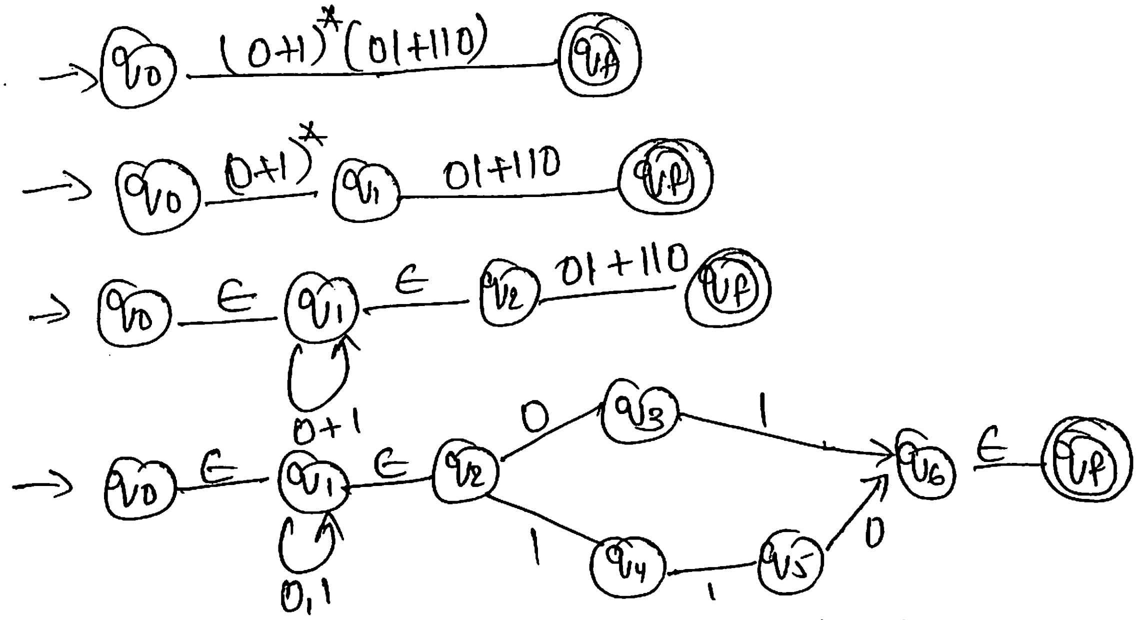
DFA ⟹

δ'	a	b
$\rightarrow q_0$	$[q_0q_1]$	$\cancel{\emptyset}$-D
$[q_0q_1]$	$[q_0q_1q_3]$	$[q_2]$
$[q_2]$	$[q_3]$	D
$[q_0q_1q_3]$	$[q_0q_1q_3]$	D
$[q_3]$	D	D

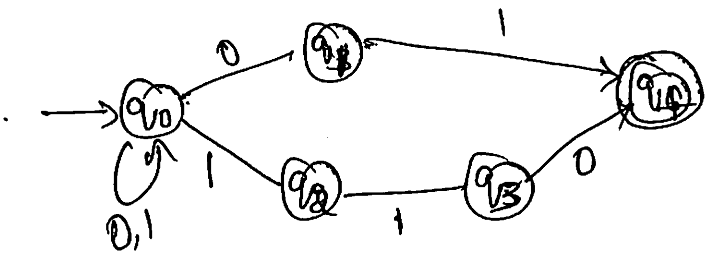


→ $(0+1)^*(01+110)$

→ First construct NFA with $\in$



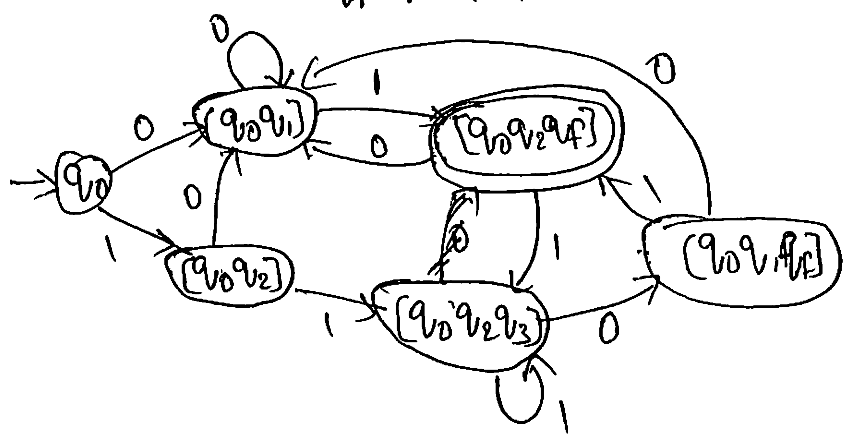
Convert NFA-$\in$ to NFA without $\in$



δ	0	1
$\to q_0$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
q_1	—	$\{q_f\}$
q_2	—	$\{q_3\}$
q_3	$\{q_f\}$	—
$*q_f$	—	—

