Regular Expressions: (RE)

Regular Expressions are useful for representing certain sets of strings in an algebric fashion.

Regular Expression describes the language accepted by finite automata.

Definition of Regular Expression over Σ.

- Any terminal symbol/element of Σ is Regular Expression

(a) Ex: $\emptyset$ is regular expression and denotes the empty set.

ϵ is a regular expression and denotes the set $\{\epsilon\}$

a is a regular expression and denotes the set $\{a\}$.

(b) Union: Union of two regular expressions R_1 and R_2 is a regular expression R ie $R = R_1 + R_2$

Ex: Let 'a' be a regular expression R_1

'b' be a regular expression R_2

$(a+b)$ is also a regular expression, R, having elements $\{a, b\}$.

(c) Concatenation: Concatenation of two regular expressions R_1 and R_2 is also a regular expression ie $R = R_1 R_2$.

Ex: Let 'a' be a regular expression R_1,

'b' be a regular expression R_2, then

(ab) is also a regular expression R having the element $\{ab\}$.

(d) Iteration (Closure) of a regular expression R is also a regular expression.

Let 'a' be a regular expression

then $\epsilon, a, aa, \cdots$ are also regular expressions.

If 'L' is a language represented by the regular expression R, then the Kleene closure of L is denoted by L^* and is given by

$$L^* = \bigcup_{i=0}^{\infty} L_i$$

the positive closure of L, denoted by L^+, is the set

$$L^+ = \bigcup_{i=1}^{\infty} L_i$$

- If R is a regular expression, then $(R)^*$ is also regular expression.
- Regular expressions over Σ are precisely those expressions obtained recursively by the application of the above rules, once or several times.

Regular sets:

Any set represented by a regular expression is called a regular set.

If a, b are the elements of Σ, then regular expressions

- a denotes the set {a}
- a+b denotes the set {a+b}

ab denotes the set {ab}

a^* denotes the set $\{\epsilon, a, aa, aaa, \ldots\}$

$(a+b)^*$ denotes the set $\{\epsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}$

Regular Set	Regular Expression
$\{10\}$	10
$\{\epsilon, a\}$	$\epsilon + a$
$\{\epsilon, a, aa, ab, ba, b, bb, \cdots\}$	$(a+b)^*$
$\{ab, ba\}$	$ab + ba$
$\{1, 11, 111, 1111, \cdots\}$	1^+

Ex:-

1. All strings of 0's and 1's.

Sol:- The language has the elements of $\{\epsilon, 0, 1, 01, 10, 11, 00, 000, \cdots\}$

Hence the regular expression is $(0+1)^*$.

2. Set of all the strings of 0's and 1's ending 00.

Sol:- The language has the elements of $\{00, 000, 100, 0000, 1000, 0100, 1100, \cdots\}$

This can be written as $\{\epsilon, 0, 1, 01, 10, 11, 000, \cdots\}00$.

Hence the regular expression is $(0+1)^*00$

3. Set of all the strings of 0's and 1's beginning with 0 and ending with 1.

Sol:- The language has the elements of $\{01, 001, 011, 0001, 0011, 0101, 0111, 00001, \cdots\}$

This can be written as $0\{\epsilon, 0, 1, 01, 10, 11, 000, \cdots\}1$.

Hence the regular expression is $0(0+1)^*1$.

4.

4. Set of all the strings having even number of 1's.

Sol: The language has the elements (or) even number of elements $= \{\in, 11, 1111, \ldots\}$

Hence the regular expression $(11)^*$

5. Set of all the strings having odd number of 1's.

Sol:- the language has the odd elements of $\{1, 111, 11111, 1111111, \ldots\}$

This can be written as $\{\in, 11, 1111, 111111, \ldots\}1$.

Hence the regular expression $1(11)^*$ or $(11)^*1$.

6. Strings of 0's and 1's with atleast two consecutive 0's.

Sol:- The language has the elements of $\{00, 000, 001, 100, 0000, 0001, 1000, 0010, 0011, 0100, 1100, \ldots\}$.

These strings can be generated by the regular expression

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

7. All the strings of 0's and 1's beginning with 1 or 0 and not having two consecutive 0's.

Sol:- Regular expression for strings with 0's and 1's and that do not have two consecutive 0's is $(1+10)^*$

Since the string may either start with 0 or 1 then expression can be written as $(0+\varepsilon)(1+10)^*$.

8. set of all the strings ending with 011.

Sol:- The language has the elements of $\{011, 0011, 1011, 00011, 01011, 10011, 11011, \ldots\}$.

This can be written as $\{\in, 0, 1, 00, 01, 10, \ldots\}011$.

Hence, the regular expression is $(0+1)^*011$

Exclude bibliography On

9. Set of all the strings with 0's followed by 1's, followed by 2's such that it has at least one 0 followed by at least one 1 followed by at least one 2.

Sol:- The language has the elements of {012, 0012, 0112, 0122, 00012, 00112, 01112, 00122, ...}

Hence the regular expression is $0^+1^+2^+$.

10. Set of all the strings of 0's and 1's whose last two symbols are same.

Sol:- The language has the elements of {00, 11, 011, 000, 0000, 0011, 1000, 1011, 1111, ...}

Hence the regular expression is $(0+1)^*(00+11)$.

11. String of 0's and 1's with a substring 1100.

Sol:- The language has the elements {1100, 11000, 11001, 01100, 11100, 011001, 011000, ...}

RE is $(0+1)^*1100(0+1)^*$

12. set of all three-lettered words, starting with b over {a, b}

Sol:- elements = {baa, bab, bba, bbb}

RE = R = baa + bab + bba + bbb

13. Set of all the words starting with a and ending with b.

elements = {ab, aab, abb, aaab, aabb, ...}

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

Identity Rules for Regular Expressions

P and Q are two equivalent regular expressions. i.e P and Q represent the same set of strings. Then, to simplify the regular expressions, the following identity rules can be used.

(1) $\emptyset + R = R$

(2) $\epsilon + R = R + \epsilon$

(3) $\epsilon R = R\epsilon = R$

(4) $R + R = R$

(5) $RR^* = R^*R = R^+$

(6) $\epsilon + RR^* = R^* = \epsilon + R^*R$

(7) $\emptyset R = R\emptyset = \emptyset$

(8) $\epsilon^* = \epsilon$, and $\emptyset^* = \epsilon$

(9) $(R^*)^* = R^*$

(10) $(PQ^*)P = P(QP)^*$

(11) $(P+Q)R = PR + QR$ and

(12) $R(P+Q) = RP + RQ$

(13) $(P+Q)^* = (P^*Q^*)^* = (P^* + Q^*)^*$

ARDEN's Theorem :

If P and Q are two regular expressions over Σ, and if P does not contain ϵ, then the following equation in R given by $R = Q + RP$ has unique solution i.e $R = QP^*$

$R = Q + RP$ —①

Substitute $R = QP^*$ in equation ①, then

$R = Q + QP^*P$

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

$R = QP^*$ ∴ so, it is a solution of the given equation.

$R = Q + RP$ —②

replace $R = Q + RP$ in equation ②, then

$R = Q + (Q+RP)P$

$= Q + QP + RP^2$

$= Q + QP + (Q+RP)P^2$ $[\because R = Q + RP]$

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

$= Q + QP + QP^2 + [Q+RP]P^3$ $[\because R = Q + RP]$

$= Q + QP + QP^2 + QP^3 + RP^4$

$= Q + QP + QP^2 + QP^3 + \cdots + QP^n + RP^{n+1}$ ~~replace $R = QP^*$~~

replace $R = QP^*$

$= Q + QP + QP^2 + QP^3 + \cdots + QP^n + QP^*P^{n+1}$

$= Q[\epsilon + P + P^2 + P^3 + \cdots P^n + P^*]$

$R = QP^*$ proved

Ex: Prove that $(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$ equal to $0^*1(0+10^*1)^*$.

Sol:

$$\text{LHS} = (1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$$

$$= (1+00^*1)[\epsilon + (0+10^*1)^*(0+10^*1)] \quad \because \epsilon + R^*R = R^*$$

$$= (1+00^*1)[(0+10^*1)^*] \quad \epsilon.R = R$$

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

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

$$= 0^*1(0+10^*1)^* = \text{RHS}$$

Ex: (a) Given an Regular expression for representing the set L of strings in which every 0 is immediately followed by at least two 1's.

(b) Prove that the regular expression $R = \epsilon + 1^*(011)^*(1^*(011)^*)^*$ also describes the same set of strings.

Sol:-

(a) If w is in L, then (i) w does not contain any 0. or ii) it contains a '0' preceded by 1 and followed by 11.

So w can be written as $w_1 w_2 \cdots w_n$ where each w_i is either 1 or 011. So L is represented by the regular expression as $(1+011)^*$

$R = \epsilon + 1^*(011)^*(1^*(011)^*)^*$

$= \epsilon + P_1 P_1^*$

$= P_1^*$

$= [1^*(011)^*]^*$

$= (P_2^* P_3^*)^*$

$= P_2^* + P_3^*$

$= (1^* + 011)^*$

$R = (1^* + 011)^*$

So R

$P_1 = 1^*(011)^*$

$[\because P_1^* = \epsilon + P_1 P_1^*]$

$[\because P_2 = 1, \; P_3 = 011$

$\because (P^* Q^*)^* = P^* + Q^* \;] = (P+Q)^*$

Manipulations of Regular Expressions:

Let R and S be any two regular expressions.

(i) Concatenation - RS denoting the set $\{xy \mid x \in R \text{ and } y \in S\}$.

Ex: if $R = \{ab, c\}$, and $S = \{d, ef\}$

then $RS = \{abd, abef, cd, cef\}$.

(ii) Union : The union R∪S denotes the set union of R and S.

Ex: $R = \{ab, c\}$ and $S = \{d, ef\}$

then $R \cup S = \{ab, c\} \cup \{d, ef\}$

$= \{ab, c, d, ef\}$

(iii) Kleene closure or Star closure - R^* denotes the smallest superset of R that contains ε and is closed under string concatenation. This is the set of all strings, that can be made by concatenating zero or more strings in R.

Ex: $\{ab, c\}^* = \{\varepsilon, ab, c, abab, cab, cc, ababab, \ldots\}$

$\{01, 2\}^* = \{\varepsilon, 01, 2, 012, 0101, 201, 012012, \ldots\}$

Ex: form the string sets for the regular expression given below ⑥

(a) 1^*0.

Sol: $1^*0 = \{\epsilon, 1, 11, 111, \cdots\}.0$

$= \{0, 10, 110, 1110, \cdots\}$.

(b) 00^*

Sol: $00^* = 0.\{\epsilon, 0, 00, 000, \cdots\}$

$= \{0, 00, 000, 0000, \cdots\}$

(c) 10^*1

Sol: $10^*1 = 1.\{\epsilon, 0, 00, 000, \cdots\}.1$

$= \{11, 101, 1001, 10001, \cdots\}$

(d) $(100^+)^*$

Sol: $(100^+)^* = (10.(0^+))^*$

$= (10.\{0, 00, 000, \cdots\})^*$

$= \{100, 1000, 10000, \cdots\}^*$

$= \{\epsilon, 100, 1001000, 100100, 10010000, \cdots\}$

(e). a^*b^*

Sol: $a^*b^* = \{\epsilon, a, aa, aaa, \cdots\}.\{\epsilon, b, bb, bbb, \cdots\}$

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

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

Sol: $(0+1)^* = 0^* \cup 1^*$

$= \{\epsilon, 0, 00, 000, \cdots\} \cup \{\epsilon, 1, 11, 111, \cdots\}$

$= \{\epsilon, 0, 00, 1, 11, \cdots\}$

(h) $(1+10)^*$

Sol $(1+10)^* = 1^* \cup (10)^*$

$= \{\epsilon, 1, 11, 111, \cdots\} \cup \{\epsilon, 10, 1010, \cdots\}$

$= \{\epsilon, 1, 11, 111, 10, 1010, \cdots\}$

(i) $(0+1)^* 011$

Sol: $(0+1)^* 011 = (0 \cup 1)^* . 011$

$= (0^* 011) \cup (1^* 011)$

$= \{\epsilon, 0, 00, \cdots\} 011 \cup \{\epsilon, 1, 11, 111, \cdots\} 011$

$= \{011, 00011, 000011 \cdots\} \cup \{1011, 11011, 111011, \cdots\}$

$= \{011, 00011, 00011, 1011, 11011, 111011, \cdots\}$

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

Sol: $(0^* + 1^*)^* = (0^* \cup 1^*)^*$

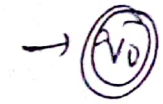
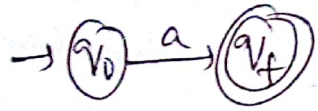
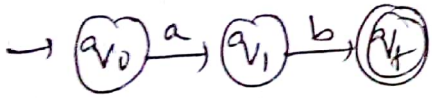
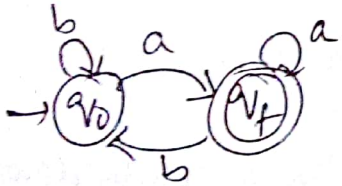
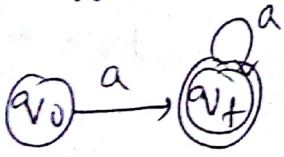
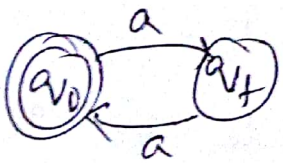
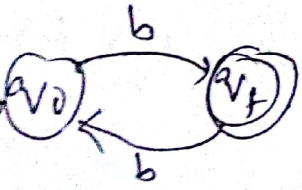
$= (\{\epsilon, 0, 00, 000, \cdots\} \cup \{\epsilon, 1, 11, 111, \cdots\})^*$

$= \{\epsilon, 0, 00, 1, 11, \cdots\}^*$

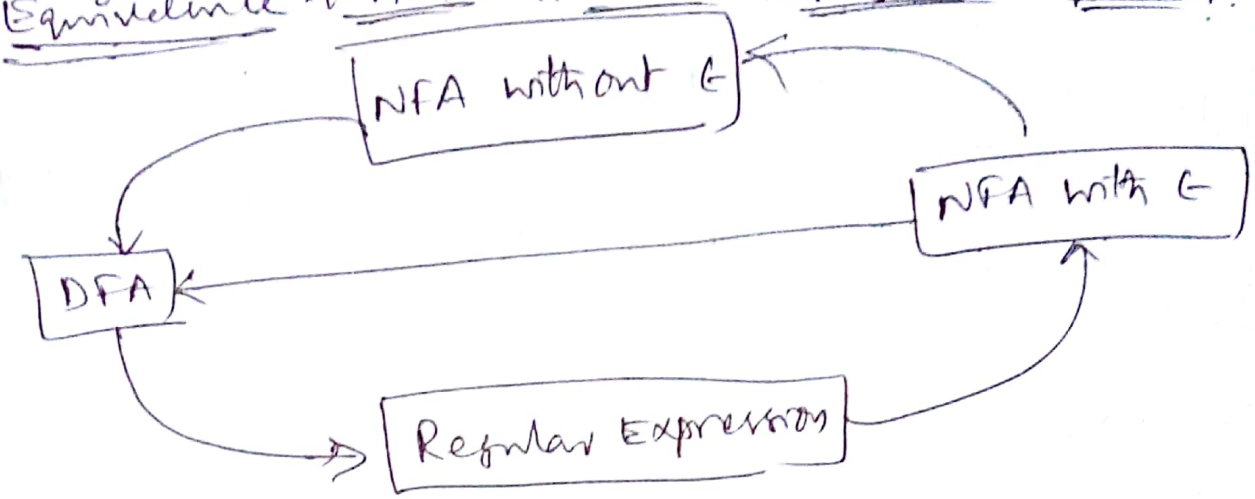
$= \{0^*, (00)^*, (1)^*, (11)^*, \cdots\}$

$=$

Relation between F.A, RS & RE.

