UNIT-II

Regular Expressions

→ The language accepted by finite automata can easily described by simple expression called regular expression (R.E). It is most effective way to represent any language.

→ The languages accepted by some regular expression are refered to as regular languages.

→ A Regular expression can also be described as a sequence of pattern that defines a string.

. Let Σ be an alphabet which is used to denote the input set. The regular expression over Σ can be defined as follows.

1. ϕ is a regular expression which denotes the empty set.
2. ε is a regular expression and denotes the set $\{\varepsilon\}$ and it is a null string.
3. For each 'a' in Σ 'a' is a regular expression and denotes the set $\{a\}$.
4. If r and s are regular expressions denoting the languages L_1 and L_2 respectively, then

   r+s is equivalent to $L_1 \cup L_2$ i.e. Union

   rs is equivalent to $L_1 L_2$ i.e. Concatenation

   r^* is equivalent to L_1^* i.e. closure

The r^* is known as kleen closure or closure which indicates occurrences of r for ∞ number of times.

for Instance

→ In Regular Expression, x^* means zero & more occurrences of x
it can generate $\{\epsilon, x, xx, xxx, \ldots\}$

→ In Regular Expression, x^+ means one or more occurrences of x.
It can generate $\{x, xx, xxx, \ldots\}$

Operations on Regular Expressions

① Union :— If L & M are two regular languages their union L∪M is also union.

$L \cup M = \{S / S \text{ is in } L \text{ or } S \text{ is in } M\}$

② Intersection :- If L & M are two regular languages then their intersection is also an intersection.

$L \cap M = \{st / s \text{ is in } L \text{ & } t \text{ is in } M\}$

③ Kleen closure :- If L is regular language then its kleen closure L^* will also be regular language.

L^* = zero or more occurrences of language.

Example

① Design a Regular Expression to accept all possible combination of a's & b's over $\Sigma = \{a, b\}$

Sol: Given $\Sigma = \{a, b\}$

$L = \{ \epsilon, a, b, ab, ba, aab, \ldots \}$

Regular Expression $= (a+b)^*$ (it generates all possible combinations)

$L = \{a, b, ab, ba \ldots\} = (a+b)^+$

② Ends with 00 over an alphabet $\{0, 1\}$

Ans:- $\Sigma = \{0, 1\}$

$L = \{00, 100, 000, 0100, \ldots\}$

$R.E = (0+1)^*.00$

③ Starts with 1 and ends with 0

Sol:- $\Sigma = \{0, 1\}$

$L = \{10, 100, 1100, \ldots\}$

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

④ Any no. of a's followed by b's followed by any no. of c's

Ans: $\Sigma = \{a, b, c\}$

$L = \{abc, abbc, aabcc, \ldots\}$

$R.E = a^*b^*c^*$

⑤ Atleast one A, followed by one B, followed by one C.

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

$L = \{abc, \ldots\}$

$R.E = a^+b^+c^+$

⑥ which accepts all the strings begin or ends with either 00 or 11

Sol:- $\Sigma = \{0,1\}$

$L = \{00, 001, 0010, 110, 011, \ldots\}$

L_1 = Begin with 00 or 11 — $(00+11)(0+1)^*$

L_2 = Ends with 00 or 11 — $(0+1)^*(00+11)$

$R.E = L_1 + L_2$

$R.E = [(00+11)(0+1)^* + (0+1)^*(00+11)]$

⑦ The 3rd character from right end is always a over an alphabet $\Sigma = \{a,b\}$

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

$L = \{aba, aaab, \ldots\}$

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

⑧ Atleast 2 b's over an alphabet $\Sigma = \{a,b\}$

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

$L = \{bb, abb, bba, aabb, abab, abbb, \ldots\}$

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

⑨ Exactly 2 b's over an $\Sigma = \{a,b\}$

$R.E = a^* b a^* b a^*$

⑩ Construct R.E which Contains even length strings over an alphabet $\Sigma = \{0\}$

$L = \{00, 0000, 000000, \ldots\}$

$R.E = (00)^*$

(11) Design R.E which contain odd length of strings over $\Sigma = \{1\}$

Sol: $L = \{1, 111, 11111, \ldots\}$

$R.E = 1.(11)^*$

(12) Accept all strings over no. of a's are divisible by 3 over an alphabet $\Sigma = \{a, b\}$

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

$R.E = (b^*ab^*ab^*ab^*)^3$

(13) Atmost 2 a's over an alphabet $\Sigma = \{a, b\}$

Any: If Zero a's $\longrightarrow b^*$

One a's $\longrightarrow b^*ab^*$

Two a's $\longrightarrow b^*ab^*ab^*$

$R.E = b^* + b^*ab^* + b^*ab^*ab^*$

(14) Set of strings of length '3' $\Sigma = \{0, 1\}$

$R.E = (0+1)^3$

(15) Set of strings of atmost 3 $\Sigma = \{0, 1\}$

$R.E = (0+1)^0 + (0+1)^1 + (0+1)^2 + (0+1)^3$

(16) Does not contain a substring ab over an alphabet $\{a, b\}$

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

$R.E = b^*a^*$

(17) Not ends with aa over an $\Sigma = \{a, b\}$

$R.E = ((a+b)^*(ab+ba+bb)) + \epsilon + a + b$

(18) Accepting even no. of b's over an $\Sigma = \{a, b\}$

$L = \{\epsilon, abb, bba, bab, bbaa, aa, a, \ldots\}$

$R.E = a^*(ba^*ba^*)^*$

$R.E = a^* + (a^*ba^*ba^*)^*$

(19) Design R.E. which contains 3 consecutive 0s

$R.E = (0+1)^* 000 (0+1)^*$

(21) which contains substring ab

$R.E = (a+b)^* ab (a+b)^*$

(22) Design R.E, $L = \{a^n b^m / n \geq 4, m \leq 3\}$

$L = \{aaaabbb, aaaabb, \ldots\}$

$R.E = aaaa\, a^* (\epsilon + b + bb + bbb)$

(or)

$R.E = aaaa^+ (\epsilon + b + bb + bbb)$

(23) Design R.E, $L = \{a^n b^n / n+m$ is even$\}$

$R.E = (aa)^* (bb)^* + a(aa)^* b(bb)^*$

(24) Design R.E for accepting no 2 consecutive 0's.

No 2 Consecutive zeros, i.e. 0 must be followed by 1, no restriction on 1s.

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

(25) Accepting no 3 consecutive b's.

$R.E = (ba + a + bba)^* . (b + bb + \epsilon)$

(26) The set of all strings where the 10th symbol from right end is 1.

Ans:

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

(or)

$R.E = (0+1)^* 1 (0+1)^9$

Algebraic Laws for Regular Expressions

Let P, Q and R are regular expressions then the identity rules are

1. $\epsilon R = R\epsilon = R$
2. $\epsilon^* = \epsilon$ ϵ is null string.
3. $(\phi)^* = \epsilon$ ϕ is empty string
4. $\phi R = R\phi = \phi$
5. $\phi + R = R$
6. $R + R = R$
7. $R \cdot R^* = R^* \cdot R = R^+$
8. $(R^*)^* = R^*$
9. $\epsilon + RR^* = R^*$
10. $(P+Q)R = PR + QR$
11. $(P+Q)^* = (P^*Q^*) = (P^* + Q^*)^*$
12. $R^*(\epsilon + R) = (\epsilon + R)R^* = R^*$
13. $(R+\epsilon)^* = R^*$
14. $\epsilon + R^* = R^*$
15. $(PQ)^* P = P(QP)^*$
16. $R^* R + R = R^* R$

Applications of Regular Expressions

1. Regular expressions in UNIX

These are various UNIX notations used for regular expressions

The range of characters can be specified using square brackets

For example:- [a-z] denotes set of lower case letters.

For matching with some special character a backslash is used.

For example: \- means match with character -.

The operator | is used to denote union.

? is used to denote preceding zero or single character.

For example: ? [0-9] means the number can be positive or negative number.

+ is used to denote one or more occurences.

* is used to denote zero or more occurences.