δ'	0	1
$\to q_0$	~~$[q_0q_1]$~~	~~$[q_0q_2]$~~
~~$[q_0q_1]$~~	~~$[q_0q_3q_f]$~~	~~$[q_0q_4]$~~
~~$[q_0q_4]$~~		

δ'	0	1
$\to q_0$	$[q_0q_1]$	$[q_0q_2]$
$[q_0q_1]$	$[q_0q_1]$	$[q_0q_2q_f]$
$[q_0q_2]$	$[q_0q_1]$	$[q_0q_2q_3]$
$*[q_0q_2q_f]$	$[q_0q_1]$	$[q_0q_2q_3]$
$[q_0q_2q_3]$	$[q_0q_1q_f]$	$[q_0q_2q_3]$
$*[q_0q_1q_f]$	$[q_0q_1]$	$[q_0q_2q_f]$



* Convertion from FA to RE

Arden's Theorem:

→ It is useful for checking the equivalence of two R.E & also useful in Conversion of DFA to R.E

Let P, q, r be regular expressions without empty sets then $r = P + qr \Rightarrow r = pq^*$

1) If q_i, q_j are two nodes & there is an edge label 'a' from state q_j to q_i, then the equation is

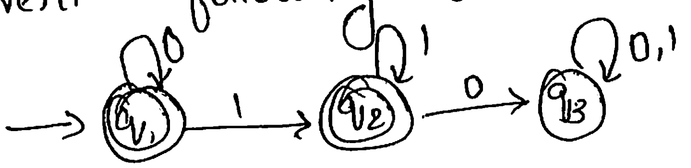
$$\boxed{q_i = q_j \cdot a}$$

2) If q_i has edges from $q_1, q_2, \ldots q_n$ with labels '$a_1, a_2, \ldots a_n$' then the equation is

$$\boxed{q_i = q_1 a_1 + q_2 a_2 + \cdots + q_n a_n}$$

3) Add ϵ if it is initial state

Eg:- Convert following automata into regular expression



Sol:

$q_1 = q_1 \cdot 0 + \epsilon$ — ①

$q_2 = q_1 \cdot 1 + q_2 \cdot 1$ — ②

$q_3 = q_2 \cdot 0 + q_3 (0+1)$ — ③

Consider equ ① $q_1 = \epsilon + q_1 0$

$r = P + rq \Rightarrow r = pq^*$

$q_1 = \epsilon \cdot 0^* = 0^*$ $[\because \epsilon \cdot r = r]$

Sub q_1 in ②

$\underset{r}{q_2} = \underset{P}{0^* \cdot 1} + \underset{rq}{q_2 \cdot 1}$

$q_2 = 0^* \cdot 1 \cdot 1^*$

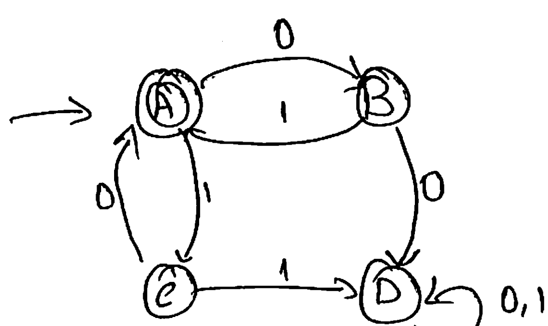
In given automata q_1 & q_2 are final states hence R.E is $q_1 + q_2$

$q_1 + q_2 = 0^* + 0^* \cdot 1 \cdot 1^*$

$= 0^*(\epsilon + 1 \cdot 1^*)$ $\therefore \epsilon + rr^* = r^*$

$= 0^* \cdot 1^*$ //

→ Convert the automata to R.E



$A = C \cdot 0 + B \cdot 1 + \epsilon$ — ①

$B = A \cdot 0$ — ②

$C = A \cdot 1$ — ③

$D = C \cdot 1 + B \cdot 0 + D(0+1)$ — ④

Consider eq - ① by substuting eq ②

$A = C \cdot 0 + A \cdot 0 \cdot 1 + \epsilon$

sub eq ③

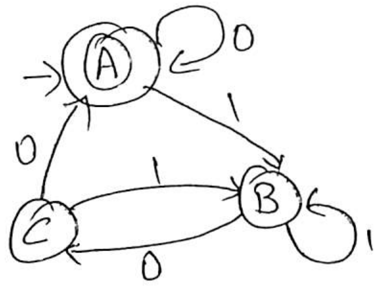
$A = A \cdot 0 \cdot 1 + A \cdot 0 \cdot 1 + \epsilon$

$\Rightarrow \underset{r}{A} = \underset{P}{\epsilon} + \underset{r}{A}(\underset{q}{0 \cdot 1 + 1 \cdot 0})$ $\because r = P + rq$