Finite Automata	Regular set	Regular Expression
(i) 	ϕ	ϕ
(ii) 	$\{\epsilon\}$	ϵ
(iii) 	$\{a\}$	a
(iv) 	$\{a, b\}$	$a+b$, $a \mid b$
(v) 	$\{ab\}$	ab
(vi) 	$\{a, ba, baa, aa, aba, baba \ldots\}$	$b^*a(a^*bb^*a)^*$ $(b^*aa^*b)^*b^*a$
(vii) 	$\{\epsilon, a, aa, aaa, \ldots\}$	a^*
(viii) 	$\{a, aa, aaa, \ldots\}$	a^+
(ix) 	$\{\epsilon, aa, aaaa, \ldots\}$	$(aa)^*$
(x) 	$\{b, bbb, bbbbb, \ldots\}$	$b(bb)^*$ $=$

Equivalence of finite Automata & Regular Expressions:



Equivalence of FA & RE

Kleene's theorem:

(i) Every language accepted by a finite automaton can be defined by a regular expression.

(ii) for every regular expression, there is an equivalent NFA with ∈-transition.

Equivalence of DFA and Regular Expressions:

Step 1: for each of the states $q_1, q_2 \ldots q_n$ in DFA, write down the equations by considering all edges, that enter into that state.

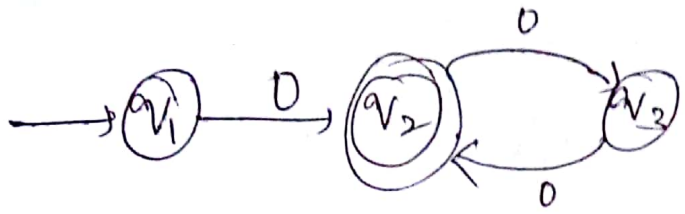
Step 2: for the initial state of DFA, the equation is added with ∈.

Step 3: Compute the equations for each state.

Step 4: Substitute the results of each state equation into the final state equations of DFA, to get a RE for DFA.

Ex:- Construct RE for the ~~DFA~~ following DFA.

$R = Q + RP$
$R = QP^*$



Sol:-

① $q_1 = \epsilon$ —— ①

② $q_2 = q_1 0 + q_3 0$ —— ②

③ $q_3 = q_2 0$ —— ③

Final state is q_2, so compute q_2.

$q_2 = q_1 0 + q_3 0$

Substitute eqn ① & ③ in eqn ②, then

$q_2 = \epsilon.0 + q_2 0.0$

$R \quad Q \quad R \quad P$

$R = Q + RP \Rightarrow R = QP^*$

$\Rightarrow q_2 = \epsilon 0 (00)^*$

$\Rightarrow RE = 0(00)^*$

=

Construct a RE for following DFA

Sol:

$$q_1 = \epsilon + q_1 0 \quad -(1)$$

$$q_2 = q_1 1 + q_2 1 \quad -(2)$$

$$q_3 = q_2 0 + q_3 0 + q_3 1 \quad -(3)$$

final states are q_1 and q_2.

$q_1 = \epsilon + q_1 0$
R Q R P

$R = Q + RP$
$\Rightarrow R = Qp^*$

then $q_1 = \epsilon 0^*$

RE i.e. $q_1 = 0^* \quad -(4)$ [$\because \epsilon . R = R$] Identity

$q_2 = q_1 1 + q_2 1$.

Substitute eqn (4) in eqn (2), then

$q_2 = 0^* 1 + q_2 1$
R Q R P

$R = Q + RP$ then
$R = QP^*$

RE: $q_2 = (0^* 1)(1)^* \quad -(5)$

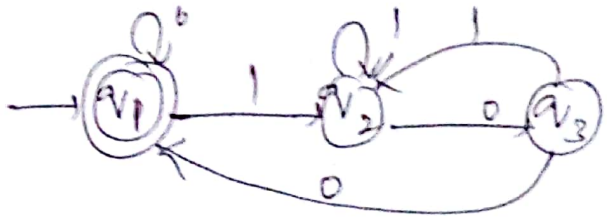
There are two final states, ie q_1 and q_2, then the regular expression is union of these two states. ie $q_1 \cup q_2$

Regular expression $q_1 + q_2 = 0^* + 0^* 1 1^*$
$= 0^* (\epsilon + 11^*)$
$= 0^* 1^*$.

$\because \epsilon + RR^* = R^*$

So, the final RE $= 0^* 1^*$

③ Construct a regular expression for the following DFA.



Sol: Equations for each states

$$q_1 = q_1 0 + q_3 0 + \epsilon \quad -(1) \quad (\epsilon \text{ only for initial state})$$

$$q_2 = q_1 1 + q_2 1 + q_3 1 \quad -(2)$$

$$q_3 = q_2 0 \quad -(3)$$

Substitute eqn (3) in eqn (2), then

$$q_2 = q_1 1 + q_2 1 + q_2 0 1$$

$$q_2 = \underbrace{q_1}_{R}\underbrace{1}_{Q} + \underbrace{q_2}_{R}\underbrace{(1+01)}_{P}$$

$R = Q + RP$

$\Rightarrow R = QP^*$

$$q_2 = q_1 1 (1+01)^* \quad -(4)$$

Now find state of q_1

$$q_1 = q_1 0 + q_3 0 + \epsilon$$

$$= q_1 0 + q_2 00 + \epsilon$$

$$= q_1 0 + q_1 1 (1+01)^* 00 + \epsilon$$

$$= q_1 (0 + 1(1+01)^* 00) + \epsilon$$

$$\underbrace{q_1}_{R} = \underbrace{\epsilon}_{Q} + \underbrace{q_1}_{R}\underbrace{(0+1(1+01)^* 00)}_{P}$$

Then

$\epsilon . R = R$

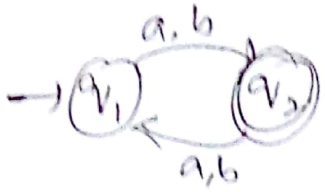
$$q_1 = \epsilon (0+1(1+01)^* 00)^*$$

$$q_1 = (0+1(1+01)^* 00)^*$$

R.E is

Since q_1 is the final state, then RE is $(0+1(1+01)^* 00)^*$.

Ex: Construct a RE for the given transition diagram.



Sol: Equations

$q_1 = q_2a + q_2b + \epsilon$ — (1) $\Rightarrow q_1 = q_2(a+b) + \epsilon$

$q_2 = q_1a + q_1b$ — (2) $q_2 = q_1(a+b)$

Here final state is q_2.

Computing q_2 state

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

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

$q_2 = \underbrace{q_2}_{R} = \underbrace{\epsilon}_{Q} + \underbrace{q_2}_{R}\underbrace{(a+b)(a+b)}_{P}$ $\quad R = Q + RP$

$\Rightarrow R = QP^*$

$q_2 = \epsilon((a+b)(a+b))^*$ $\quad q_2 = (a+b)((a+b)(a+b))^*$

$q_2 = ((a+b)(a+b))^*$

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

Ex: 

Construct regular expressions for the given DFA.

Sol: Equations are

$q_0 = q_01 + q_10 + \epsilon$ — (1)

$q_1 = q_00 + q_10$ — (2)

Equation (2) is the final state, then

$q_1 = \underbrace{q_0}_{Q}\underbrace{0}_{P} + \underbrace{q_1}_{R}\underbrace{0}_{P}$ $\quad R = Q + RP$

$\Rightarrow R = QP^*$

$q_1 = q_00(0)^*$ — (3)

Substitute eqn (3) in eqn (1), then

$$q_0 = q_0 1 + q_1 1 + \epsilon$$
$$= q_0 1 + q_0 0(0)^* 1 + \epsilon$$

$$q_0 = \underset{R}{\epsilon} + \underset{Q}{\epsilon}\,\underset{R}{q_0}\underbrace{[1 + 0(0)^* 1]}_{P}$$

$R = Q + RP$
$\Rightarrow R = QP^*$

$$q_0 = \epsilon.[1 + 0(0)^* 1]^* = (1 + 00^* 1)^*$$
$$= \epsilon[1(\epsilon + 0(0)^*)]^*$$
$$q_0 = [10^*]^* \quad \text{— (4)}$$

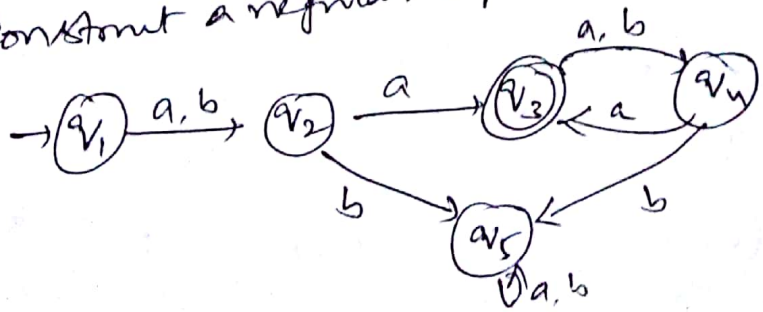
$[\because \epsilon + RR^* = R^*$
$\epsilon + 00^* = 0^*]$

Substitute eqn (4) in eqn (3), q_1 is final state, then

$$q_1 = q_0 0(0)^*$$
$$q_1 = (10^*)^* 00^*$$

Then the RE is $(10^*)^* 00^*$

Ex:- Construct a regular expression for the DFA



Sol:-

$$q_1 = \epsilon \quad \text{— (1)}$$
$$q_2 = q_1 a + q_4 b \quad \text{— (2)} \quad = q_1(a+b)$$
$$q_3 = q_2 a + q_4 a \quad \text{— (3)}$$
$$q_4 = q_3(a+b) \quad \text{— (4)}$$
$$q_5 = q_2 b + q_4 b + q_5(a+b) \quad \text{— (5)}$$

Since $q_1 = \epsilon$,

$q_2 = \epsilon(a+b)$

final state is q_3, then Compute q_3,

$q_3 = q_2 a + q_4 a$ $\quad [\because q_4 = q_3(a+b)]$

$q_3 = (a+b)a + q_3(a+b).a$

$R \quad Q \quad R \quad P$

$R = Q + RP$

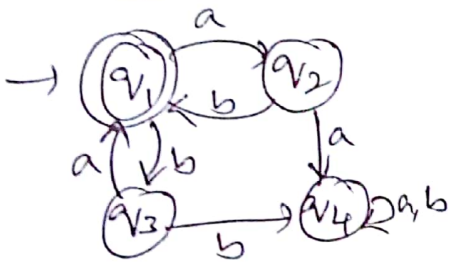
$\Rightarrow R = Qp^*$

$q_3 = [(a+b)a][(a+b)a]^*$

then $RE = [(a+b)a][(a+b)a]^*$

$=$

Ex: DFA to RE



Sol: $q_1 = q_2 b + q_3 a + \epsilon$ — (1)

$q_2 = q_1 a$ — (2)

$q_3 = q_1 b$ — (3)

$q_4 = q_3 b + q_2 a + q_4 a + q_4 b$ — (4)

final state is q_1

then $q_1 = q_2 b + q_3 a + \epsilon$

$= q_1 ab + q_1 ba + \epsilon$

$q_1 = \epsilon + q_1(ab+ba)$

$R \quad Q \quad R \quad P$

$R = Q + RP \Rightarrow R = Qp^*$

$q_1 = \epsilon(ab+ba)^*$

$q_1 = (ab+ba)^*$

So, The required RE is $(ab+ba)^*$

$=$

Conversion of RE to FA

There are 3 cases that we need to apply to construct NFA for the given Regular Expression.

Case 1: If r_1 and r_2 are two regular expressions, then the NFA equivalent to "$r = r_1 + r_2$" is as given below.

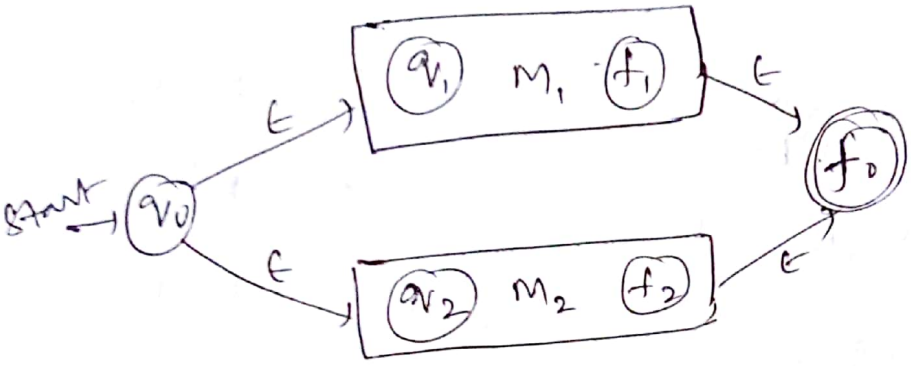


fig: NFA for $r = r_1 + r_2$.