2. Lexical analyzers: Compiler uses this program of lexical analyzer in the process of compilation. The task of lexical analyzer is to scan the input program and separate out the 'tokens'.

For example: Identifier is a category of token in the source language and it can be identified by regular expression as

(letter)(letter + digit)*

3. Finding patterns in text

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

For example: The strings ending with digits is a pattern in the given text which can be recognized by following regular expression- [a-z A-Z]+[0-9]+

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

4. GREP utility in Unix

The Unix there is a GREP utility for searching the desired pattern in the file. For searching the pattern from the Grep makes use of regular expressions. When particular string is present in the file, then using grep we get the line containing that string from the file.

For example: If we type the Grep command on UNIX prompt then we will get the output as follows $ grep

hello test.txt

hello friends

$ grep b.g test.txt.

big brother ← these are are lines

bug ← present in test.txt.

bag of books ← file containing

bigger ← b[any single character]g

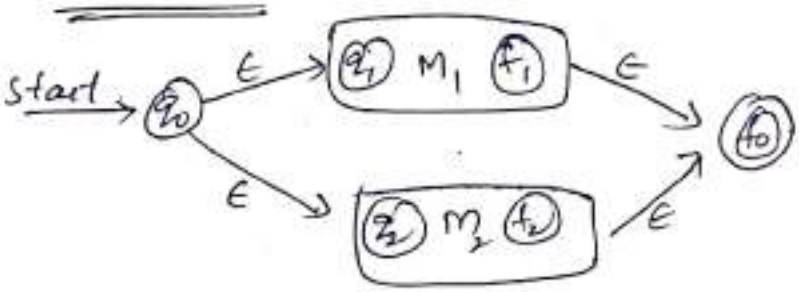
Here dot stands for matching with one character.

Conversion of Regular Expression to Finite Automata

In any type of regular expression there are only three cases possible.

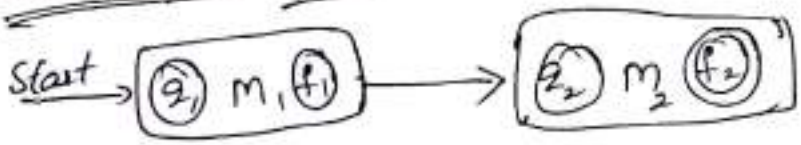
1. Union 2. Concatenation 3. Closure.

Case 1: Union Case.



The construction of machine M is the transition from q_0 to f_0 must begin by going to q_1 or q_2 on ϵ. If the path goes to q_1, then it follows the path in machine M_1 and goes to the state f_1 and then to f_0 on ϵ. Similarly, if the path goes to q_2, then it follows the path in machine M_2 and goes to state f_2 and then to f_0 on ϵ. Thus the $L(M) = L(M_1) \cup L(M_2)$ i.e. either the path in machine M_1 or M_2 will be followed.

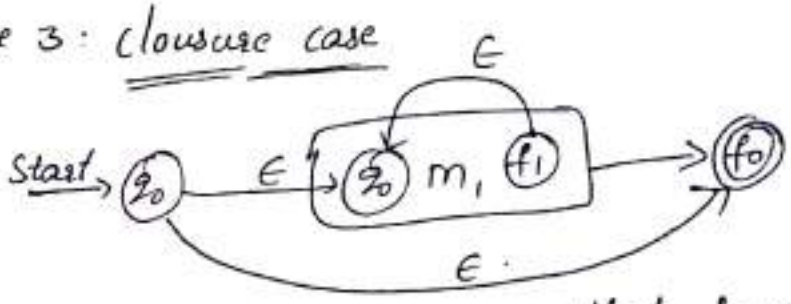
Case 2: Concatenation Case



The initial state is q_1 by some input a the next state will be f_1. And on receiving ϵ the transition will be from f_1 to q_2 and the final state will be f_2. The transition from q_2 to f_2 will be on receiving some input b.

$L(M) = ab$, $L(M) = L(M_1) . L(M_2)$

Case 3: Clousure case



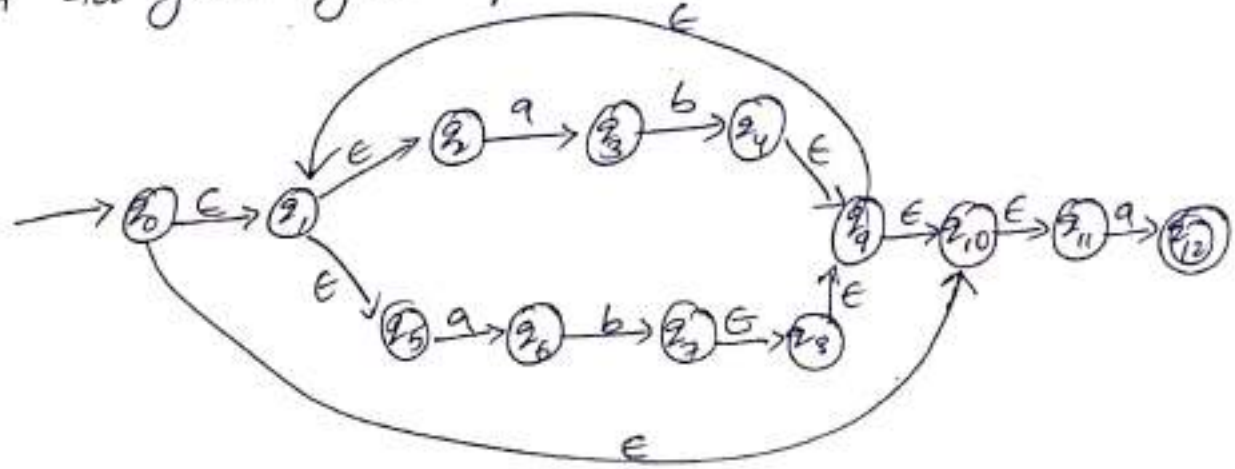
The machine M_1 shows that from q_0 to q_1 there is a transition on receiving ε Similarly, from q_0 to f_0 on ε there is a path. The path exists from f_1 to q_1, a back path. Similarly a transition from f_1 to f_0 final state, on receiving ε. The total recursion is possible. Thus one can derive ε, a, aa, aaa, ... for the input a.

Thus $L(M) = L(M_1)^*$ is Proved.

① Construct DFA for the following regular expression $(abUaba)^*a$

Ans: R.E = $(abUaba)^*a$

NFA for given regular expression



Now will find ε-closure for q_0 state

ε-closure(q_0) = $\{q_0, q_1, q_2, q_5, q_{10}, q_{11}\}$ → call it as state A.

Now we will apply input transitions on state A.

$\delta'(A,a) = \epsilon\text{-closure}(\delta(AA))$

$= \epsilon\text{-closure}(\delta(q_0,q_1,q_2,q_5,q_{10},q_{11}),a)$

$= \epsilon\text{-closure}(\delta(q_0,a) \cup \delta(q_1,A) \cup \delta(q_3,A) \cup \delta(q_5,a) \cup \delta(q_{10},a) \cup \delta(q_{11},a))$

$= \epsilon\text{-closure}(\phi \cup \phi \cup q_3 \cup q_6 \cup \phi \cup q_{12})$

$= \epsilon\text{-closure}(q_3) \cup \epsilon\text{-closure}(q_6) \cup \epsilon\text{-closure}(q_{12})$

$= \{q_3\} \cup \{q_6\} \cup \{q_{12}\}$

$= \{q_3, q_6, q_{12}\} \longrightarrow$ call it as state B.

$\delta'(A,a) = B.$

$\delta'(A,b) = \epsilon\text{-closure}(\delta(A,b))$

$= \epsilon\text{-closure}(\delta(q_0,q_1,q_2,q_5,q_{10},q_{11}),b)$

$= \epsilon\text{-closure}(\delta(q_0,b) \cup \delta(q_1,b) \cup \delta(q_2,b) \cup \delta(q_5,b) \cup \delta(q_{10},b) \cup \delta(q_{11},b))$

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

$= \phi$

$\therefore \delta'(A,b) = \phi$

now apply δ transitions on state B.