$\Rightarrow A = \epsilon \cdot (0 \cdot 1 + 1 \cdot 0)^*$ $r = Pq^*$

$\Rightarrow (01 + 01)^*$ $\therefore \epsilon \cdot r^* = r^*$

→



$A = A \cdot 0 + C \cdot 0 + \epsilon$ — ①

$B = A \cdot 1 + B \cdot 1 + C \cdot 1$ — ②

$C = B \cdot 0$ — ③

Sub C in eq ② $B = A \cdot 1 + B \cdot 1 + B \cdot 0 \cdot 1$

$B = A \cdot 1 + B(1 + 0 \cdot 1)$ $r = p + rq$

$\boxed{B = A \cdot 1 \cdot (01 + 1)^*}$

$A = A \cdot 0 + B \cdot 0 \cdot 0 + \epsilon$

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

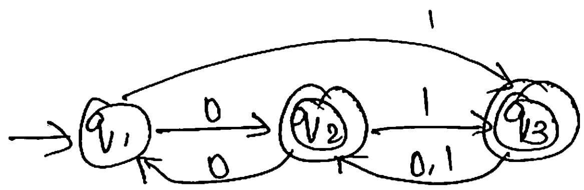
$A = A(0 + 1(1+01)^* 00) + \epsilon$

$A = \epsilon + A(0 + 1(1+01)^* 00)$ $r = p + rq$

$A = \epsilon \cdot (0 + 1(1+01)^* 00)^*$ $= \epsilon \cdot r^* = r^*$

$r = (0 + 1(1+01)^* 00)^*$ //

$\rightarrow$



$q_1 = q_2 \cdot 0 + \epsilon$ —— ①

$q_2 = q_1 \cdot 0 + q_3(0+1)$ —— ②

$q_3 = q_1 \cdot 1 + q_2 \cdot 1$ —— ③

sub eq ① in equ ②

$q_2 = (q_2 \cdot 0 + \epsilon) \cdot 0 + q_3(0+1)$

$= 0 + q_2 \cdot 0 \cdot 0 + q_3(0+1)$ —— ④

sub eq ① in eq ③

$q_3 = (q_2 \cdot 0 + \epsilon) \cdot 1 + q_2 \cdot 1$

$= q_2 \cdot 01 + 1 + q_2 \cdot 1$

$= q_2 \cdot 01 + q_2 \cdot 1 + 1$

$= q_2(01+1) + 1$ —— ⑤

~~R+rq~~

sub ⑤ in ④

$q_2 = 0 + q_2 \cdot 0 \cdot 0 + (q_2(01+1)+1)(0+1)$

$= 0 + q_2 \cdot 00 + q_2(01+1)(0+1) + 1(0+1)$

$q_2 = \underbrace{0+1(0+1)}_{p} + \underbrace{q_2}_{r}(\underbrace{00 + (01+1)(0+1)}_{q})$

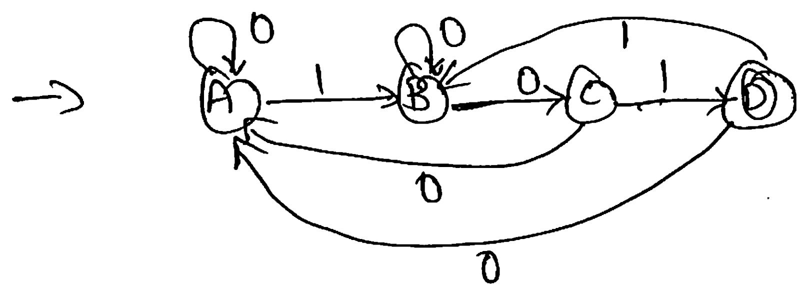
$= 0+1(0+1) \cdot (00 + (01+1)(0+1))^*$

$\gamma = q_2 + q_3$

$= q_2 + q_2(01+1)+1$

$= 1 + q_2(\epsilon + (01+1))$

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



$A = A\cdot0 + C\cdot0 + D\cdot0 + \epsilon$

$B = A\cdot1 + B\cdot0 + D\cdot1$

$C = B\cdot0$

$D = C\cdot1$

Sub C in eq ①

$A = A\cdot0 + B\cdot0\cdot0 + D\cdot0 + \epsilon$

$\underset{\gamma}{A} = (\underbrace{B\cdot00 + D0 + \epsilon}_{P}) + \underset{\gamma}{A}\cdot\underset{q}{0}$