Case 2: If r_1 and r_2 are two regular expressions, then the NFA equivalent to "$r = r_1 r_2$" is shown in below fig

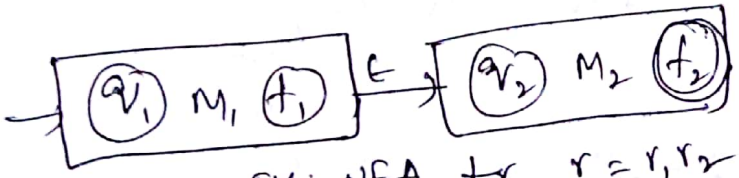


fig: NFA for $r = r_1 r_2$

Case 3: If r_1 ~~and r_2~~ are is a regular expression, then the equivalent NFA to the regular expression "$r = r_1^*$" is shown in below fig

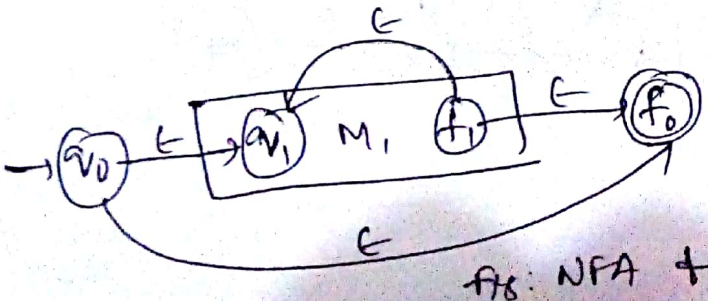


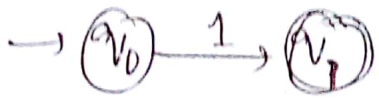
fig: NFA for $r = r_1^*$

Ex: Construct NFA for the regular expression $(1+0)0^*$

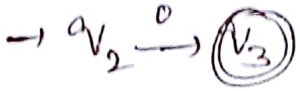
Sol:- The given expression can be expressed as a regular expression formed by applying a sequence of operations on individual expressions as shown below

$R = (R_1 + R_2) R_3^*$ where $R_1 = 1$, $R_2 = 0$, $R_3 = 0$.

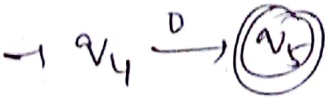
NFA for R_1 is $(R_1 = 1)$



NFA for R_2 is $(R_2 = 0)$



NFA for R_3 is $(R_3 = 0)$

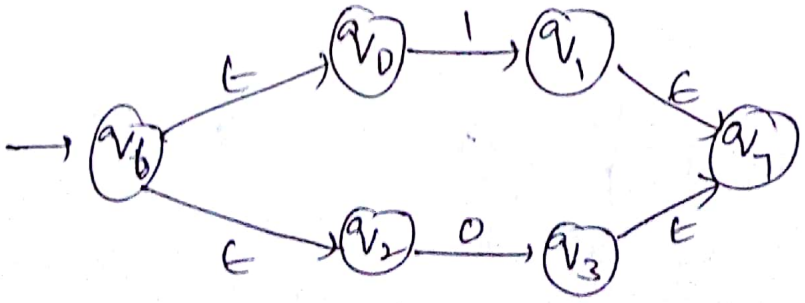


$R_4 = R_1 + R_2$

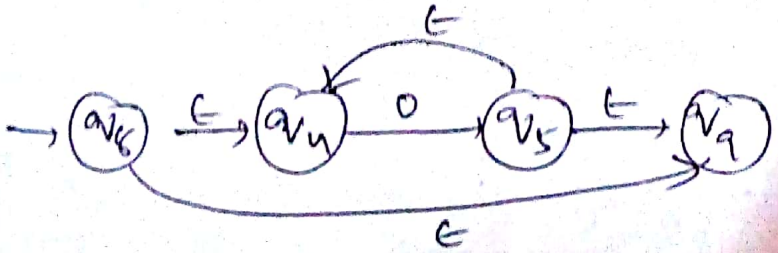
$R_5 = R_3^*$

$R_6 = R_4 \cdot R_5$

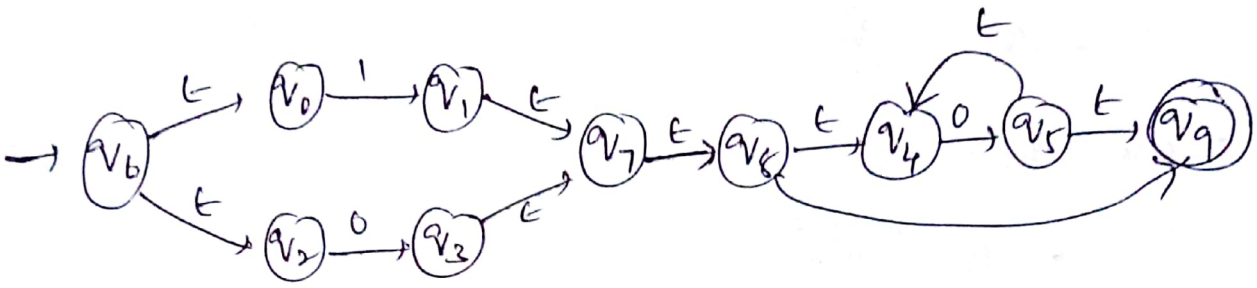
Applying Union rules for $R_4 = R_1 + R_2$.



Apply closure rules for $R_5 = R_3^*$



Apply Concatenation rule for $R_6 = R_4 . R_5$.



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

=.

Ex:- Construct an equivalent NFA with ϵ- transition for the following Regular expression $r = 0^* + 11$.

Sol:- The Components of the given regular expression 'r' are

$r = 0^* + 11$

$r_1 = 0$

$r_2 = r_1^*$

$r_3 = 1$

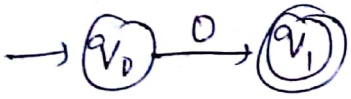
$r_4 = 1$

$r_5 = r_3 r_4$

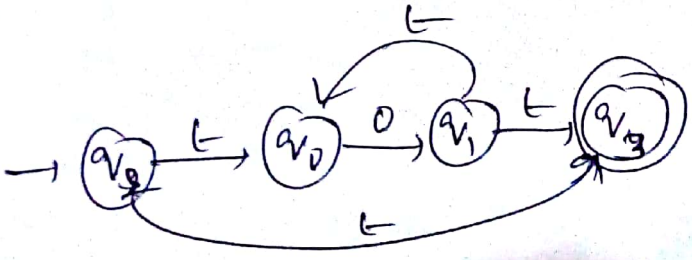
$r_6 = r_2 + r_5$

$r = r_6$

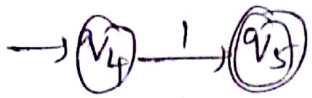
Step1: Let $r_1 = 0$



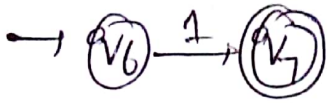
Step2: $r_2 = r_1^*$



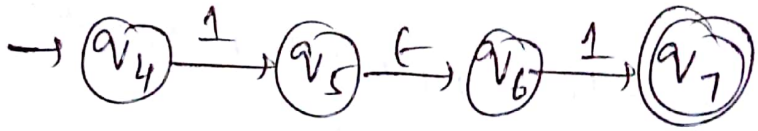
Step 3: $r_3 = 1$



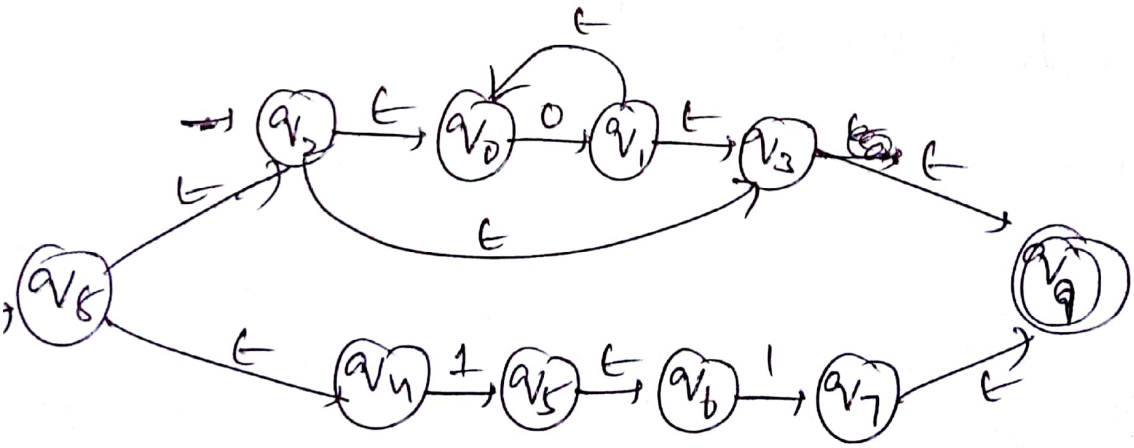
Step 4: $r_4 = 1$



Step 5: $r_5 = r_3 r_4$



Step 6: $r_6 = r_2 + r_5$



FA for $0^* + 11$

Construct NFA with ε-moves for $00^* + 1$.

Sol:

$$r = 00^* + 1$$

$r_1 = 0$

$r_2 = 0$

$r_3 = r_2^*$

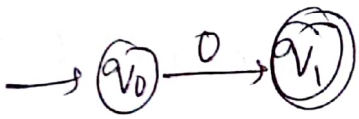
$r_4 = r_1 r_3$

$r_5 = 1$

$r_6 = r_4 + r_5$.

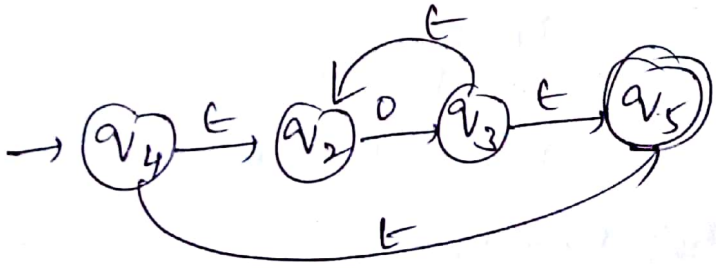
Then

Step 1: Let $r_1 = 0$

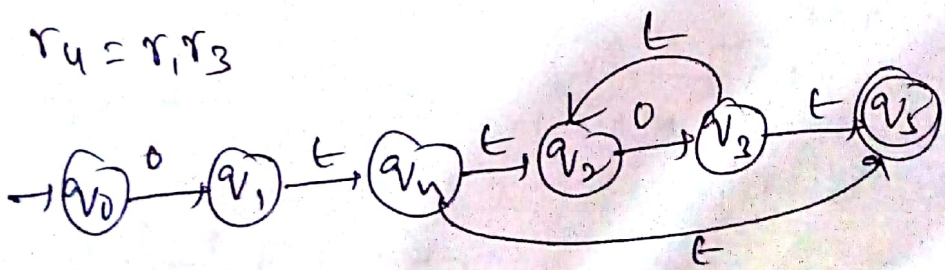


Step 2: $r_2 = 0$

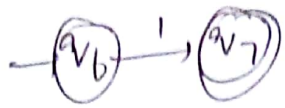
Step 3: $r_3 = r_2^*$



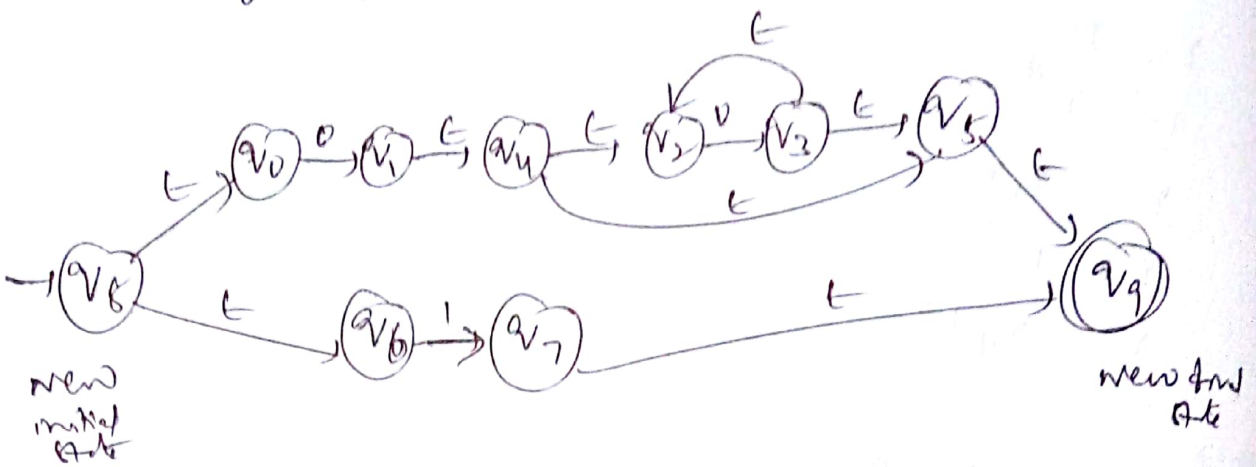
Step 4: $r_4 = r_1 r_3$



Step 5: $r_5 = 1$



Step 6: $r_6 = r_4 + r_5$



NFA-ϵ for

Ex: Convert the following RE into NFA with ϵ-transitions.