$\delta'(B,a) = \epsilon\text{-closure}(\delta(B,a))$

$= \epsilon\text{-closure}(\delta(q_3,q_6,q_{12}),a)$

$= \epsilon\text{-closure}(\delta(q_3,a) \cup \delta(q_6,a) \cup \delta(q_{12},a))$

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

$\delta'(B,a) = \phi$

$\delta'(B,b) = \epsilon\text{-closure}(\delta(B,a))$

$= \epsilon\text{-closure}(\delta(q_3,q_6,q_{12}),b)$

$= \epsilon\text{-closure}(\delta(q_3,b) \cup \delta(q_6,b) \cup \delta(q_{12},b))$

$= \epsilon\text{-closure}(q_4 \cup q_7 \cup \phi)$

$= \epsilon\text{-closure}(q_4) \cup \epsilon\text{-closure}(q_7)$

$= \{q_4, q_9, q_1, q_2, q_5, q_{10}, q_{11}\} \cup \{q_7\}$

$= \{q_1, q_2, q_4, q_5, q_7, q_9, q_{10}, q_{11}\} \rightarrow$ call it as C.

Now apply δ transitions on state C.

$\delta'(C, a) = \epsilon\text{-closure}(\delta(C, a))$

$= \epsilon\text{-closure}(\delta(\{q_1, q_2, q_4, q_5, q_7, q_9, q_{10}, q_{11}), a)$

$= \epsilon\text{-closure}(\delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_4, a) \cup \delta(q_5, a) \cup \delta(q_7, a) \cup \delta(q_9, a) \cup \delta(q_{10}, a) \cup \delta(q_{11}, a))$

$= \epsilon\text{-closure}(q_3, q_6, q_8, q_{12})$

$= \epsilon\text{-closure}(q_3) \cup \epsilon\text{-closure}(q_6) \cup \epsilon\text{-closure}(q_8) \cup \epsilon\text{-closure}(q_{12})$

$= \{q_3\} \cup \{q_6\} \cup \{q_8, q_9, q_{10}, q_{11}, q_1, q_2, q_5\} \cup \{q_{12}\}$

$= \{q_1, q_2, q_3, q_5, q_6, q_8, q_9, q_{10}, q_{11}, q_{12}\} \rightarrow$ call it as D.

$\delta'(C, a) = D$

$\delta'(C, b) = \epsilon\text{-closure}(\delta(C, b))$

$= \epsilon\text{-closure}(\delta(q_1, q_2, q_4, q_5, q_7, q_8, q_{10}, q_{11}), b)$

$\epsilon\text{-closure}(\delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_4, b) \cup \delta(q_5, b) \cup \delta(q_7, b) \cup \delta(q_8, b) \cup \delta(q_{10}, b) \cup \delta(q_{11}, b))$

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

$\delta'(C, b) = \phi$

$\delta'(D, a) = \epsilon\text{-closure}(\delta(q_1, q_2, q_3, q_5, q_6, q_8, q_9, q_{10}, q_{11}, q_{12}), a)$

$= \epsilon\text{-closure}(\delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) \cup \delta(q_5, a) \cup \delta(q_6, a) \cup \delta(q_8, a) \cup \delta(q_9, a) \cup \delta(q_{10}, a) \cup \delta(q_{11}, a) \cup \delta(q_{12}, a))$

$= \epsilon\text{-closure}(\phi \cup q_3 \cup \phi \cup q_6 \cup \phi \cup \phi \cup \phi \cup \phi \cup q_{12} \cup \phi)$

$= \epsilon\text{-closure}(q_3, q_6, q_{12})$

$= \epsilon\text{-closure}(q_3) \cup \epsilon\text{-closure}(q_6) \cup \epsilon\text{-closure}(q_{12})$

$= \{q_3, q_6, q_{12}\}$ i.e. State B.

$\delta'(D, a) = B$

$\delta'(D, b) = \epsilon\text{-closure}(\delta(q_1, q_2, q_3, q_5, q_6, q_8, q_9, q_{10}, q_{11}, q_{12}), b)$

$= \epsilon\text{-closure}(\delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_3, b) \cup \delta(q_5, b) \cup \delta(q_6, b) \cup \delta(q_8, b) \cup \delta(q_9, b) \cup \delta(q_{10}, b) \cup \delta(q_{11}, b) \cup \delta(q_{12}, b))$

$= \epsilon\text{-closure}(\phi \cup \phi \cup q_4 \cup \phi \cup q_7 \cup \phi \cup \phi \cup \phi \cup \phi \cup \phi)$

$= \epsilon\text{-closure}(q_4, q_7)$

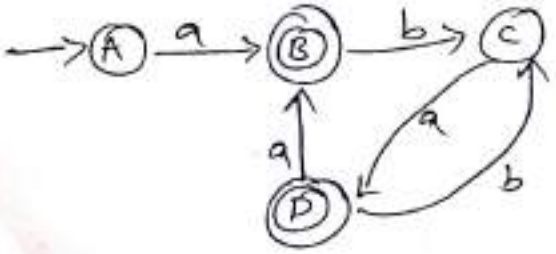
$= \epsilon\text{-closure}(q_4) \cup \epsilon\text{-closure}(q_7)$

= i.e. State C.

$\delta'(D, b) = C$.

State I/P	a	b
A	B	ϕ
B	ϕ	C
C	D	ϕ
D	B	C

transition diagram for DFA will be



(2) Convert the following regular expression into NFA: $[[(00)^*(11)] \cup 01]^*$

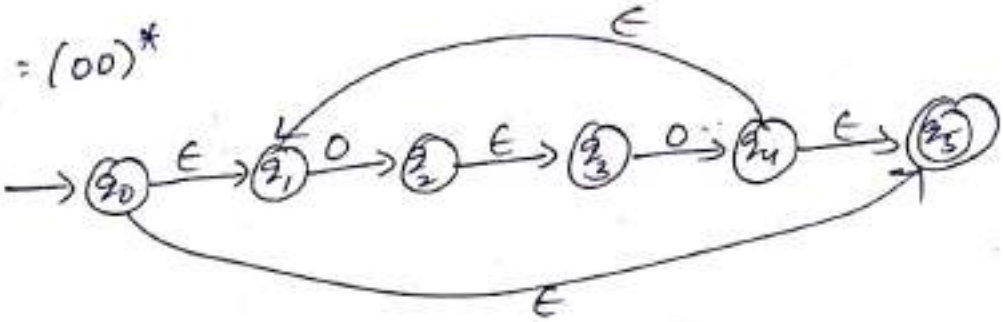
Sol:- Divide the given R.E into smaller regular expressions.