$A = (B\cdot00 + D0 + \epsilon)\cdot0^*$ —— ⑤

Sub eq ⑤ in ②

$B = A\cdot1 + B\cdot0 + D\cdot1$

$= (B\cdot00 + D0 + \epsilon)\cdot0^* + B0 + D1$

$= (B000^* + D00^* + 0^*)1 + B0 + D1$

$= B000^*1 + D00^*1 + 0^*1 + B0 + C11$ sub D

$$= B000^*1 + B00^*1 + 0^*1 + B0 + B011$$

$$= B000^*1 + B0100^*1 + 0^*1 + B0 + B011$$

$$\underbrace{B}_{r} = \underbrace{0^*1}_{P} + \underbrace{B}_{r}(\underbrace{000^*1 + 0100^*1 + 0 + 011}_{q})$$

$$B = 0^*1(000^*1 + 0100^*1 + 0 + 011)^*$$

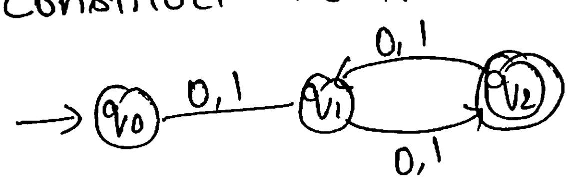
$$D = B01$$

$$D = 0^*1(000^*1 + 0100^*1 + 0 + 011)^*01$$

$$\therefore r = D$$

$$= 0^*1(000^*1 + 0100^*1 + 0 + 011)^*01$$

→ Construct the R.E for following DFA



$$q_0 = \epsilon \quad \text{—— (1)}$$

$$q_1 = q_0(0+1) + q_2(0+1) \quad \text{—— (2)}$$

$$q_2 = q_1(0+1) \quad \text{—— (3)}$$

Sub (1) in (2)

$$q_1 = \epsilon(0+1) + q_2(0+1)$$

$$= (0+1) + q_2(0+1) \quad \text{—— (4)}$$

Sub (4) in (3)

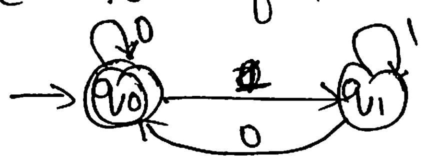
$$q_2 = ((0+1) + q_2(0+1))(0+1)$$

$$\underbrace{q_2}_{r} = \underbrace{(0+1)(0+1)}_{P} + \underbrace{q_2}_{r}\underbrace{(0+1)(0+1)}_{q}$$

$$\Rightarrow q_2 = (0+1)(0+1)\cdot[(0+1)(0+1)]^*$$

$$R\cdot E = (0+1)(0+1)\,[(0+1)(0+1)]^*$$

$\rightarrow$ R·E for following DFA



$$q_0 = q_0 \cdot 0 + q_1 \cdot 0 + \epsilon \quad — ①$$

$$q_1 = q_0 \cdot 1 + q_1 \cdot 1 \quad — ②$$

Consider eq ②

$$\underset{r}{q_1} = \underset{p}{q_0 1} + \underset{r}{q_1}\,\underset{q}{1}$$

$$\Rightarrow q_0 1 \cdot 1^* \qquad \therefore RR^* \Rightarrow R^*$$

$$q_1 \Rightarrow q_0 1^* \quad — ③$$

sub eq ③ in ①

$$q_0 = q_0 0 + q_0 1^* 0 + \epsilon$$

$$\underset{r}{q_0} = \underset{p}{\epsilon} + \underset{r}{q_0}\underset{q}{(0+1^*0)}$$

$$q_0 = \epsilon\cdot(0+1^*0)^* \qquad \therefore \epsilon\cdot R = R$$

$$\therefore R\cdot E = \underline{(0+1^*0)^*}$$

Pumping Lemma for Regular languages:-

Pumming lemma is used to Prove that certain languages are not regular.

→ It is a negative test whether a language is regular or not.

→ Every finite language is a regular language.

→ An infinite language may or may not be regular.

Eg:- L = {ab, abab, ababab, - - - - - }

An infinite language can be represented by using finite automata with the help of loops where (on the loop there should be a pattern by which it generates all strings in language like ab, abab, ababab - - -

If there is no such patterns no finite automata is possible.

Note:- An infinite language which has patterns necessarily need not to be a regular language.

Hence Pumping lemma says that if a lang does not have patterns then it is not a regular language.

Step-1 Assume that language L is regular & n be the no. of states in the corresponding FA.

Step-2: choose a string i.e 'z' such that $|z| \geq n$

Using Pumping lemma we can write $z = uvw$ in such a way that the following conditions are satisfied.

i) $|uv| \leq n$

ii) $|v| \geq 1$

step 3: find a suitable integer 'i' such that uv^iw doesnot belong to L. Hence the language L is not regular.

Eg:- $L = \{a^n b^n \mid n \geq 1\}$

let $L = \{\underset{n=1}{ab}, \underset{n=2}{aabb}, \underset{n=3}{aaabbb}, \ldots\}$

Consider $z = aabb$ in the form of $z = uvw$

i.e $\underset{u}{a}\,\underset{v}{ab}\,\underset{w}{b}$

now verify the cases

Case 1: $|uv| \leq n$ $\therefore n = 4$ - Total length of string

$|aab| \leq 4$

$3 \leq 4$ ✓

Case 2: $|v| \geq 1$

$|ab| \geq 1$

$2 \geq 1$ ✓

Case 3: uv^iw let $i=2$

$\therefore uv^2w \Rightarrow a(ab)^2w$

$\Rightarrow aababb \notin L$

Hence the given language is regular language.