i) $1^*0 + 1101$

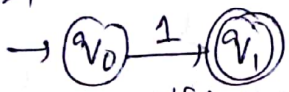
Sol:- $r = 1^*0 + 1101$

$r_1 = 1$
$r_2 = r_1^*$
$r_3 = 0$
$r_4 = r_2 r_3$

$r_5 = 1$
$r_6 = 1$
$r_7 = 0$
$r_8 = 1$
$r_9 = r_5 r_6 r_7 r_8$

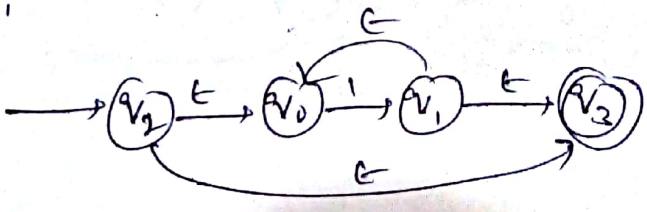
$r_{10} = r_4 + r_9$

Step 1: $r_1 = 1$



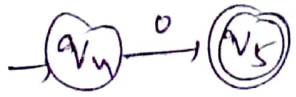
NFA-ϵ for $r_1 = 1$

Step 2: $r_2 = r_1^*$



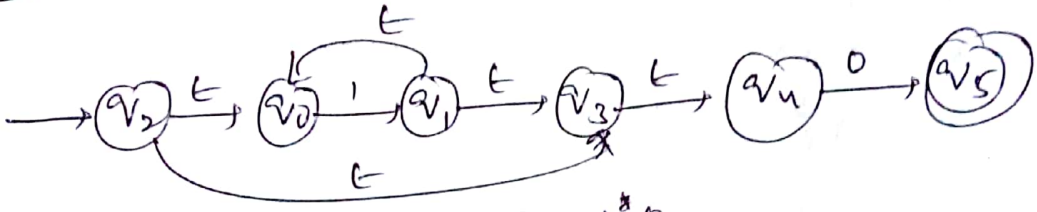
NFA-ϵ for $r_2 = r_1^*$

Step 3: $r_3 = 0$



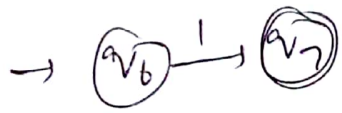
NFA-ε for $r_3 = 0$

Step 4: $r_4 = r_2 r_3$



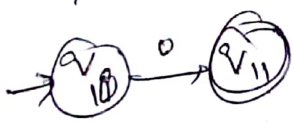
NFA-ε for $r_2 r_3 = 1^*0$

Step 5: $r_5 = 1$

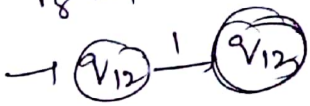


Step 6: $r_6 = 1$

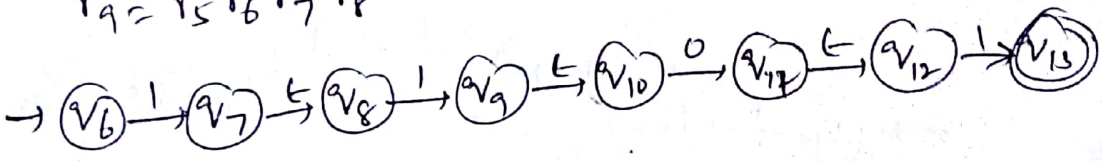
Step 7: $r_7 = 0$



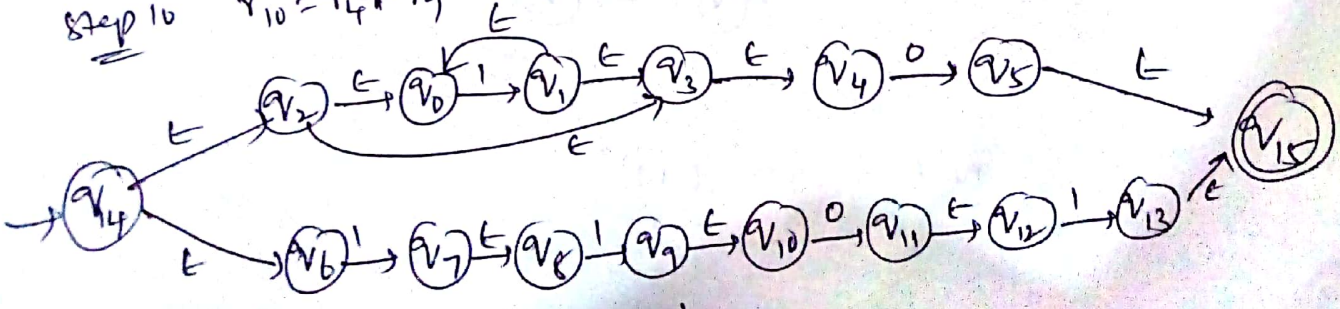
Step 8: $r_8 = 1$



Step 9: $r_9 = r_5 r_6 r_7 r_8$



Step 10 $r_{10} = r_4 + r_9$



NFA-ε for $1^*0 + 1101$.

Ex:- Convert following R.E to NFA with ϵ.

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

Sol. $r = (0+1)^*$

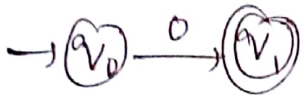
$r_1 = 0$

$r_2 = 1$

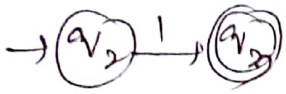
$r_3 = r_1 + r_2$

$r_4 = r_3^*$

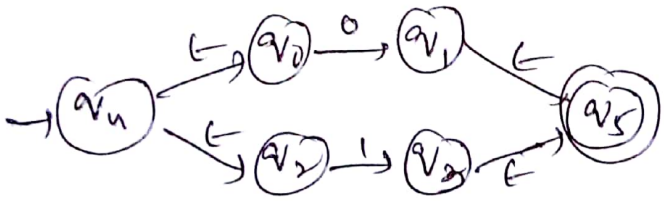
Step 1: $r_1 = 0$



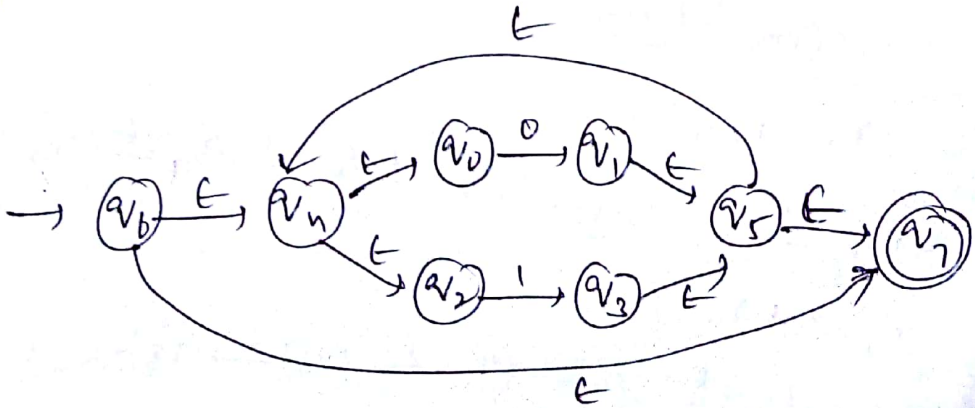
Step 2: $r_2 = 1$



Step 3: $r_3 = r_1 + r_2$



Step 4: $r_4 = r_3^*$



NFA-ϵ for $(0+1)^*$

Ex:- Construct a NFA-ε for the RE $(10+11)^*00$.

Sol Given RE is $r = (10+11)^*00$

$r_1 = 1$

$r_2 = 0$

$r_3 = r_1 \cdot r_2$

$r_4 = 1$

$r_5 = 1$

$r_6 = r_4 r_5$

$r_7 = r_3 + r_6$

$r_8 = r_7^*$

$r_9 = 0$

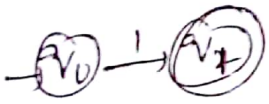
$r_{10} = 0$

$r_{11} = r_9 r_{10}$

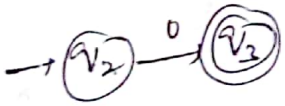
$r_{12} = r_8 r_{11}$.

Construction of NFA-ε moves

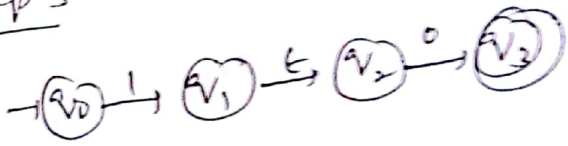
Step 1: $r_1 = 1$



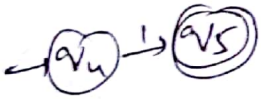
Step 2: $r_2 = 0$



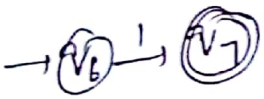
Step 3: $r_3 = r_1 r_2$



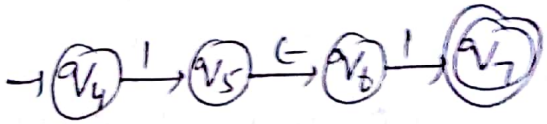
Step 4: $r_4 = 1$



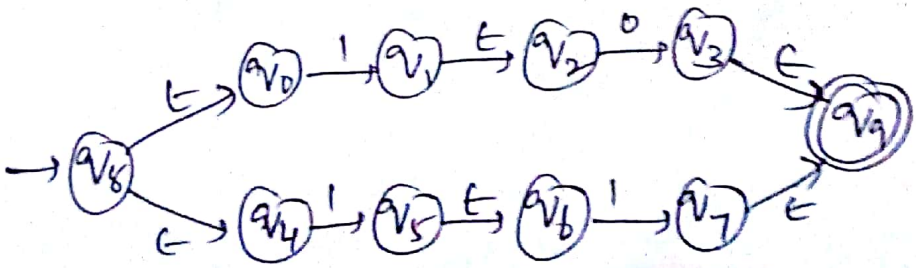
Step 5: $r_5 = 1$



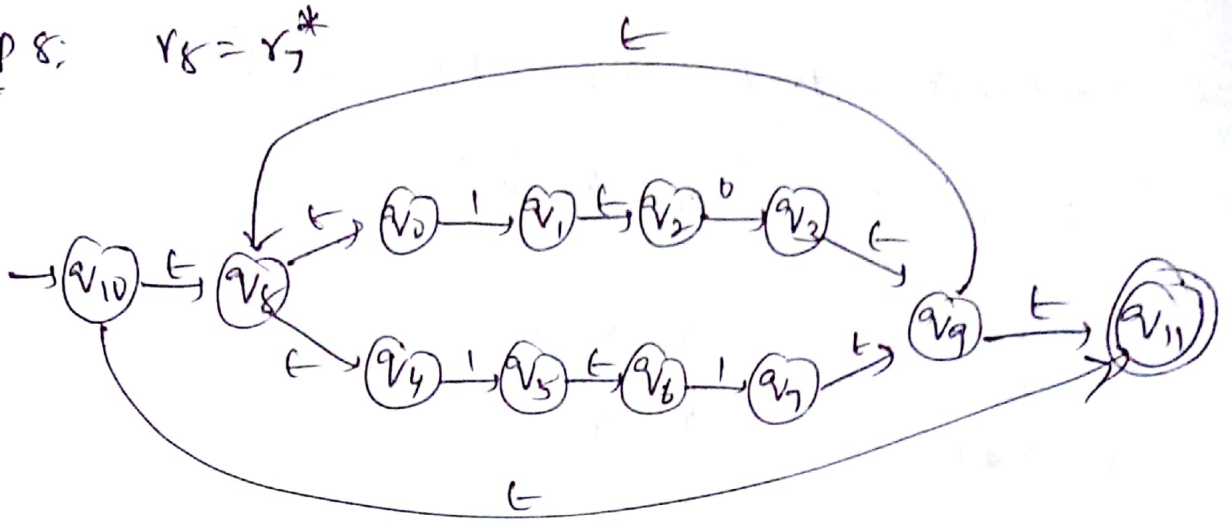
Step 6: $r_6 = r_4 r_5$



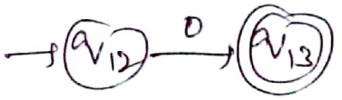
Step 7: $r_7 = r_3 + r_6$



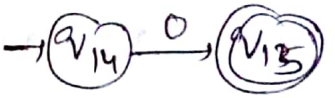
Step 8: $r_8 = r_7^*$



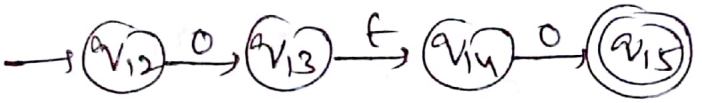
Step 9: $r_9 = 0$



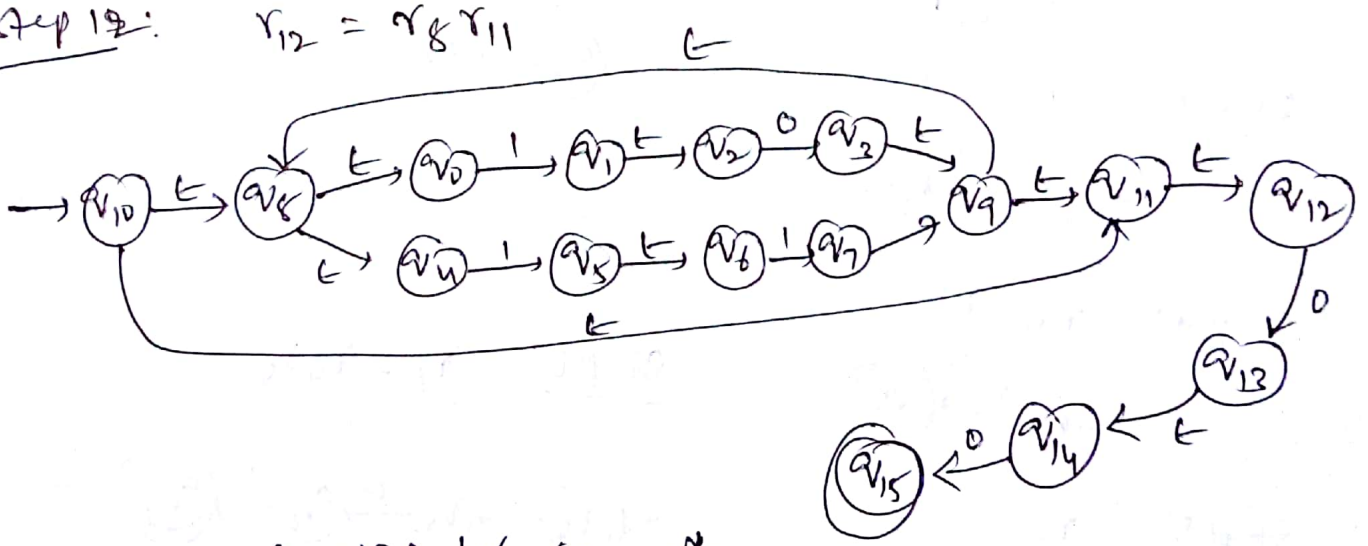
Step 10 $r_{10} = 0$



Step 11: $r_{11} = r_9 r_{10}$



Step 12: $r_{12} = r_8 r_{11}$

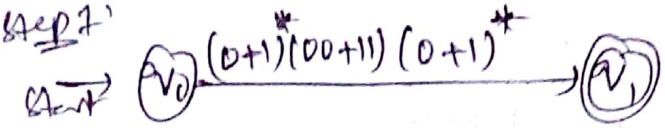


ϵ-NFA for $(10+11)^*00$

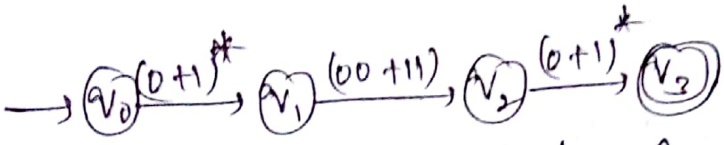
Ex: Construct a finite state automata equivalent to the regular expression $(0+1)^*(00+11)(0+1)^*$.