$$R.E = (((00)^*(11)) \cup 01)^*$$

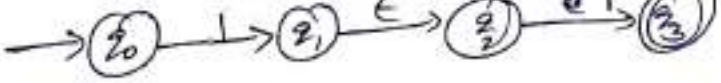
r_1 r_2 r_4

r_3

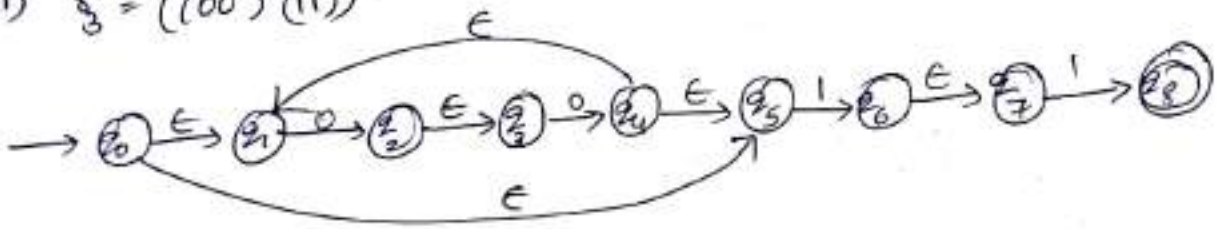
(i) $r_1 = (00)^*$



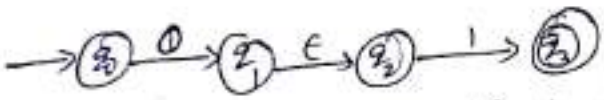
(ii) $r_2 = 11$



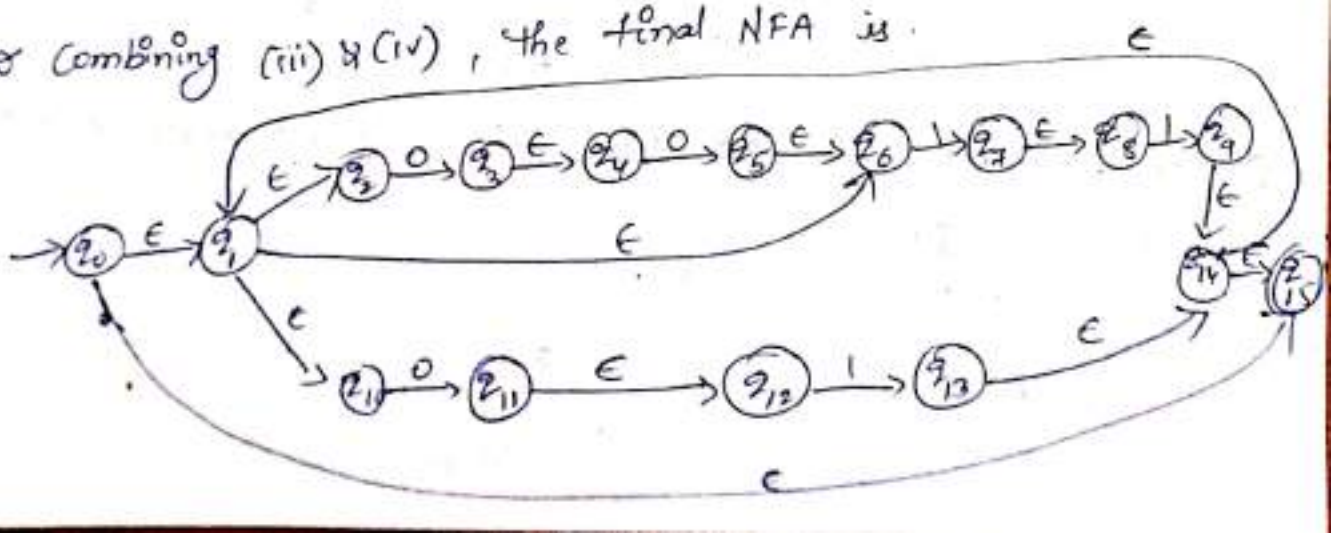
(iii) $r_3 = ((00)^*(11))$



(iv) $r_4 = 01$

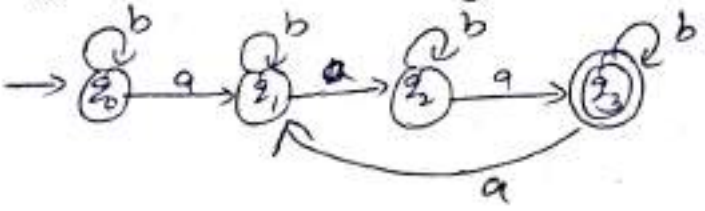


Now combining (iii) & (iv), the final NFA is



(3) Design FA to accept string with 'a' and 'b' such that the number of a's are divisible by 3.

Ans



(4) Design NFA that accepts the language $(aa^*(a+b)^*)$

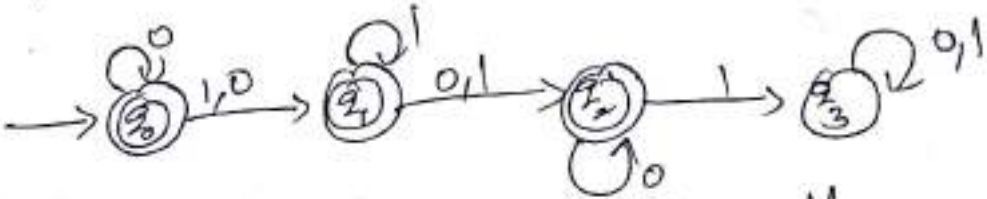
Ans:

(5) Construct DFA and NFA accepting the set of all strings not containing 101 as a substring.

Ans:- First design FA for the strings containing 101 as a sub string.

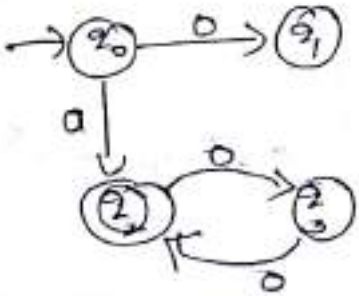
Now for accepting the language that does not containing the substring 101, just change the non-final states into final state and final state into non-final. The DFA will be

The NFA will be



(6) Construct NFA for R.E that contains odd number of 0s over $\Sigma = \{0\}$.

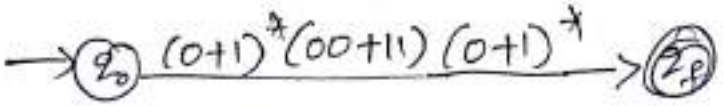
$R.E = 0(00)^*$



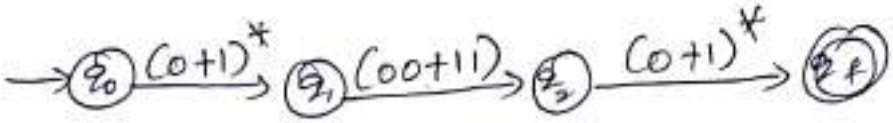
NFA is

(7) Construct finite automation to accept the regular expression $(0+1)^*(00+11)(0+1)^*$.

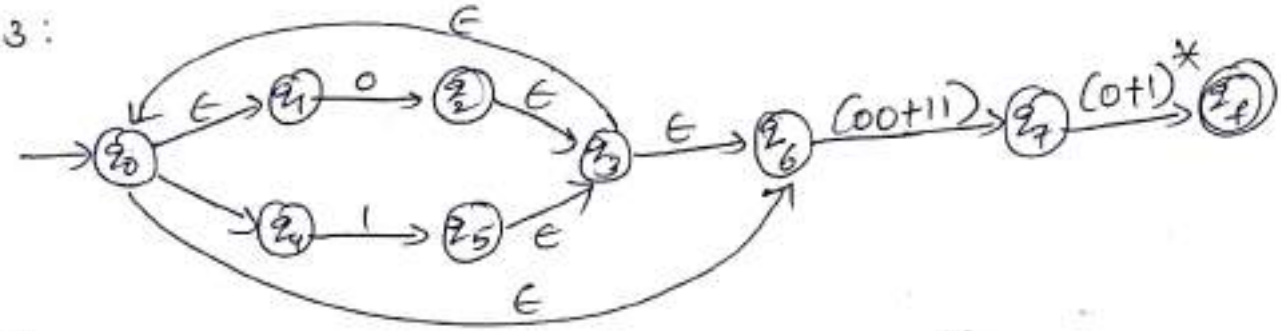
Ans: Step 1 : The FA for r.e $= (0+1)^*(00+11)(0+1)^*$