→ P.T the language $L = \{yy \mid y \in \{0,1\}^*\}$ is not regular

if $y = 01$
then yy
$\underline{0101}$

Assume L is regular

$L = \{0101, 0^p10^p1, \ldots\}$

let p is Pumping length

Consider $p = 3$

$L = \underbrace{00}_{u}\underbrace{0}_{v}\underbrace{10001}_{w}$

Case 1 : $|uv| \leq n$ $n = 8$

$3 \leq 8$ ✓

Case 2 : $|v| \geq 1$

$2 \geq 1$ ✓

Case 3 : uv^iw $\therefore i = 2$

uv^2w

$\underbrace{00000}_{4}1\underbrace{0001}_{4}$ as first y is not equal to 2nd y

it is not a regular grammar.

→ let $L = \{a^n \mid n \text{ is even}\}$

i.e $n = 0, 2, 4 \ldots$

$\therefore L = \{a^0, a^2, a^4, \ldots\}$. Arthematic Progression (AP)

as the language has a Pattern $0, 2, 4, \ldots$

it is a Regular language.

→ let $L = \{a^P \mid P \text{ is Prime}\}$

i.e $P = 3, 5, 7, 11, 17$

diff 3 & 5 = 2

diff ~~5 &~~ 7 & 11 - 3

∴ no particular pattern hence, it is not a regular language.

Proof!– let $L = \{aaa, aaaaa, aaaaaaa, - - - - -\}$

Consider $Z = \underbrace{aa}_{u}\underbrace{a}_{v}\underbrace{aa}_{w}$ & length $n = 5$

Case 1: $|uv| \leq n$

$|aaa| \leq 5$

$3 \leq 5$. ✓

Case 2: $|v| \geq 1$

$1 \geq 1$ ✓

Case 3: uv^iw let $i = 3$

$uv^3w \Rightarrow aaaaaaaaa$ = length 9 which is not Present in given language hence it is not a regular language.

→ $L = \{a^n \mid n \text{ is odd}\}$ $n = 1, 3, 5, 7, 9, - - - -\}$ A.P

hence it is a regular language

→ $L = \{a^n \mid n \text{ is } n^{1010}\}$ is R.L

→ S.T the language $L = \{ww \mid w \in (ab)^*\}$

$w = a^p b$ where p is length of Pumping lemma

$L = \{\cancel{a^p b},\ a^p b a^p b\}$

if $p = 2$ aabaab

$z = \underbrace{aa}_{u}\underbrace{ba}_{v}\underbrace{ab}_{w}$ $n = 6$

Case 1: $|uv| \le n$

$4 \le 6$ ✓

Case 2: $|v| \ge 1$

$2 \ge 1$ ✓

Case 3: uv^iw $i = 2$

$\underbrace{aababa}_{w}\underbrace{aab}_{w}$ are not same

hence it is not a regular language.

→ S.T the language $L = \{www \mid w \in (ab)^*\}$

→ S.T the language $L = \{a^n b^n c^n \mid n \ge 0\}$

Closure Properties of Regular sets:-

Let L and M be regular languages. Then the following languages are all regular.

1) Union: For any regular L and M, then $L \cup M$ is regular.
   let $L = L(E)$ & $M = L(F)$ then $L(E+F) = L \cup M$

2) Intersection: If L & M are regular, then $L \cap M$ is also regular
   By Demorgan's law $L \cap M = \overline{\bar{L} \cup \bar{M}}$

3) Difference: If L & M are regular then $L - M$ i.e strings in L but not M.

4) Complement: The Complement of a language 'L' with respect to alphabet 'Σ' such that Σ^* contains L is $\Sigma^* - L$.

5) Reversal: L is a language, L^R is the set of strings whose reversal is in L.
   ex: $L = \{0, 01, 100\}$ $L^R = \{0, 10, 001\}$

6) Homomorphism: If L is a regular language & h is a homomorphism on its alphabet then $h(L) = \{h(w) \mid w \text{ is in } L\}$ is also a regular language

7) Inverse Homomorphism:- let h be a homomorphism & L be a language whose alphabet is the output of Language of h.
   $h^{-1}(L) = \{w \mid h(w) \text{ is in } L\}$

	Regular sets	DCF's	CFL's	CSL's	recursing sets	R.E sets
Union	✓	✗	✓	✓	✓	✓
Concatenation	✓	✗	✓	✓	✓	✓
Kleen Closure	✓	✗	✓	✗	✗	✓
Intersection	✓	✗	✗	✓	✓	✓
Complementation	✓	✓	✗	✓	✓	✗
Homomorphism	✓	✗	✓	✗	✗	✓
Inverse Homomorphism	✓	✓	✓	✓	✓	✓
substitution	✓	✗	✓	✗	✗	✓
Reversal	✓	✗	✓	✓	✓	✓
Intersection Regular sets	✓	✓	✓	✓	✓	✓
Quotient with Regular sets	✓	✓	✓	✗	✗	✓