Sol:

Step 1:

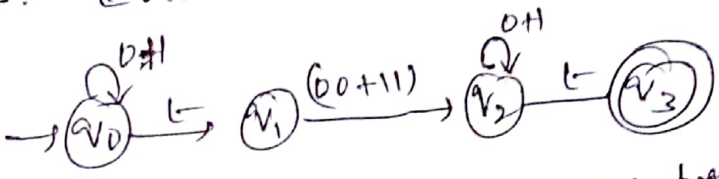


Step 2: Elimination of Concatenation operator

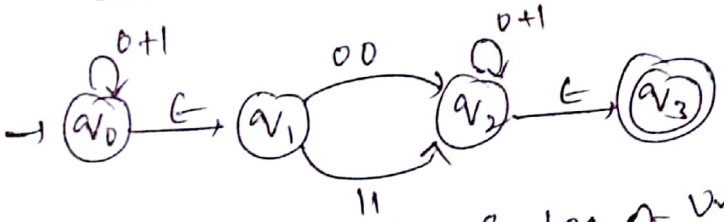


F.A after elimination of Concatenation operator

Step 3:- Elimination of Kleen closure:

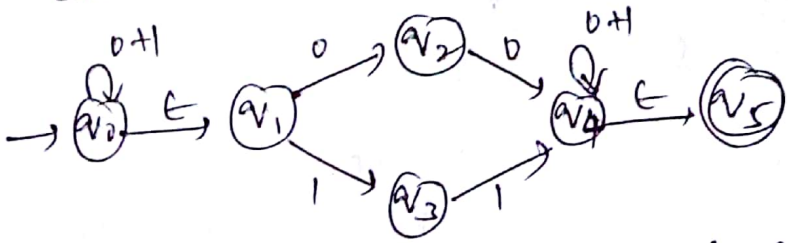


Step 4: Elimination of Union operation



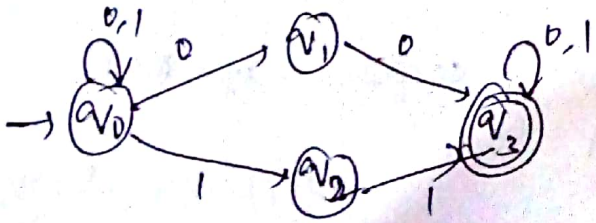
F.A after elimination of Union

Step 5: Elimination of Concatenation operation



F.A after elimination of Concatenation

Step 6: Elimination of $\in$- transition



F.A after $\in$-transitions.

Ex:- Construct a finite automata with ε-transition for the regular expression $r = 01^* + 10$.

Sol:- The given RE is $r = 01^* + 10$

The components of r is listed below

$r_1 = 0$ $\quad$ $r_5 = 1$

$r_2 = 1$ $\quad$ $r_6 = 0$

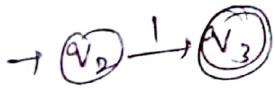
$r_3 = r_2^*$ $\quad$ $r_7 = r_5 r_6$

$r_4 = r_1 r_3$ $\quad$ $r_8 = r_4 + r_7$.

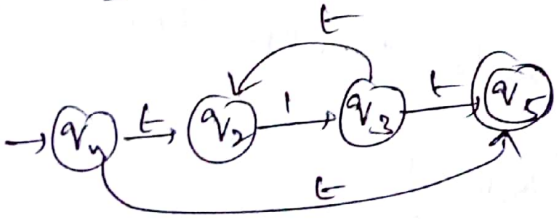
Construction of NFA-ε

Step 1: $r_1 = 0$

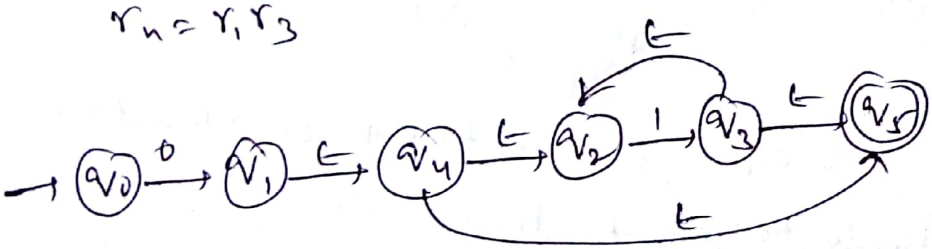
Step 2: $r_2 = 1$



Step 3: $r_3 = r_2^*$

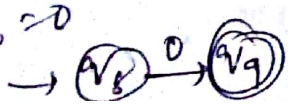


Step 4: $r_4 = r_1 r_3$

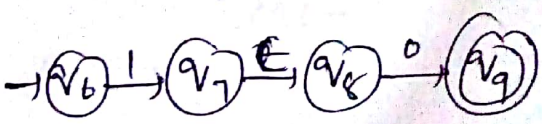


Step 5: $r_5 = 1$

Step 6: $r_6 = 0$

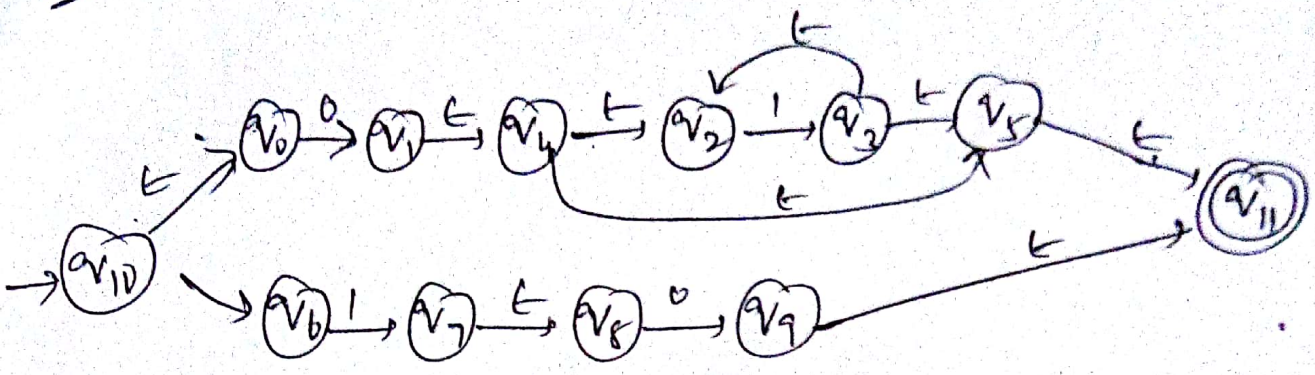


Step 7: $r_7 = r_5 r_6$



NFA-ε for $r_5 r_6$

Step 8: $r_8 = r_4 + r_7$



NFA-ε for $r = 01^* + 10$

Pumping Lemma for RE

II-C

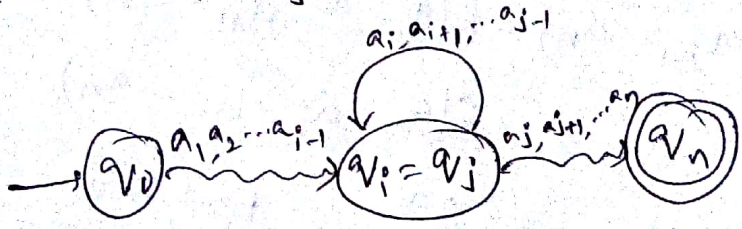
. The pumping lemma is generally used to prove a language is not regular.

Lemma (Theorem): If L is a regular language represented with an automaton with a maximum of n states, then there is a word in L such that $|z| \geq n$, further we can write $z = uvw$ in such a way that $|uv| \leq n$, $|v| \geq 1$ and for all $i \geq 0$, $uv^i w$ is in L.

Proof: Consider a DFA which has n states. For a string which is valid, which goes through all the states will have a length at least n-1.

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

If we consider a string of length n or more than n then there would be two states in the path, q_i and q_j such that $q_i = q_j$.



The string with length n would go through n+1 states. But there are only n states there at least one state is repeated. If q_i and q_j are two such states, then the string $z = a_1 a_2 \cdots a_{i-1} a_i a_{i+1} \cdots a_{j-1} a_j a_{j+1} \cdots a_n$ formed can be divided into three substrings uvw such that the string $u = a_1 a_2 \cdots a_{i-1}$ has a path from q_0 to q_i;

the string $v = a_i, a_{i+1}, \cdots a_{j-1}$ has a path from q_i to q_j and string $w = a_j a_{j+1} \cdots a_m$ has a path from q_j to q_f.

Since the string $a_i a_{i+1} \cdots a_{j-1}$ takes from q_i to q_j, if this string is pumped any number of times there will be path from q_1 to q_m as $a_1 a_2 \cdots a_{i-1} (a_i a_{i+1} \cdots a_{j-1})^* a_j a_{j+1} \cdots a_n$ which ~~takes~~ makes the string $(uv^i w)$ is valid.

To make this possible, the substrings to be considered should satisfy the following conditions:

(i) $|uv| \leq n$. This is the maximum possible string that can be selected to have path using n unique states.

(ii) $|v| > 0$. To show that there is a path from q_j to q_i, there should be at least one more ~~as~~ which makes to leave the state and enter the state.

$$\delta(q_0, a_1 a_2 \cdots a_{j-1} a_j a_{j+1} \cdots a_n) = \delta(q_i, a_j a_{j+1} \cdots a_n)$$
$$= \delta(q_j, a_j, a_{j+1}, \cdots a_n)$$
$$= \delta q_n \text{ . final state}$$

Applications of Pumping Lemma:

This theorem can be used to prove that certain sets are not regular.

Steps: Steps needed for proving that a given set is not regular.

Step 1: Assume that L is regular. Let n be the number of states in the corresponding finite automata.

Step 2: Choose the string z in L such that $|z| \geq n$. Use pumping lemma to write z = uvw, with $|uv| \leq n$ and $|v| > 0$.

Step 3: Find a suitable integer 'i' such that $uv^i w \notin L$. This contradicts our assumption. Hence L is not regular.

Ex: Show that $L = \{a^p \mid p \text{ is a prime}\}$ is not regular.

Proof:
- Let us assume L is a regular and p is a prime number
- Let $z = a^p$, by pumping, we can write z as

$z = uvw$ for $i = 1$

$|z| = uvw$ $|uv| \leq n$, $|v| > 0$.

Now Consider $L = uv^i w$ where $i = 2$

$= uvvw$

Adding 1 to p we get,

$p < |uvvw|$

$p < p+1$.

But p+1 is not a prime number. Hence, when we assumed it comes contradictory. Thus L behaves as it is not a regular language.

Show that set $L = \{0^i 1^i \mid i \geq 1\}$ is not regular.

Sol: Assume that $L = \{0^i 1^i \mid i \geq 1\}$ is regular. n is the number of states in finite automata.

$z = 0^n 1^n$ such that $|z| = 2n$.

By pumping lemma, we can write

$z = uvw$ such that $|uv| \leq n$ and $|v| \neq 0$.

Suppose $n = 3$, then

$z = 000111$ $\quad |uv| \leq n$

$z = uvw$ $\quad |v| \neq 0$.

Case 1: v is in the '0' part, then

$\underbrace{00}_{u}\underbrace{0}_{v}\underbrace{11}_{w}1$

Case 2: v is in the '1' part then

$\underbrace{000}_{u}\underbrace{11}_{v}\underbrace{1}_{w}$

Case 3: v is in the '01' part then

$\underbrace{00}_{u}\underbrace{01}_{v}\underbrace{11}_{w}$

for Case 1: for $i = 2$

$uv^i w = uv^2 w$

$\Rightarrow 00\,00111 \quad i.e.\ 0^4 1^3 \Rightarrow \notin L.$

for Case 2: for $i = 2$

$uv^i w = uv^2 w$

$000\,11\,11 \Rightarrow 0^3 1^4 \Rightarrow \notin L.$

for case 3: for $i=2$

$uv^iw = uv^2w$

$00010101 = 0^3 1^1 0^1 1^3 \notin L \Rightarrow$ which is not follow the $0^n 1^n$ .So

Hence for all these 3 cases it is clear that the language L is not regular.

Ex:- Prove that $L = \{ ww^R \in \{a,b\} \mid w = w^R \}$.

Sol Consider $L = \{ ww^R \in \{a,b\} \mid w = w^R \}$

$w = abab$

$w^R = baba$

Hence $L = ww^R = abababba$

ab ab ab ab
n n n n

The string z can be denoted by

$z = uvw$ where $|z| \geq n$, $|uv| \leq n$, $|v| \geq 1$.

we assume $z = \underbrace{abab}_{uv}\underbrace{baba}_{w}$

$|u| = n-1$

$|v| = 1$

$|uv| = |u| + |v| = n-1+1 = n$.

according to pumping lemma,

$z = uv^iw$ for $i=1$

$z = ababbaba \in L$

for $i=0$

$z = uw = aabababa \notin L, \Rightarrow ww^R$ —①

for $i=2$, then

$z = uv^2w = abbababaa \notin L, \neq ww^R$

$\neq ww^R$ —②

from equations ① & ②, we can say that

$z = uv^i w \notin L$.

Hence our assumption that $L = ww^R$ being regular wrong. Hence we can prove that $z = ww^R$ is not regular.

Ex:- $L = \{a^n b^{2m} \mid m, n > 0\}$ regular or not.

Sol:- let $n = 1, 2, 3, \ldots$

$m = 1, 2, 3, \ldots$

$L = \{a^1 b^2, a^1 b^4, a^1 b^6, \ldots\}$

$= \{abb, aabbbb, aabbbbbb, \ldots\}$

Let ~~$z = uvy$~~ (2) =

$z = uvy$, $|uv| \leq$ ~~length~~ string $= 2$

$|v| \geq 1$

According to pumping lemma,

$z = uv^i w$

$z = \underbrace{aa}_{u}\underbrace{bb}_{v}\underbrace{bb}_{w}$ $|z| = 6$

for $i = 0$,

$z = uw = aabbb$ $|z| = 5$

$\neq a^n b^{2m}$ $\notin L$

$\neq a^2 b^3$.

for $i = 2$, $z = uv^2 w$

$z = aa\,bb\,bbb$

$= a^2 b^5 \notin L$

Hence $L = a^n b^{2m}$ is not regular

Ex: $L = \{a^{2n} \mid n \geq 1\}$ is regular or not.

Sol: $n = 1, 2, 3, \ldots$

$L = \{a^2, a^4, a^6, a^8, \ldots\}$

$L = \{aa, aaaa, aaaaaa, \ldots\}$

$z_i = uv^i w$

String aaaa

$z = \underbrace{a}_{u}\underbrace{a}_{v}\underbrace{a}_{w}\underbrace{a}_{w}$

- for $i = 0$

$z = uv^0 w$

$= uw$

$z = aaa = a^3 \neq a^{2n}$

$= \notin L$

- for $i = 2$

$z = uv^2 w$

$= aa\,aa\,a$

$= a^5 \neq a^{2n}$

$= \notin L$

So, $L = a^{2n}$ is not regular.