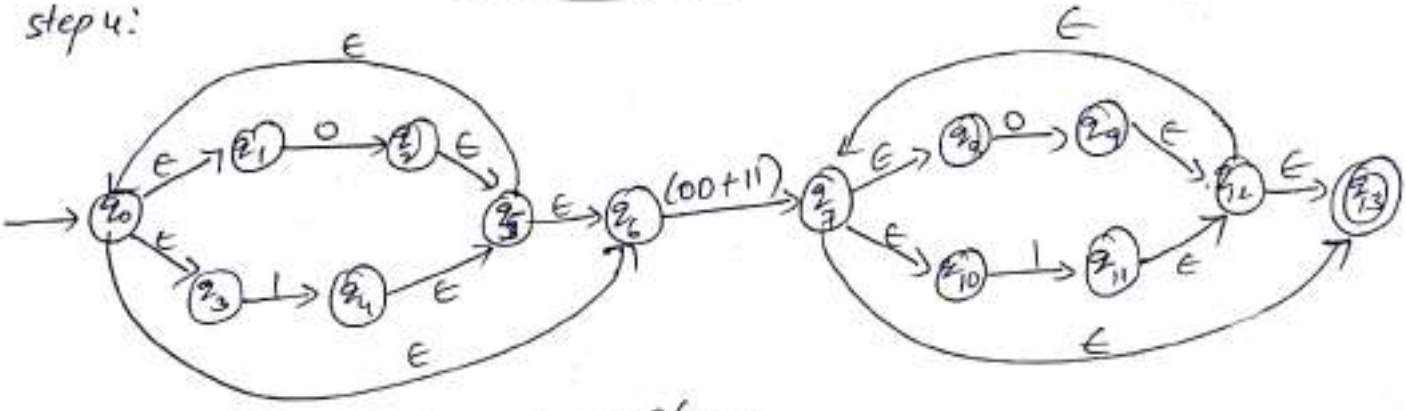
Step 2 :



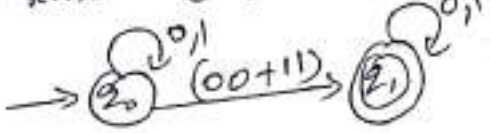
Step 3 :



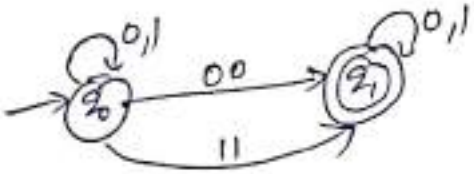
step 4:



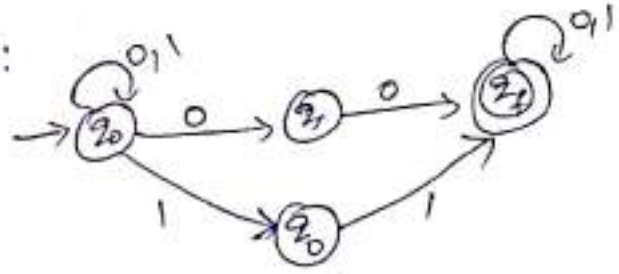
Step 5 : Now remove ϵ-transitions



Step 6 :



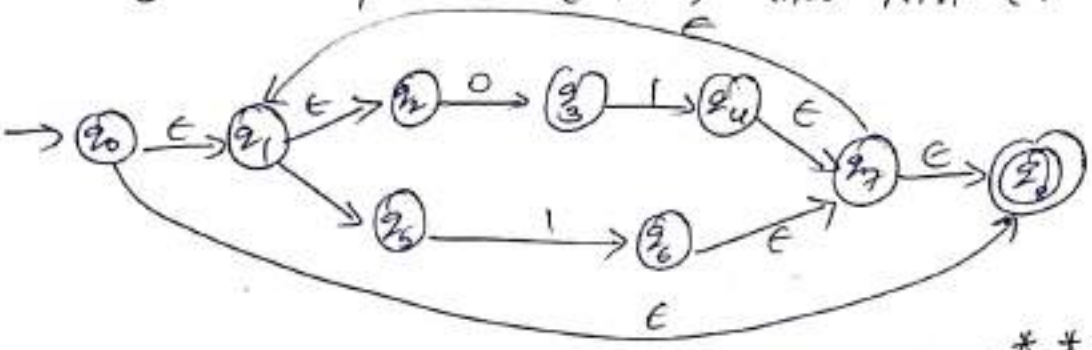
step 7:



∴ Required FA.

(8) Convert the regular expression $(0+1)^*$ into NFA-ϵ.

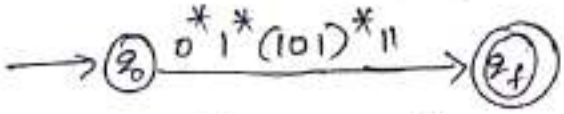
Ans:



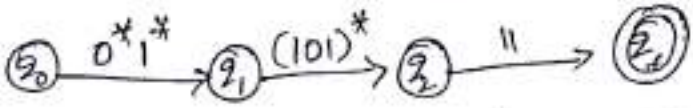
(9) Construct Finite Automata for the regular expression $0^*1^*(101)^*11$.

Ans:

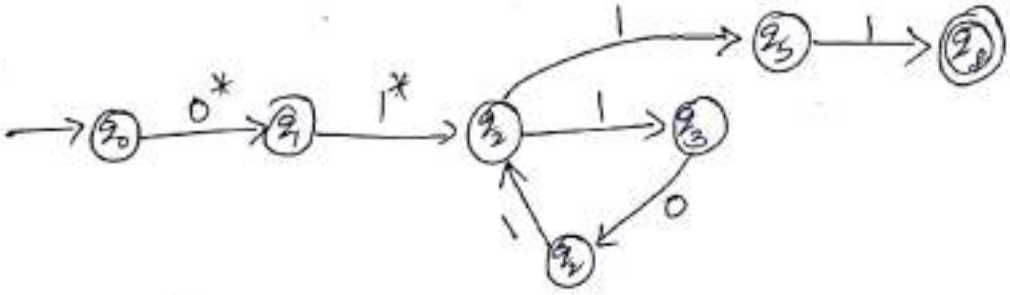
Step 1:



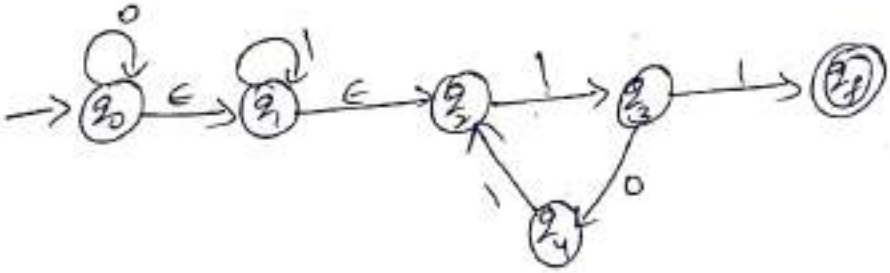
Step 2:



Step 3:



Step 4:



(10) Construct a DFA for the regular language consisting of any number of a's and b's.

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

Ans:-

DFA will be

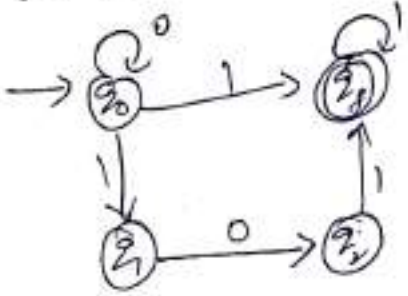
(11) Design FA for the following languages

a) $(0^*1^*)^*$ (b) $(0+1)^*111^*$ c) (0^*11^*+101)

Ans:- (a) $(0^*1^*)^*$

(b) $(0+1)^*111^*$

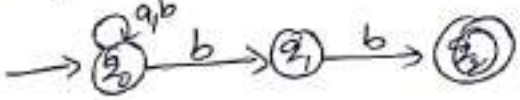
(c) (0^*11^*+101)



(12) Design NFA to accept strings with a's and b's such that the string end with bb.

Ans: R.E $= (a+b)^*bb$

NFA is



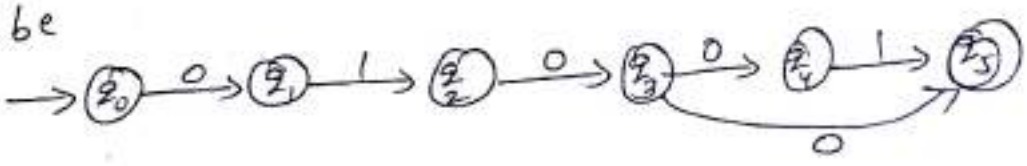
(13) $L_1 = \{010\}$, $L_2 = \{01, 0\}$, then find (i) L_1L_2 (ii) L_1^*

(iii) L_2^+ (iv) $L_1^* + L_2^*$

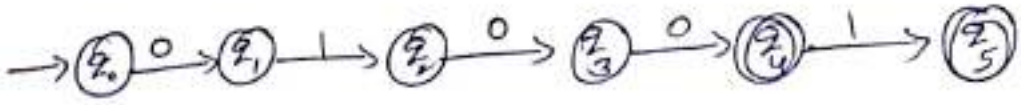
Ans:- (i) For L_1L_2 the regular expression is $\{010\}\{01+0\}$

$= (010)(01+0)$

FA can be

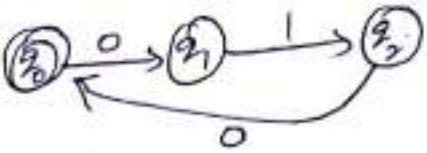


i.e.