Grammar formalism: It is denoted as

$G = (V, T, P, S)$

V = finite set of variables (Capital letters)

T = finite set of terminals (small letters)

P = set of Productions

$\alpha \rightarrow \beta \mid \alpha, \beta \in (V \cup T)$

S = Start symbol

Ex: $P: S \rightarrow AS$

$A \rightarrow ABa$ $= (\{A, B\}, \{a, b\}, P, S)$

$B \rightarrow bBb$

Regular Grammar: A grammar is said to be regular if it is either left linear grammar or right linear grammar.

→ let L be a regular grammar then L is regular set.

→ If 'L' is a regular set then L is generated by some left linear grammar or right linear grammar.

left-linear grammar:

let $G = (V, T, P, S)$ be given grammar 'G' said to be a left linear grammar if all productions are of the form

$P: A \rightarrow Bx$ $\quad A, B \in V, \; x \in T^*$

$A \rightarrow x$

Ex: $S \to Sab$
$S \to Aa$ } grammar definition
$A \to a$

string derivations

$S \to Sab$
$S \to Sabab$ $\therefore S \to Sab$
$S \to Aaabab$ $\therefore S \to Aa$
$S \to aaabab$ $\therefore A \to a$

(or)

$S \to Sab$
$S \to Aaab$ $\therefore S \to Aa$
$S \to aaab$ $\therefore A \to a$

Right-linear grammar:-

let $G = (V, T, P, S)$ be given grammar 'G' is said to be a right linear grammar if all the Productions are of the form.

P: $A \to xB$ $\quad A, B \in V,\ x \in T^*$
$A \to x$

Ex: $G: (\{A\}, \{0,1\}, P, S)$

$S \to 10S$
$S \to 1A$
$A \to 01$

string derivation

$S \to 10\underline{S}$
$S \to 1010\underline{S}$ $\therefore S \to 10S$
$S \to 10101\underline{A}$ $\therefore S \to 1A$
$S \to 1010101$ $\therefore A \to 01$

or

$S \to 10\underline{S}$
$S \to 101\underline{A}$ $\therefore S \to 1A$
$S \to 10101$ $\therefore A \to 01$

eg: $G = (\{S\}, \{a,b\}, \{S \to abS \mid a\}, S\}$ & ababa is i/P.

$G = (\{S, S_1, S_2\}, \{a, b\}, P, S\}$
P: $S \to S_1ab$, $S_1 \to S_1ab \mid S_2$, $S_2 \to a$
& i/p string is aabab & abab

Conversion from finite automata to grammar:

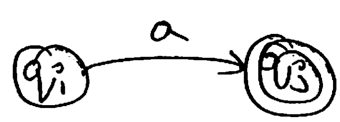
Let L be some language for finite automata $M = \{Q, \Sigma, \delta, q_0, F\}$ then there exits $\exists$ equivalent grammar G such that $L(G) = L(M)$.

→ Q: set of states consider as set of variables V

→ P: Productions are defined as follows

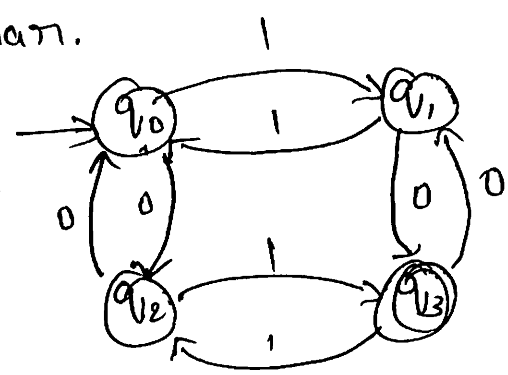
① If there is any transition of the form $\delta(q_i, a) = q_j$ then add Production as $q_i \to a \cdot q_j$ where q_j is non-final state

② If there is a transition $\delta(q_i, a) = q_j$ then add Productions $q_i \to a \cdot q_j$, $q_i \to a$



where q_j is final state

→ Convert the following automata into equivalent grammar.