Closure properties of Regular Languages:

- If certain languages are regular and language L is formed by from them by certain operations such as union or concatenation then L also a regular. Such languages represent the class of regular languages which is closed under the certain specific operations.
- The principle closure properties of both regular languages are

(1) The union of two regular languages is regular. L, M are r.l then L ∪ M

(2) The intersection of two regular languages is regular L ∩ M

(3) The complement of a regular language is regular. L, Σ. then $\Sigma^* - L$ is regular

(4) The difference of two regular languages is regular. L − M

(5) The reversal of regular regular languages is regular. L = {100}, L^R = {001}

(6) The closure of regular language is regular. ie. L r.l then L^* is regular

(7) The concatenation of regular language is regular L, M are r.l then LM r.l.

(8) A homomorphism of regular languages is regular.

Ex: A homomorphism is substitution of strings for symbols. h(0) = a, h(1) = b, then h is applied to 0011 is aabb

(9) The inverse homomorphism of regular languages is regular.

Applications of Regular Expressions:

- A regular expression gives a picture of the pattern we want to search in the text. When the pattern is specified using a regular expression, it is compiled into a deterministic or non-deterministic automata behind and then simulates the program to search for the given pattern. Regular expressions gained a wide range of usage in the UNIX operating system. The other applications are defining patterns for tokens which are used in lexical analysis phase and also in text processing.

Regular expressions in UNIX:-

There are various UNIX notations used for regular expressions. These notations have many additional capabilities and features. These notations have ability to name and refer previous string that have a matched pattern. This ability helps in recognizing non regular languages. UNIX regular expressions allow to write character classes. There are some rules for these character classes which are given below

i) The symbol . stands for any single character

ii) The sequence [1, 2, 3, ..10] means 1+2+3+...+10

iii) The range of characters can be specified using square brackets.

Ex: [a-z] denotes the set of lower case letters.

(iv) for matching with some special character a backslash is used.

Ex: \- means match with character -

(v) special notations can be used to represent some common class of characters

Ex: [:digit:] represent the class [0-9]

[:alpha:] is same as [A-Za-z]

(vi) The operator | is used to denote union.

(vii) ? used to denote preceding zero or single character.

(viii) Ex: ?[0-9] means the number can be positive or negative number

(ix) + is used to denote one or more occurances

(x) * is used to denote zero or more occurances

{n} means n copies of.

Ex: a{3} means aaa.

2. Lexical Analyzer:

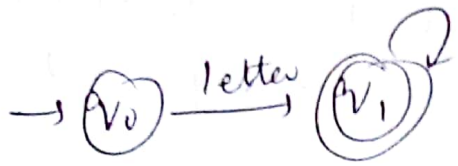
The compiler has a lexical analysis phase which forms the most important initial processing where the source programs is scanned for the recognitions of tokens. The tokens are defined by a regular expression for each pattern.

Ex: An identifier is a token which has a pattern:
- an alphabet followed by any number of alpha-numeric characters. Corresponding R.E is

$$[A-Za-z][A-Za-z0-9]^*$$

re (letter)(letter + digit)*

then the corresponding finite automata is



letter : [A-Z a-z]
DIGIT : [0-9]

Fig: DFA for identifiers.

3. Finding patterns in text:

Regular expressions are useful notations used for finding the patterns from the given text. A large class of pattern can be identified by regular expression.

Ex: the strings ending with digits is a pattern in the given text which can be recognized by following r.e

[a-z A-Z] + [0-9] +

Thus the use of regular expressions for identifying patterns from given text makes the job of compiler more simplified.

In web applications also the pattern matching is done using the regular expressions.

- To search files in the system based on a pattern
- To classify the business by their location and to answer the queries
- To create the mailing list.

Regular Grammar

Regular Grammar

A regular grammar is defined as $G = (V, T, P, S)$

where 'V' is set of symbols called nonterminals which are used to defined the rules. (variables)

'T' is set of symbols called terminals

'P' is a set of production rules

'S' is a start symbol which $\in V$.

The production rules P are of the form

$A \to aB$

$A \to a$

where A, B are nonterminals, a is terminal symbol.

Ex: $G = (V, T, P, S)$ with $V = \{S, A\}$, $T = \{0, 1\}$

S is a start symbol and production rules are given below

$S \to 0S$

$S \to 1A$

$A \to \epsilon$.

This is called a regular language and it can be represented by some DFA. Thus, when a grammar can be represented by some finite automata then it is a regular grammar.

Types of Grammar

There are two types of grammar.

(*) Right-linear grammar

(*) Left-linear grammar

Right-linear grammar:

If the non-terminal symbol appears as a right symbol in each production of regular grammar then it is called right linear grammar.

The right linear grammar is of the following form

$A \to aB$

$A \to a$

$B \to \epsilon$

where A and B are nonterminal symbols and 'a' is a termind symbol.

Left-linear Grammar:

If the non-termind symbol appears as a leftmost symbol in each production of regular grammar then it is called left linear grammar.

The left linear grammar is of the form

$A \to Ba$

$A \to a$

$B \to \epsilon$

where A and B are non-terminal symbols and a is a termind symbol.

- The language is called regular if it is accepted by either left linear or right linear grammar.

Converting

Let L be

Let us

We

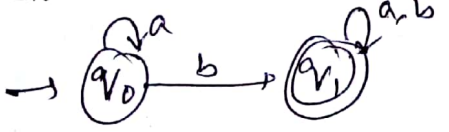
Converting Finite Automata to Regular Grammar:

Let L be the language for some DFA $M = (Q, \Sigma, \delta, q_0, F)$. Let us assume that q_0 is not the final state. For this we have to find a grammar G such that $L(G) = L(M)$.

Let $G = (Q, \Sigma, P, q_0)$, where

- Q is the set of states, which are considered as variables in the grammar.
- the set of input symbols are considered as terminal symbols
- the start state of NFA is considered as starting symbol in the grammar.
- the productions are defined from transitions δ as
  - → If there is a transition of the form $\delta(q, a) = p$, then add $q \to ap$ if p is some non-final state.
  - → if there is a transition of the form $\delta(q, a) = p$, then add $q \to ap \mid a$ if p is some final state.

Ex:- Construct regular grammar for the DFA given below



Sol: The grammar equivalent to given DFA is (right linear grammar)

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

$V = \{q_0, q_1\}$

$T = \{a, b\}$

$S = q_0$

P = { $q_0 \to a q_0$, $q_0 \to b q_1$, $q_0 \to b$, $q_1 \to a q_1$, $q_1 \to b q_1$, $q_1 \to a$, $q_1 \to b$ }

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

$V = (A, B)$

$T = \{a, b\}$

$S = \{A\}$

productions P are

$A \to aA$

$A \to bB$

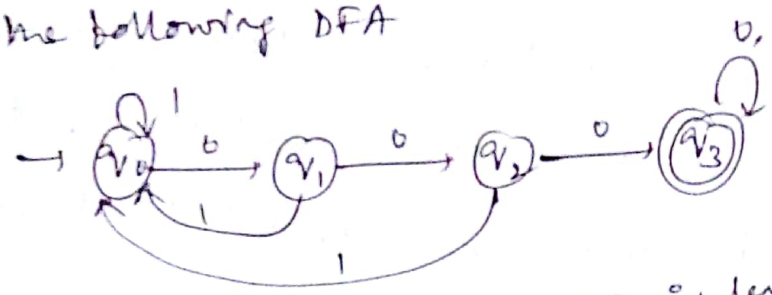
$A \to b$

$B \to aB$

$B \to bB$

$B \to a$ $B \to b$.

Ex:- find the right linear grammar (regular grammar) for the following DFA



Sol:- For the given DFA, the equivalent right linear grammar

(variables are represented as capital letters.)

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

$V = \{q_0, q_1, q_2, q_3\}$

$T = \{0, 1\}$

$S = \{q_0\}$

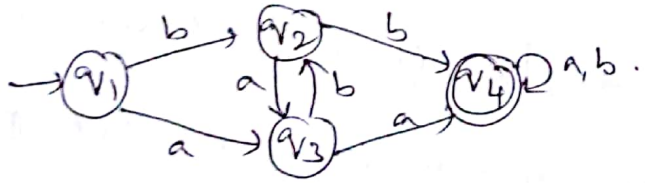
$P = \{ q_0 \to 0q_1 \mid 1q_0$

$q_1 \to 0q_2 \mid 1q_0$

$q_2 \to 0q_3 \mid 0 \mid 1q_0$

$q_3 \to 0q_3 \mid 1q_3 \mid 0 \mid 1 \}$

Ex:- Construct a regular grammar for given FA



Sol: $G = \{V, T, P, S\}$

$V = \{q_1, q_2, q_3, q_4\}$

$T = \{a, b\}$

$S = q_1$

$q_1 \to bq_2$

$q_1 \to aq_3$

$q_2 \to bq_3$

$q_2 \to bq_4 \mid b$

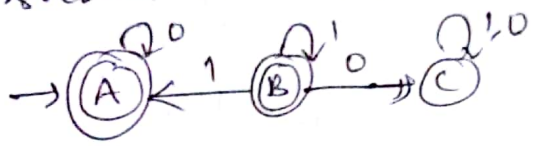
$q_3 \to aq_4 \mid a$

$q_3 \to bq_2$

$q_4 \to aq_4 \mid a$

$q_4 \to bq_4 \mid b$

Ex:- Obtain regular grammar for the following FA as shown below fig



Sol:- $G = \{V, T, P, S\}$

$V = \{A, B, C\}$

$T = \{0, 1\}$

$S = A$

$P = \{$ $A \to 0A$
$A \to 0$
$B \to 1B \mid 1A$
$B \to 1$
$B \to 0C$
$C \to 1C$
$C \to 0C$

$\}$

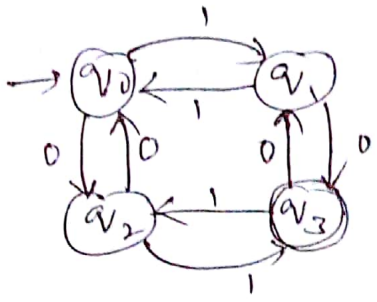
Strings generated from the grammar

A	A	A
0	0A	0A
	00A	00A
	000	000A
		0000 or ...

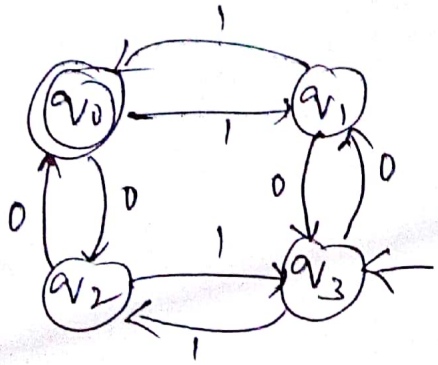
Construction of left linear grammar:

(1) Convert the given right linear grammar to equivalent FA.
(2) From the FA, interchange the initial state and the final state
(3) Reverse the directions of arrows on all edges
(4) Construct the regular grammar from this FA.

Ex:- find the left linear grammar for the DFA given below



Sol: To find the left linear grammar, first we reverse the DFA by reversing all the edges, and making the initial state as final state, and the final state as initial.



Reversed DFA.

for reversed DFA, the equivalent left linear grammar is

$G = \{V, T, P, S\}$

$V = \{q_0, q_1, q_2, q_3\}$

$T = \{0, 1\}$ $S = \{q_3\}$

R.L

$P = \{ q_3 \to 0q_1 \mid 1q_2$
$q_2 \to 0q_0 \mid 1q_3 \mid 0$
$q_1 \to 0q_3 \mid 1q_0 \mid 1$
$q_0 \to 0q_2 \mid 1q_1$ }.

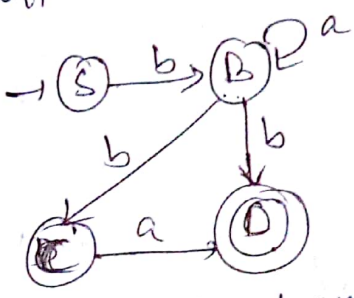
To get the left linear grammar reverse all the productions of the above grammar

$P = \{ q_3 \to q_1 0 \mid q_2 1$
$q_2 \to q_0 0 \mid q_3 1 \mid 0$
$q_1 \to q_3 0 \mid q_0 1 \mid 1$
$q_0 \to q_2 0 \mid q_1 1$ }.

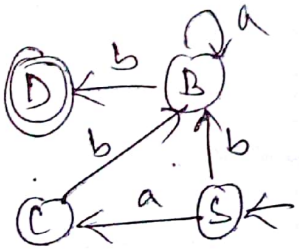
Ex: Convert the given right linear grammar into equivalent left linear grammar.

$S \to bB$
$B \to bC$
$B \to aB$
$C \to a$
$B \to b$

Sol: Given equivalent FA can be



Now, we will change the final state as initial stated and initial state as final state and change the change the edge directions.



The right linear grammar is

$S \to aC$
$S \to bB$
$B \to aB$
$B \to bD \mid b$
$C \to bB$

To get the left linear grammar, reverse all the products of the above grammar

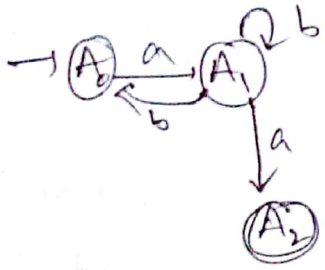
$S \to Ca$
$S \to Bb$
$B \to Ba$
$B \to Db \mid b$
$C \to Bb$

Convert the following right linear grammar to left linear grammar.

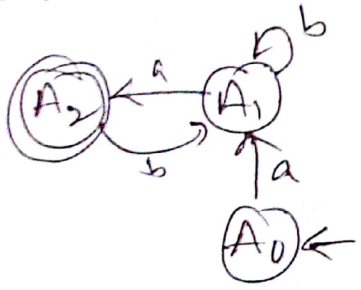
$A_0 \to bA_1$
$A_1 \to bA_1$
$A_1 \to a$
$A_1 \to bA_0$.

Sol:

First construct FA for given right linear grammar



Now, we will interchange of FA and interchange initial and final states



Now let us write left linear grammar.

R.L

$A_0 \to aA_1$
$A_1 \to aA_2$
$A_1 \to a$
$A_1 \to bA_1$
$A_2 \to bA_1$

Right linear

L.L

$A_0 \to A_1a$
$A_1 \to A_2a$
$A_1 \to a$
$A_1 \to A_1b$
$A_2 \to A_1b$

Left linear gramma.