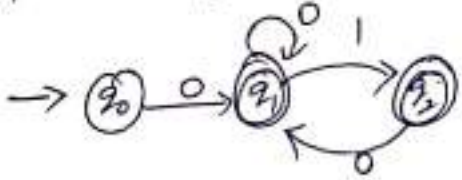


(ii) L_1^*, r.e = $(010)^*$

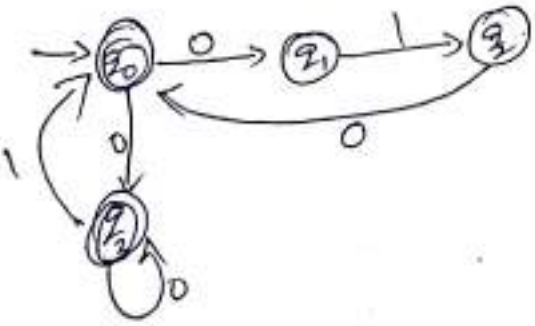
FA can be →



(iii) L_2^+, r.e = $(0+01)^+$



(iv) $L_1^* + L_2^*$ R.E = $(01)^* + (01+0)^*$



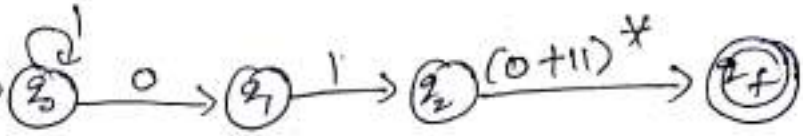
(14) DFA accepts all strings corresponding to expression $1^*01(0+11)^*$. Also explain how to convert a regular expression to DFA.

Ans:- DFA for $1^*01(0+11)^*$

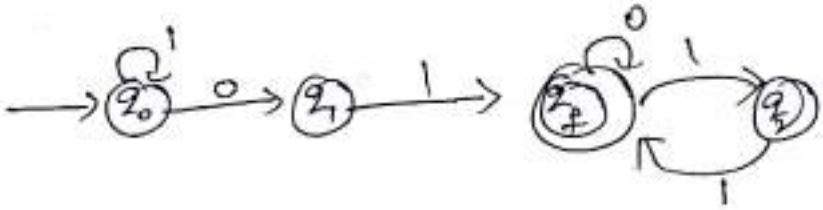
Step 1: → $\xrightarrow{1^*01(0+11)^*}$

Step 2: → $\xrightarrow{1^*}$ $\xrightarrow{01}$ $\xrightarrow{(0+11)^*}$

Step 3: →



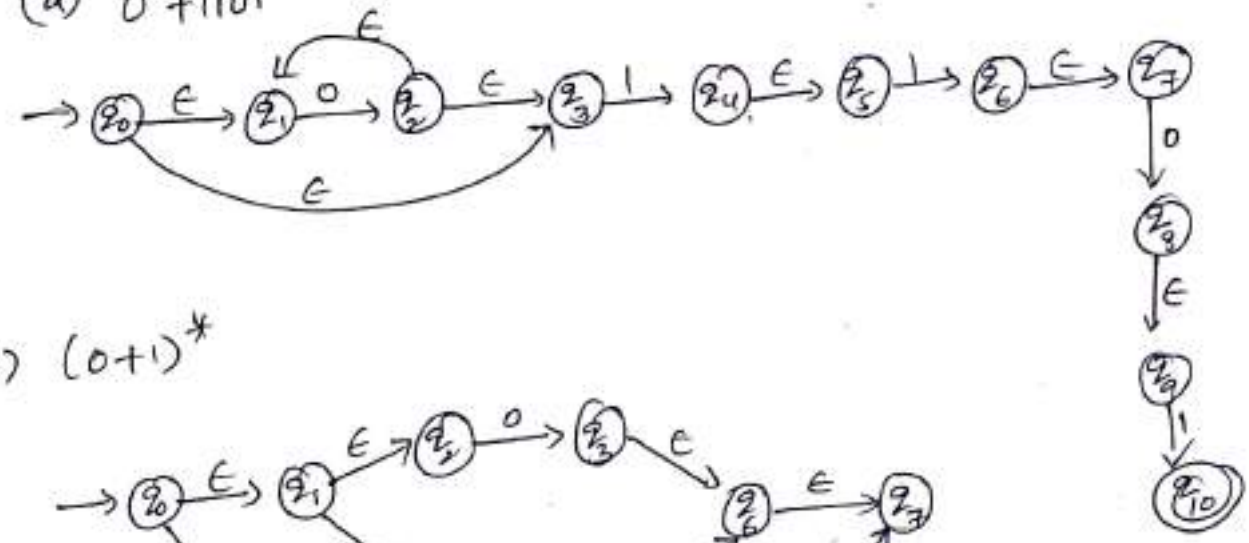
Step 4: →



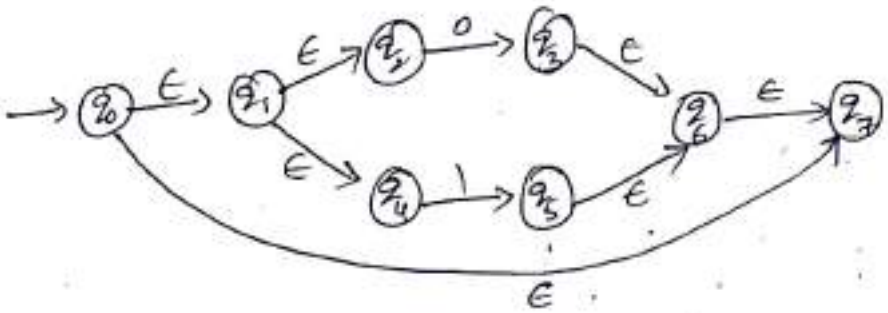
(15) Convert the following regular expression to NFA with epsilon transition.

(a) $0^*+(1101)$ (b) $(0+1)^*$

Ans:- (a) 0^*+1101

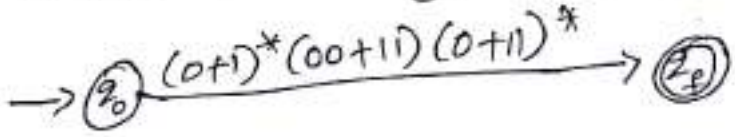


(b) $(0+1)^*$

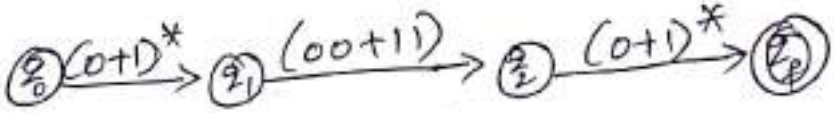


(16) Construct a DFA for the regular expression $(0+1)^*(00+11)(0+1)^*$

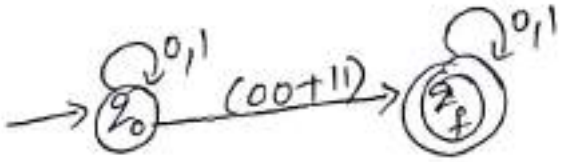
Ans:- Step 1 :



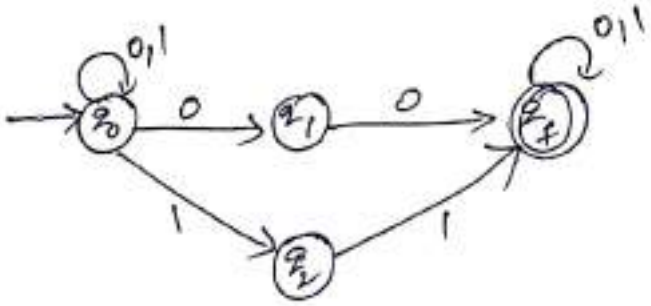
Step 2 :



Step 3 :



Step 4 :



step 5: Now Convert above NFA to DFA.