Productions:

q_0, q_1, q_2, q_3 are variables

α 0, 1 are terminals

P: $q_0 \rightarrow 1 \cdot q_1$

$q_0 \rightarrow 0 \cdot q_2$

$q_1 \rightarrow 0 \cdot q_3$

$q_1 \rightarrow 1 \cdot q_0$ $\qquad G = (\{q_0, q_1, q_2, q_3\}, \{0,1\}, P, q_3)$

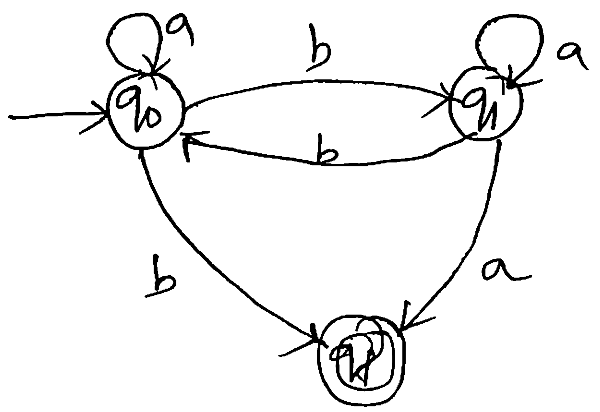
$q_2 \rightarrow 0 \cdot q_0$

$q_2 \rightarrow 1 \cdot q_3$

$q_3 \rightarrow 0 \cdot q_1$

$q_3 \rightarrow 1 \cdot q_2$

2)



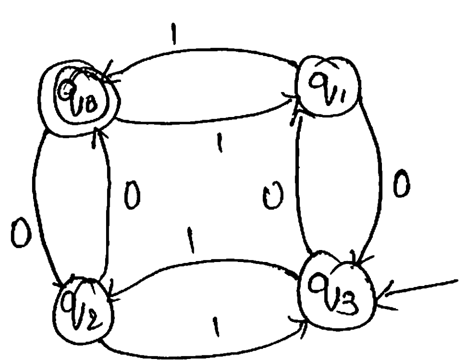
→ To obtain left linear grammar apply any one of the following Procedures:

- Construct the finite automata for given regular expression.
- Now reverse all the transitions obtained by the finite automata & also interchange initial state & final state.
- From the above diagram we can obtain right linear grammar from that we can obtain left linear grammar by interchanging the terminals & variables at the right hand side.

Method 2:

→ Reverse the given regular expression & then construct finite automata equivalent to that reversed regular expression from that we can obtain left linear grammar.

For above example using method 1 obtain left linear grammar interchange initial state q_0 with final state q_3.



equivalent grammar

$q_0 \to 0 \cdot q_2$

$q_0 \to 1 \cdot q_1$

$q_1 \to 0 \cdot q_3$

$q_1 \to 1 \cdot q_0$, $q_1 \to 1$

$q_2 \to 0 \cdot q_0$, $q_2 \to 0$

$q_2 \to 1 \cdot q_3$

$q_3 \to 0 \cdot q_1$

$q_3 \to 1 \cdot q_2$

left linear grammar

$q_0 \to q_2 \cdot 0$

$q_0 \to q_1 \cdot 1$

$q_1 \to q_3 0$

$q_1 \to 1$

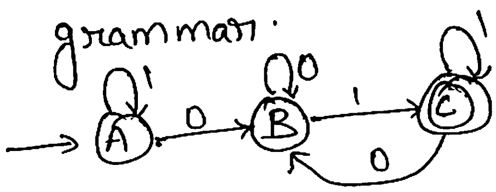
$q_2 \to 0$

$q_2 \to q_3 \cdot 1$

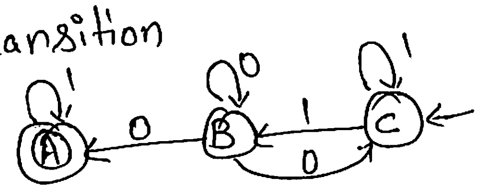
$q_3 \to q_1 \cdot 0$

$q_3 \to q_2 \cdot 1$

→ Convert the following automata into equivalent left linear grammar.



Sol: Convert the initial state to final state & interchange the transition



It is NFA, so convert it into equivalent DFA

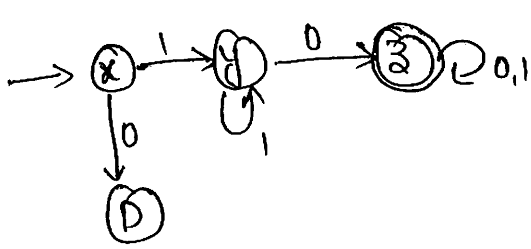
	0	1
→C	$\cancel{\emptyset}$ D	{B,C}
[BC]	[ABC]	[BC]
*[ABC]	[ABC]	[ABC]
D	D	D

for convenience consider

$C = x$

$[BC] = y$

$[ABC] = z$



Right linear grammar	Right linear grammar
$X \rightarrow 1 \cdot Y$	$X \rightarrow Y \cdot 1$
$Y \rightarrow 0 \cdot Z$, $Y \rightarrow 0$	$Y \rightarrow Z \cdot 0$, $Y \rightarrow 0$
$Y \rightarrow 1 \cdot Y$	$Y \rightarrow Y \cdot 1$
$Z \rightarrow 0 \cdot Z$, $Z \rightarrow 0$	$Z \rightarrow Z \cdot 0$, $Z \rightarrow 0$
$Z \rightarrow 1 \cdot Z$, $Z \rightarrow 1$	$Z \rightarrow Z \cdot 1$, $Z \rightarrow 1$

→ **Conversion from regular grammar to finite automata:**

Let $G = (V, T, P, A_0)$ be given grammar where $V = \{A_0, A_1, \ldots A_n\}$

we can construct finite automata equivalent to the given grammar.