String abbaa

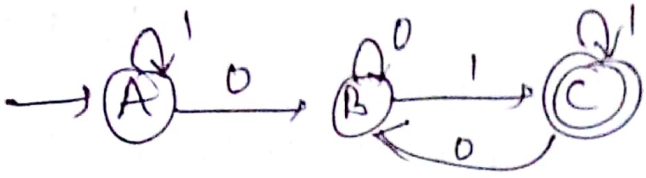
Right linear gramma

A_0
aA_1
abA_1
$abbA_0$
$abbaA_1$
$abbaa$

Left linear grammar

A_0
A_1a
A_2aa
A_1baa
A_1bbaa
$abbaa$

Ex:- Obtain left linear grammar for the following DFA.

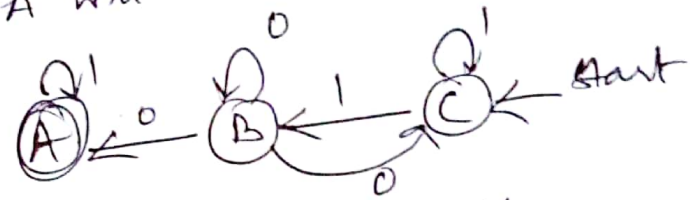


Sol:- Steps

① Interchange the initial and final states

② Reverse the directions of all the edges.

The DFA will be



The left linear grammar is

$C \rightarrow C1$

$C \rightarrow B1$

$B \rightarrow B0$

$B \rightarrow C0$

$B \rightarrow A0$

$B \rightarrow \epsilon$

$A \rightarrow A1$

$A \rightarrow 1$

Conversion of Left-Linear Grammar to Right Linear Grammar.

Method: (i) Convert the given left-linear grammar to F.A

(ii) Interchange the initial and final states of FA

(iii) Reverse the directions of all arrows on the all edges.

(iv) Construct the regular grammar from this F.A.

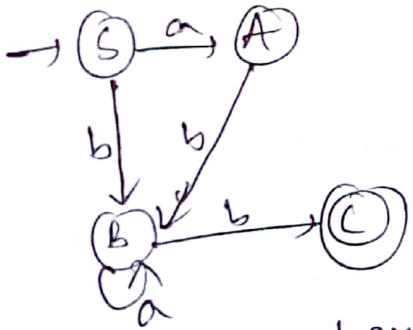
Ex: Convert the following left-linear grammar to right linear grammar

$S \to Aa \mid Bb$

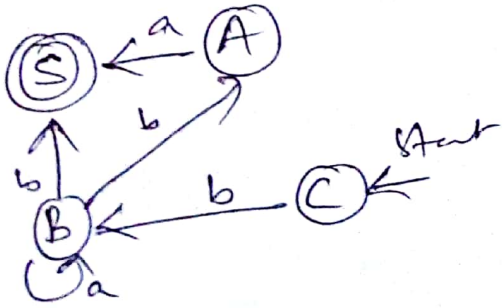
$A \to Bb$

$B \to Ba \mid b$

Sol: Step 1: Design F.A for given left linear grammar



Step 2: make change initial state as final state and final state as initial state and interchange the directions of all arrows along the edges



Step 3: write right linear grammar

$C \to bB$

$B \to aB$

$B \to bA \mid bS \mid b$

$A \to aS \mid a$

Construction of FA from Regular Grammar

Let $G = (V, \Sigma, P, A_0)$ be a regular grammar. We can construct DFA 'M' where

(i) states corresponds to Variables

(ii) Initial state correspond to A_0.

(iii) Transitions in M correspond to productions in P

If there is a production of the form $A_i \to a$, the corresponding transition terminates at a new state. This is the unique final state.

Thus, the DFA M can be defined as

$$M = \{(q_0, q_1, \ldots q_f), \Sigma, \delta, q_0, \{q_f\}\}$$

The 'δ' defined as

(i) Each production $A_i \to a A_j$ induces a transition from q_i to q_j with a label 'a'.

(ii) Each production $A_K \to a$ induces a transition from q_K to q_f with label 'a'.

Therefore $L(G)$ can be given by corresponding FA M.

Ex:- ① Construct DFA equivalent to the grammar

$S \to aS \mid bS \mid aA$, $A \to bB$, $B \to aC$, $C \to \epsilon$.

Sol:- The DFA can be constructed using following rules

(i) Each production $A_i \to aA_j$ induces a transition from q_i to q_j with label a on the edge.

(ii) Each production $A_k \to a$ induces a transition from q_k to q_f with label a on the edge.

The DFA can be

$S \to aS$ $\quad A \to bB$
$S \to bS$ $\quad B \to aC$
$S \to aA$ $\quad C \to \epsilon$.

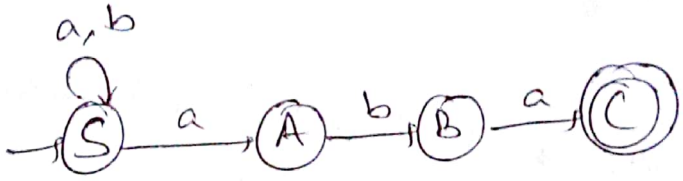


fig: DFA.

Ex:- Construct FA recognizing L(G), where G is the grammar $S \to aS \mid bA \mid b$ and $A \to aA \mid bS \mid a$.

Sol:- Given that

$S \to aS$ $\quad A \to aA$
$S \to bA$ $\quad A \to bS$
$S \to b$ $\quad A \to a$

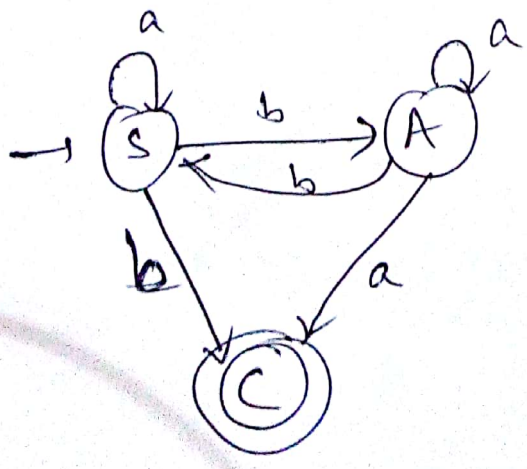


fig: DFA

Ex:- Let $G = (\{A_0, A_1\}, \{a, b\}, P, A_0)$ where P consists of $A_0 \to aA_1$, $A_1 \to bA_1$, $A_1 \to a$, $A_1 \to bA_0$. Construct a transition system M accepting L(G).

Sol:- Given that

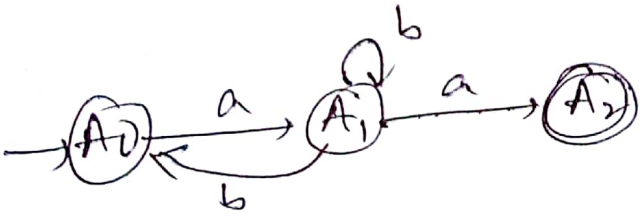
$G = (\{A_0, A_1\}, \{a, b\}, P, A_0)$

$A_0 \to aA_1$

$A_1 \to bA_1$

$A_1 \to a$

$A_1 \to bA_0$



$A_0 = q_0$

$A_1 = q_1$

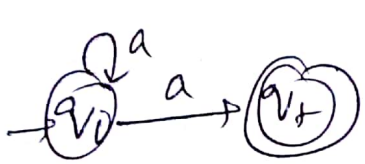
$A_2 = q_f$

Ex:- If a regular grammar G is given by $S \to aS \mid a$, find M accepting L(G).

$S \to aS$

$S \to a$

Sol:-



$M = (\{q_0, q_f\}, a, \delta, q_0, \{q_f\})$

Construction of Regular Grammars from Regular Expression

- We can covert the regular expression into RG equivalent regular grammar by using following method.

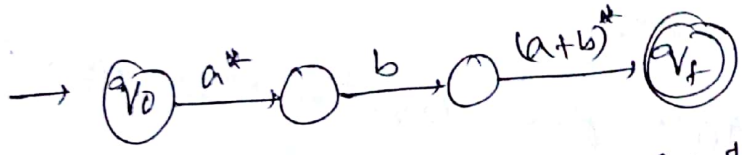
(•) Construct a NFA with ∈ moves from given regular expression.

(•) Eliminate ∈ transitions and convert it to equivalent DFA.

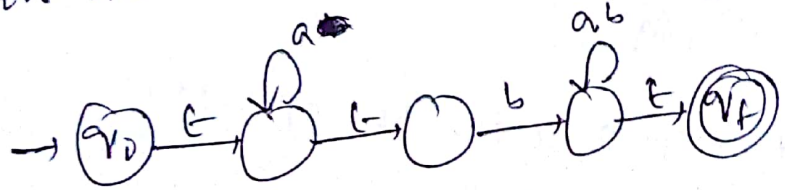
(•) from constructed DFA, the ~~constructed~~ corresponding states becomes nonterminal symbols and transitions made are equivalent to production rules.

Ex:- Construct a regular grammar for the regular expression $a^*b(a+b)^*$.

Sol:- We will first construct a DFA in a straight forward manner for given regular expression.



Now we will convert it to NFA with ∈-transitions



eliminate ∈ moves

we can write the regular grammar as

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

where $V = \{A_0, A_1\}$

$T = \{a, b\}$

$S = A_0$

$P = \{ A_0 \to aA_0$

$A_0 \to bA_1$

$A_0 \to b$

$A_1 \to aA_1$

$A_1 \to bA_1$

$A_1 \to a$

$A_1 \to b$

$\}$

$q_0 = A_0$

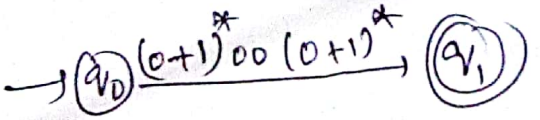
$q_1 = A_1$

Step 3: elim

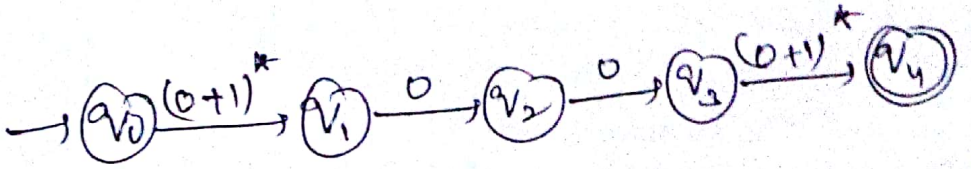
Ex:- Construct regular grammar for the given regular expression $(0+1)^*00(0+1)^*$.

Sol:- we have to construct the regular finite automata for the given regular expression.

Step 1: The FA for the given regular expression is



Step 2: Elimination of concatenation operation



F.A after elimination of concatenation operation

Step 3: elimination of Kleen closure

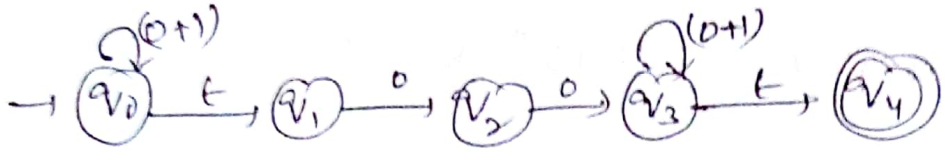


fig: finite automata after elimination of Kleen closure

Step 4: elimination of Union operation

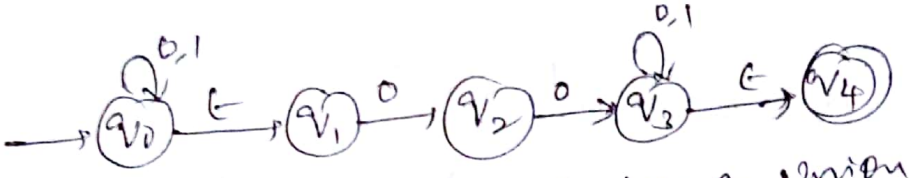
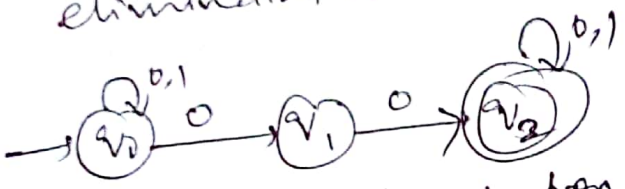


fig: FA after elimination of Union.

Step 5: elimination of ϵ-moves



F.A after elimination of ϵ-moves.

Then the Regular Grammar $G = \{V, T, P, S\}$

$V = \{A, B, C\}$

$T = \{0, 1\}$

$S = qA$

$q_0 = A$
$q_1 = B$
$q_2 = C$

Right linear grammar

$P = \{ A \to 0A$
$A \to 1A$
$A \to 0B$
$B \to 0C$
$B \to 0$
$C \to 0C$
$C \to 1C$
$C \to 0$
$C \to 1$
$\}$

Left linear Grammar is after changing the position of terminals & non-terminals.

$A \to A0$
$A \to A1$
$A \to B0$
$A \to C0$
$B \to 0$
$C \to C0$
$C \to C1$
$C \to 0$
$C \to 1$

Construct regular grammar for $(a|b)^*abb$

Sol) The required NFA for $(a|b)^*abb$ is

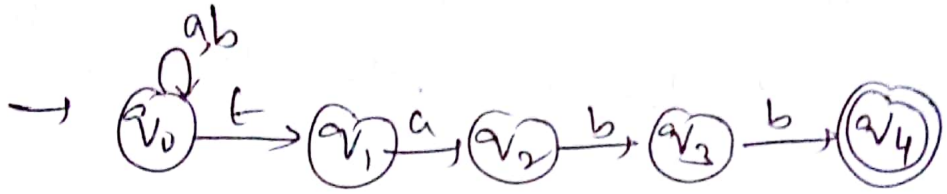


fig: NFA with ε moves

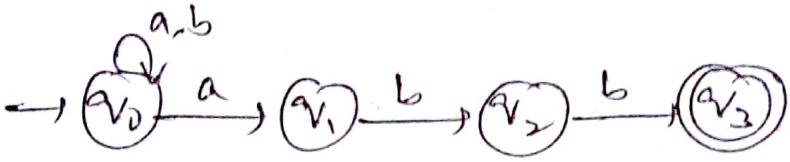


fig NFA without ε for $(a+b)^*abb$.