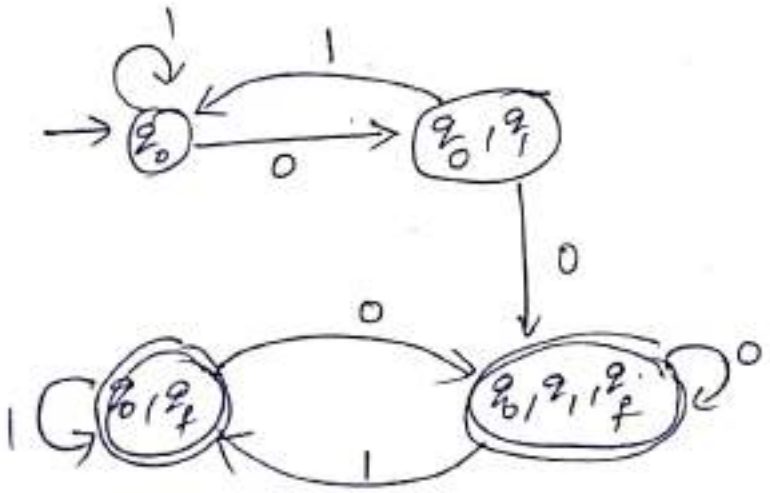
transition table for NFA

δ	0	1
→ q_0	$\{q_0, q_1\}$	q_0
q_1	q_f	ϕ
q_2	ϕ	q_f
(q_f)	q_f	q_f

transition table for DFA

δ	0	1
→ q_0	$\{q_0, q_1\}$	q_0
$\{q_0, q_1\}$	$\{q_0, q_1, q_f\}$	q_0
$\{q_0, q_1, q_f\}$	$\{q_0, q_1, q_f\}$	$\{q_0, q_f\}$
$\{q_0, q_f\}$	$\{q_0, q_1, q_f\}$	$\{q_0, q_f\}$

DFA is



Arden's Theorem

Let P and Q be the two regular expressions over the input set Σ. The regular expression R is given as $R = Q + RP$ which has a unique solution as $R = QP^*$

Proof: P and Q are 2 R.E's and P doesn't contains ϵ then

given $R = Q + RP$

Substitute $R = QP^*$ in the RHS

$$R = Q + QP^* \cdot P$$

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

$$R = QP^*$$

To Prove it has a unique solution substitute $R = Q + RP$ in RHS

$$R = Q + RP$$

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

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

$$= Q + QP + (Q + RP)P^2$$

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

$$\vdots$$

$$= Q + QP + QP^2 + \cdots QP^k + RP^{k+1}$$

$$= Q(\epsilon + P + P^2 + \cdots P^k) + RP^{k+1}$$

If we consider a string with length 'k' it is not accepted / belongs to RP^{k+1}

So, R and QP^* are belongs to same set then conclude

$$\boxed{R = QP^*}$$

Conversion of FA to Regular Expression

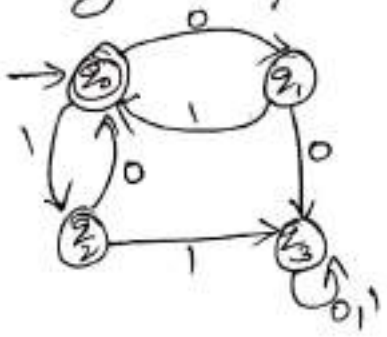
* Consider a FA machine then our objective is construct an equivalent Regular expression using Arden's theorm.

(i) Construct state equations for all the states based on incoming transitions.

(ii) If It is a initial state we need to add `ε' also

(iii) To eliminate the states by using Arden's theorm.

① Construct Consider the Finite Automata and Construct equivalent Regular Expression.



Ans $q_0 = \epsilon + q_1 \cdot 1 + q_2 \cdot 0$

$q_1 = q_0 \cdot 0$

$q_2 = q_0 \cdot 1$

$q_3 = q_3 \cdot 0 + q_3 \cdot 1 + q_2 \cdot 1 + q_1 \cdot 0$

Consider q_0

$q_0 = q_1 \cdot 1 + q_2 \cdot 0 + \epsilon$

$= q_0 \cdot 01 + q_0 \cdot 10 + \epsilon$

$q_0 = q_0(01+10) + \epsilon$

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

$q_0 = \epsilon(01+10)^*$

$q_0 = (01+10)^*$

(2) Consider FA and construct RE

Ans: $q_1 = q_1 \cdot 1 + q_2 \cdot 1 + \epsilon$

$q_2 = q_1 \cdot 0$

$q_3 = q_2 \cdot 0 + q_3 \cdot 0 + q_3 \cdot 1$

Consider $q_3 = q_2 \cdot 0 + q_3 \cdot 0 + q_3 \cdot 1$

$q_3 = q_1 \cdot 00 + q_3 \cdot 0 + q_3 \cdot 1$

$q_3 = q_1 \cdot 00 + q_3(0+1)$

consider

$q_3 = (1+01)^* 00 + q_3(0+1)$

R Q R P

$q_3 = (1+01)^* 00 (0+1)^*$

consider q_1

$q_1 = q_1 \cdot 1 + q_2 \cdot 1 + \epsilon$

$q_1 = q_1 1 + q_1 01 + \epsilon$

$q_1 = q_1(1+01) + \epsilon$

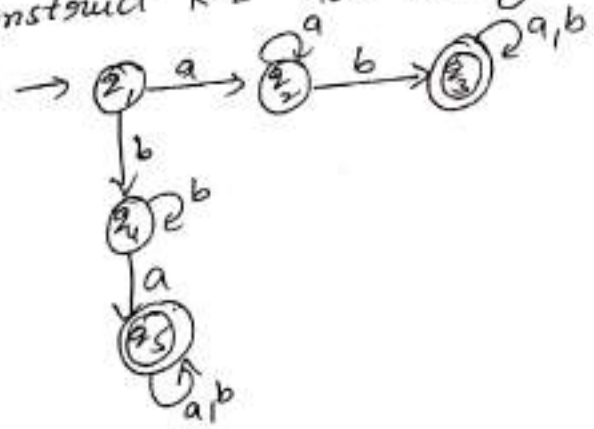
R R P Q

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

$q_1 = (1+01)^*$

sub q_1 in q_3

(3) Construct R.E for the given FA.



Ans: $q_1 = \epsilon$

$q_2 = q_1 \cdot a + q_2 \cdot a$

$q_3 = q_2 \cdot b + q_3(a+b)$

$q_4 = q_1 \cdot b + q_4 \cdot b$

$q_5 = q_4 \cdot a + q_5(a+b)$

Consider q_3

$q_3 = q_2 \cdot b + q_3(a+b)$

$q_3 = q_1 ab + q_2 ab + q_3(a+b)$

Consider q_2

$q_2 = q_1 a + q_2 a$

$q_2 = a + q_2 a$

$q_2 = aa^* = a^*$

$q_3 = q_1 ab + q_2 ab + q_3(a+b)$

$q_3 = ab + aa^* ab + q_3(a+b)$

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

$q_3 R = (ab + aa^* b)(a+b)^*$

$q_5 = q_4 \cdot a + q_5(a+b)$

$q_5 = q_1 b + q_4 b + q_5(a+b)$

Consider q_4

$q_4 = q_1 b + q_4 b$

$q_4 = b + q_4 b$

$q_4 = bb^* = b^*$

$q_5 = q_4 b + q_4 b + q_5(a+b)$

$= bb^* + b + q_5(a+b)$

$q_5 = (b^* + b) + q_5(a+b)$

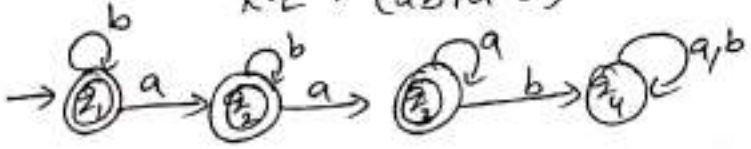
$R \quad\quad Q \quad\quad R \quad P$

$q_5 = (b^* + b)(a+b)^*$

$\therefore R.E = q_3 + q_5$

$R.E = (ab + a^* b)(a+b)^* + (b^* + b)(a+b)^*$

(4)