1. states of 'M' Corresponds to the variables of given grammar 'G'.
2. Initial states of 'M' Corresponds to the A_0 the grammar 'G'.
3. Input symbols corresponds to the terminals 'T' of the given grammar 'G'.
4. We can define $M = (\{q_0, q_1 \ldots q_n\}, \Sigma, \delta, q_0, q_f)$ and we can define transition function as follows.

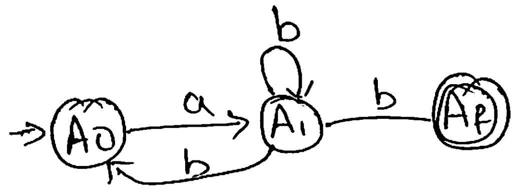
→ If there is a production of the form $A_i \rightarrow a \cdot A_j$ then there is a transition from state q_i to q_j on i/p symbol 'a'. $q_i \xrightarrow{a} q_j$

→ If there is a production of the form $A_i \rightarrow a$ then there is a transition from q_i to final state q_f.

$q_i \xrightarrow{a} q_f$

→ Convert the following grammar into equivalent finite automata.

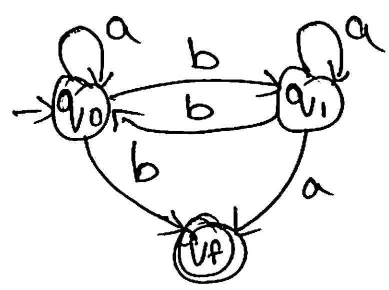
$A_0 \rightarrow a \cdot A_1$
$A_1 \rightarrow b \cdot A_1$
$A_1 \rightarrow b$
$A_1 \rightarrow b \cdot A_0$



→ Convert the following grammar into equivalent F.A.

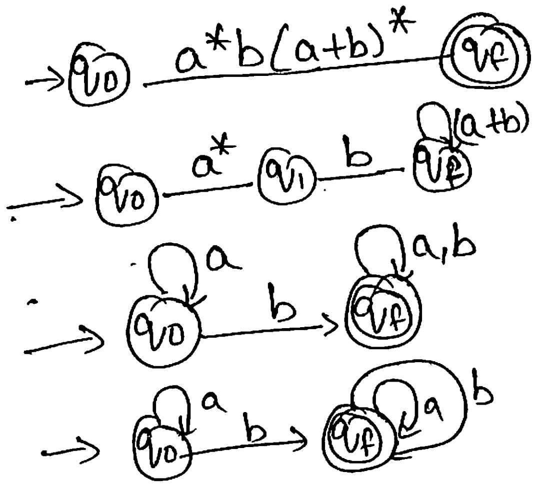
$S \rightarrow aS \mid bA \mid b$
$A \rightarrow aA \mid bS \mid a$

let $S \rightarrow q_0$
$A \rightarrow q_1$



Conversion of Regular Expression to Regular grammar:

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



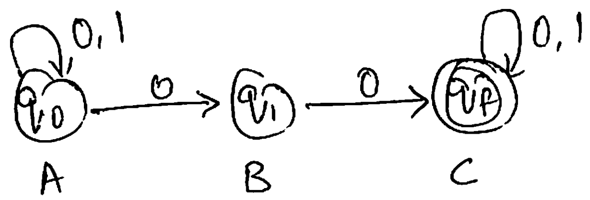
Note: To construct left linear grammar we have to reverse the given expression & construct the FA & we will write linear grammar. We reverse the variables & terminals of to obtain left linear grammar.

let A is a variable equivalent to q_0
B is a variable equivalent to q_f

$A \rightarrow aA$
$A \rightarrow bB$
$A \rightarrow b$

$B \rightarrow aB$
$B \rightarrow a$
$B \rightarrow bB$
$B \rightarrow b$

Ex:- $(0+1)^* 00 (0+1)^*$



Right linear grammar represented by regular expression

$A \to 0A$
$A \to 1A$
$A \to 0B$
$B \to 0 \cdot C$
$B \to 0$
$C \to 0 \cdot C$
$C \to 0$
$C \to 1 \cdot C$
$C \to 1$

for left linear grammar reverse the Expression

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

$A \to 0A$		$A \to A0$
$A \to 1A$		$A \to A1$
$A \to 0B$	L.L.G $\Longrightarrow$	$A \to B0$
$B \to 0C$		$B \to C0$
$B \to 0$		$B \to 0$
$C \to 0C$		$C \to C0$
$C \to 0$		$C \to 0$
$C \to 1 \cdot C$		$C \to C1$
$C \to 1$		$C \to 1$

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