Regular Grammar $G = \{V, T, P, S\}$

$V = \{A, B, C, D\}$

$T = \{a, b\}$

$S = A$

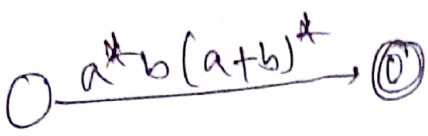
$P = \{ A \to aA$
$A \to aB$
$A \to bA$
$B \to bC$
$C \to bD$
$D \to C$
$C \to b \}$

$q_0 = A$
$q_1 = B$
$q_2 = C$
$q_3 = D$

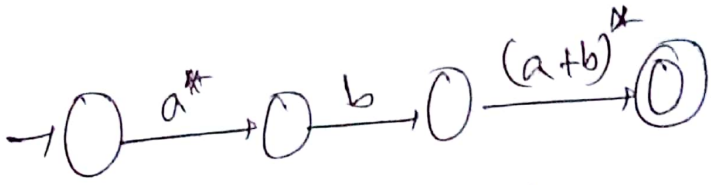
A Construct a regular grammar G generating the regular set represented by $P = a^*b(a+b)^*$.

Sol:- We construct the DFA corresponding to P using

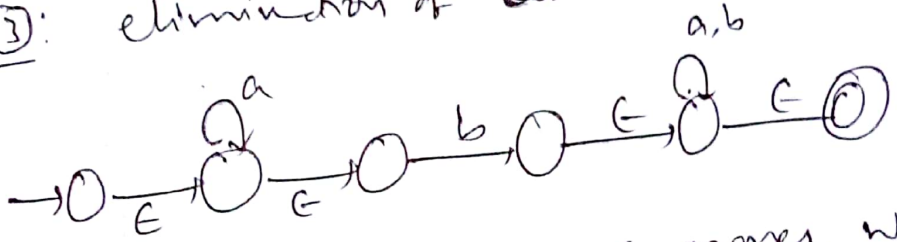
Step①



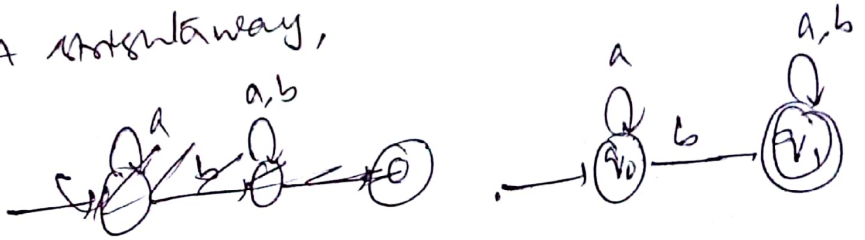
Step② elimination of concatenation



Step③: elimination of closure



Step④: After eliminating the ϵ-moves, we get the DFA diagram as,



Now $G = \{\{A_0, A_1\}, \{a, b\}, P, A_0\}$, where P is given by

$A_0 \to aA_0$ $A_0 \to b$

$A_0 \to bA_1$

$A_1 \to aA_1$

$A_1 \to bA_1$

$A_1 \to a$

$A_1 \to b$

1 Construct the left-linear and right-linear grammar for regular expression $0^*(1(0+1))^*$

Sol:- Construction of right-linear grammar $0^*(1(0+1))^*$

- first we have to construct the FA for the given regular expression.

Step 1:

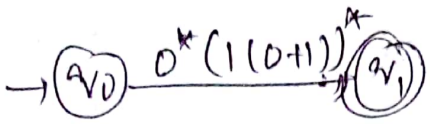


fig: FA for reg exp $0^*(1(0+1))^*$

Step 2: elimination of Concatenation

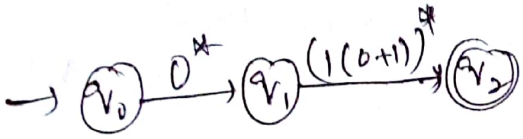


fig: FA after elimination of Concatenation.

Step 3: elimination of Kleen closure

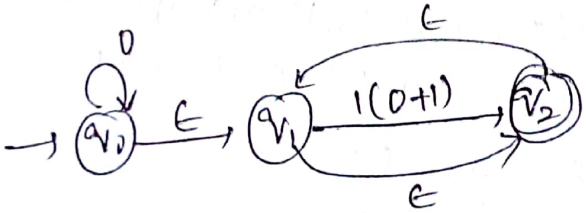
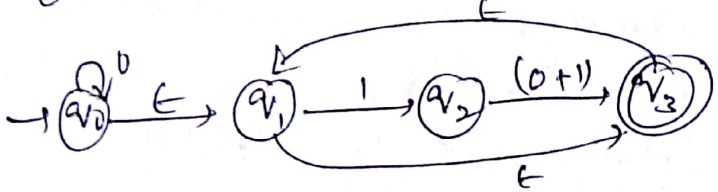
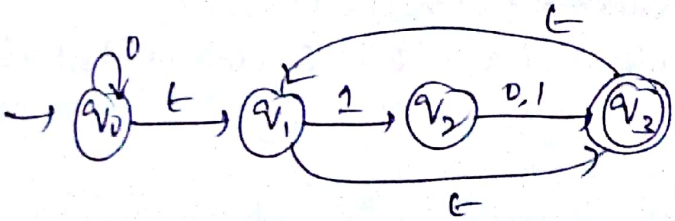


fig. FA after elimination of closure.

Step 4: elimination of Concatenation



Step 5: elimination of Union



Step: elimination of ϵ moves

State	Inputs	
	0	1
q_0	$\{q_0, q_1, q_3\}$	$\{q_2\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\{q_3, q_1\}$	$\{q_3, q_1\}$
q_3	$\emptyset$	$\{q_2\}$

ϵ-closure$(q_0) = \{q_0, q_1, q_3\}$

ϵ-closure$(q_1) = \{q_1, q_3\}$

ϵ-closure$(q_2) = \{q_2\}$

ϵ-closure$(q_3) = \{q_3, q_1\}$

$\hat{\delta}(q_0, 0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0, \epsilon), 0))$

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0), 0))$

$= \epsilon\text{-closure}(\delta(\{q_0, q_1, q_3\}, 0))$

$= \epsilon\text{-closure}(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_3, 0))$

$= \epsilon\text{-closure}(q_0 \cup \emptyset \cup \emptyset) = \epsilon\text{-closure}(q_0) = \{q_0, q_1, q_3\}$

$\hat{\delta}(q_0, 1) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_0, \epsilon), 1))$

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_0), 1))$

$= \epsilon\text{-closure}(\delta(\{q_0, q_1, q_3\}, 1))$

$= \epsilon\text{-closure}(\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_3, 1))$

$= \epsilon\text{-closure}(\emptyset \cup q_2 \cup \emptyset) = \epsilon\text{-closure}(q_2) = \{q_2\}$

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

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

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

$= \epsilon\text{-closure}(\delta(q_1, 0) \cup \delta(q_3, 0))$

$= \epsilon\text{-closure}(\emptyset \cup \emptyset) = \epsilon\text{-closure}(\emptyset) = \{\emptyset\}$

$\hat{\delta}(q_1, 1) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_1, \epsilon), 1))$

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

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

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

$= \epsilon\text{-closure}(q_2 \cup \emptyset) = \epsilon\text{-closure}(q_2) = \{q_2\}$

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

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

$\hat{\delta}(q_3,0) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_3,\epsilon),0))$
$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_3),0))$
$= \epsilon\text{-closure}(\delta(\{q_3,q_1\},0))$
$= \epsilon\text{-closure}(\delta(q_3,0) \cup \delta(q_1,0))$
$= \epsilon\text{-closure}(\emptyset \cup \emptyset) = \epsilon\text{-closure}(\emptyset) = \emptyset$

$\hat{\delta}(q_3,1) = \epsilon\text{-closure}(\delta(\hat{\delta}(q_3,\epsilon),1))$
$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(q_3),1))$
$= \epsilon\text{-closure}(\delta(\{q_3,q_1\},1))$
$= \epsilon\text{-closure}(\delta(q_3,1) \cup \delta(q_1,1))$
$= \epsilon\text{-closure}(\emptyset \cup q_2) = \epsilon\text{-closure}(q_2) = \{q_2\}$

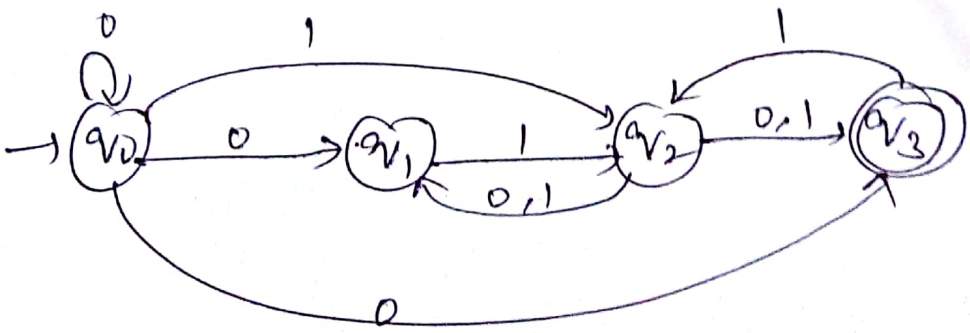


fig: FA for without ϵ moves

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

The right linear grammar is constructed. Assume the states q_0, q_1, q_2, q_3 as variables A, B, C, D respectively. Then

The grammar is $G = \{V, T, P, S\}$

$V = \{A, B, C, D\}$

$T = \{0, 1\}$

$S = A$

$P = \{ A \to 0A$

$A \to 0B$

$A \to 0D$

$A \to 0$

$A \to 1C$

$B \to 1C$

$C \to 0B$

$C \to 0D$

$C \to 0$

$C \to 1B$

$C \to 1D$

$C \to 1$

$D \to 1C \}$.

Construction Left linear grammar:

Step 1: reverse the given regular expression $r = 0^*(1(0+1))^*$

then we get $r_1 = ((1+0)1)^*0^*$

Step 2: Now, we have construct FA for the reverse r.e

then

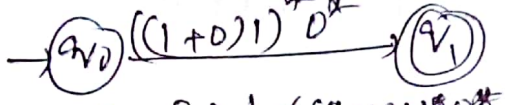


Fig: F.A for $((1+0)1)^*0^*$

Step 3: elimination of concatenation

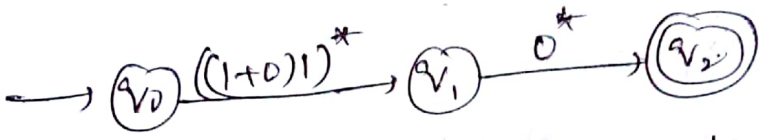


Fig: FA for $(1+0)1$ after elimination of concatenation

Step 4: elimination of kleen closure

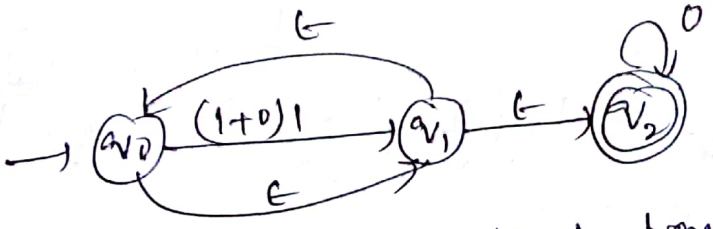


Fig: F.A after elimination of kleen closure

Step 5: elimination of concatenation

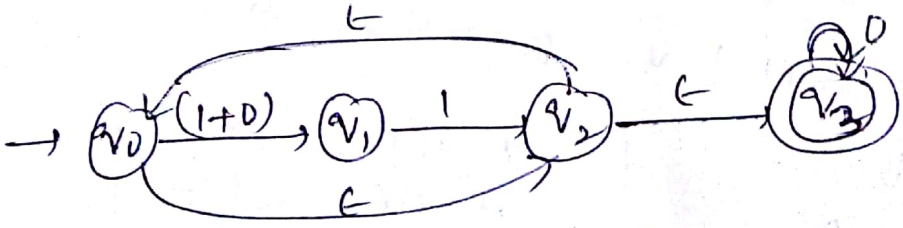


Fig: F.A after elimination of concatenation

Step 6: elimination of Union

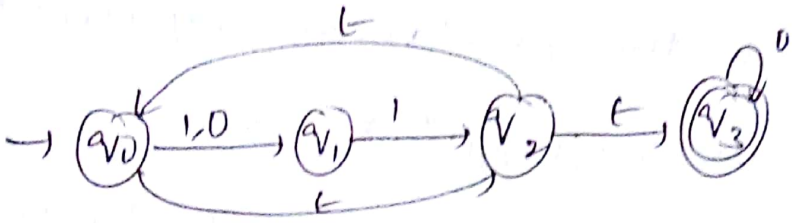
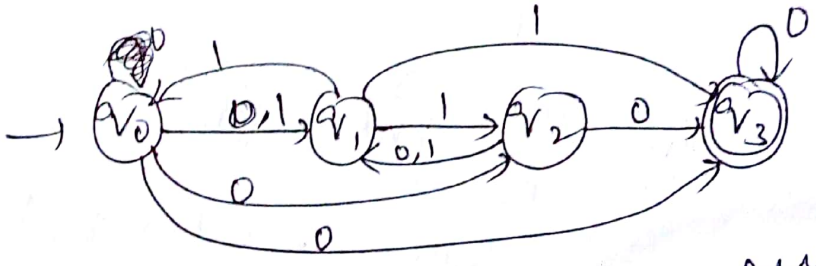


Fig: After elimination of Union

Step 7: elimination of ε-transitions

State	Input	
	0	1
q_0	$\{q_0, q_2, q_3\}$	$\{q_1\}$
q_1	ϕ	$\{q_2, q_0, q_3\}$
q_2	$\{q_1, q_3\}$	$\{q_1\}$
q_3	$\{q_3\}$	ϕ

Step 8:



We construct the right linear grammar. Assume the states q_0, q_1, q_2 and q_3 as variables A, B, C and D resp.

Then the right linear grammar is

$G = \{V, T, P, S\}$

$V = \{A, B, C, D\}$

$T = \{0, 1\}$

$S = A$

$P = \{$ $A \to 0B$, $A \to 0C$, $A \to 0D$, $A \to 0$, $A \to 1B$, $B \to 1A$, $B \to 1C$, $B \to 1D$, $B \to 1$, $C \to 0D$, $C \to 0B$, $C \to 0$, $C \to 1B$, $D \to 0D$, $D \to 0$ $\}$

we can also construct the left-linear grammar, by interchanging the positions of terminals and non-terminals of right-linear grammar as

$G = \{V, T, P, S\}$

$V = \{A, B, C, D\}$

$T = \{0, 1\}$

$S = A$

$P = \{ A \to BO,\ B \to A1,\ C \to DO,$
$A \to CO,\ B \to C1,\ C \to BO,$
$A \to DO,\ B \to D1,\ C \to O,$
$A \to O,\ B \to 1,\ C \to B1,$
$A \to B1,\ D \to DO,$
$D \to O \}$