Ans:

$q_1 = q_1 b + \epsilon$

$q_2 = q_2 b + q_1 a$

$q_3 = q_3 a + q_2 a$

$q_4 = q_4(a+b) + q_3 b$

Consider q_1

$q_1 = q_1 b + \epsilon$

$q_1 = \epsilon(b)^*$

$q_1 = b^*$

Consider q_2

$q_2 = q_2 b + q_1 a$

$q_2 = q_2 b + b^* a$

$q_2 = b^* a b^*$

Consider q_3

$q_3 = q_2 b + q_1 ba + \epsilon$

$q_3 = q_2 b + b^* ba + \epsilon$

$q_3 = q_2 b + b^* ba + \epsilon$

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

Consider q_2

$q_2 = q_2 b + q_1 a$

$q_2 = q_2 b + b^* a$

$q_2 = b^* a b^*$

Consider q_3

$q_3 = q_3 a + q_2 a$

$= q_3 a + (q_1 a + q_2 b) a$

$q_3 = q_3 a + q_1 aa + q_2 ba$

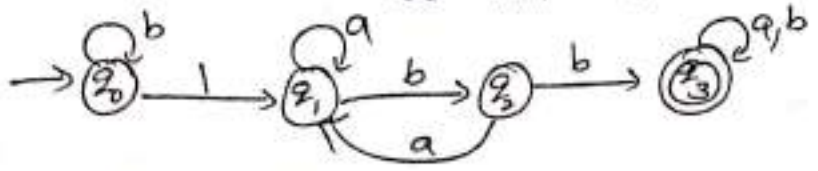
$q_3 = q_3 a + b^* aa + b^* ab^* ba$

$q_3 = (b^* aa + b^* ab^* ba) a^*$

$\therefore$ R.E $= q_1 + q_2 + q_3$

R.E $= b^* + b^*ab^* + ((b^*aa + b^*ab^*)ba)a^*$

(5)



Ans:

$q_0 = q_0 b + \epsilon$

$q_1 = q_1 a + q_0 a + q_2 a$

$q_2 = q_1 b$

$q_3 = q_3 (a+b) + q_2 \cdot b$

Consider q_3

$q_3 = q_3(a+b) + q_2 b$

$q_3 = q_3(a+b) + q_1 bb$

Consider q_1 =

$q_1 = q_1 a + q_0 a + q_2 a$

$q_1 = q_1 a + b^* a + q_1 ba$

$q_1 = b^* a + q_1 (a + ba)$

$q_1 = b^* a (a+ba)^*$

$q_0 = q_0 b + \epsilon$

$q_0 = \epsilon b^*$

$q_0 = b^*$

$q_2 = q_1 b$

sub q_1 in q_3

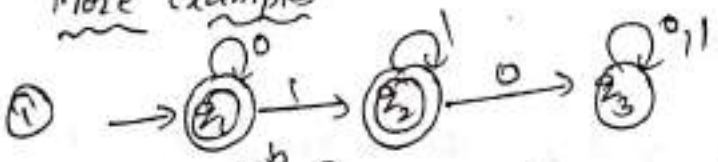
$q_3 = q_3(a+b) + q_1 bb$

$q_3 = q_3 (a+b) + b^* a (a+ba)^* bb$

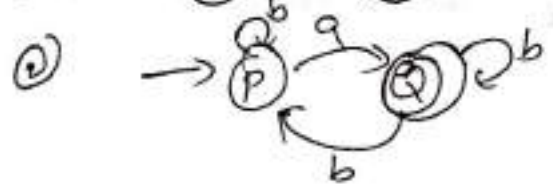
$\boxed{q_3 = b^* a (a+ba)^* bb (a+b)^*}$

More examples

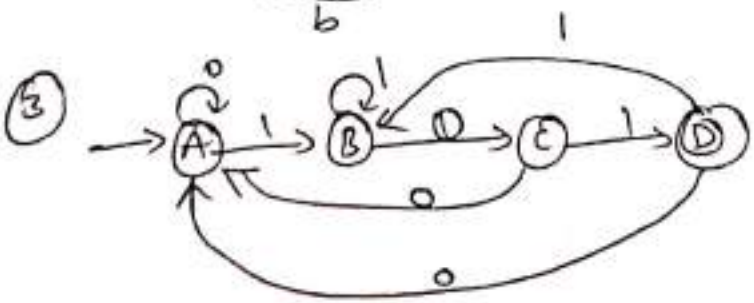
(1)



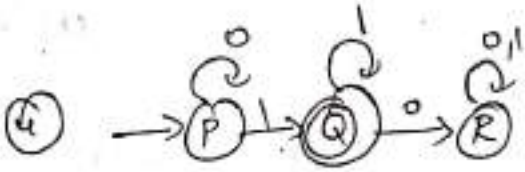
(2)



(3)



(4)



Pumping lemma for Regular Language

→ Pumping lemma is useful for showing the given language is not Regular language.

→ Consider a FA and a string 'w', if we increase the length of the string on the middle portion of the resultant string is also accepted by Finite Automata then we conclude it as a Regular language.

Statement for Pumping lemma

⇒ Let 'L' be the regular language and n is a positive Integer constant and 'z' is any string in 'L'. Then we can say 'L' is regular if it is satisfies the following properties

→ Divide z into 3 parts

$z = uvw$, the length $|z| \geq n$

→ The length of uv (or) $|uv| \leq n$ and $|v| \geq 1$

→ Pump some stuff from the middle of the word

$uv^i w$, $i \geq 1$

→ The resultant string is $\in L$ then conclude the given language is Regular.

① Prove $L = \{a^n b^n / n \geq 1\}$ is not regular.

$L = \{ab, aabb, aaabbb, \ldots\}$

$L = \{a^n b^n\}$ $\quad |uv| \leq n \quad |v| \geq 1$

$z = a^n b^n$ $\quad |a^{n-1}.a| \leq n \quad |1| \geq 1$

$z = a^{n-1} a\, b^n$ $\quad |n-1+1| \leq n$

$\underline{u}\ \underline{v}\ \underline{w}$ $\quad |n| \leq n$

* $uv^i w$; $i = 2$

$= a^{n-1}.a^2.b^n$

$= a^{n-1+2}.b^n$

$= a^{n+1}.b^n$

∴ L is not a regular Language.

② Prove $L = \{ab, aabb, \ldots\}$ $n=3$

$z = aabb$ $\quad i=2$

$\underline{u}\ \underline{v}\ \underline{w}$ $\quad |aab| \leq 3$ ✓

$\quad |ab| \geq 1$ ✓

$uv^i w$

$= a(ab)^2 b$

$= aababb \notin L$

∴ z is not a regular language.

③ $L = \{a^p / p$ is prime$\}$ is not regular.

$L = \{aa, aaa, aaaaa, aaaaaaa, \ldots\}$

$n=3$ $\quad |aa| \leq 3$

$z = a|a|a$ $\quad |a| \geq 1$

$u\ v\ w$

$uv^i w = a(a^2)a$

$= aaaa \notin L$. is not regular

④ State a language $L = \{0^{n^2} / n \geq 0\}$ is not regular.

$L = \{\epsilon, 0, 0000, 00000000 \cdots\}$

$n = 2$

$z = \underset{u}{0} \underset{v}{00} \underset{w}{0}$ $|uv| \leq 1$ ✗.

$z = \underset{u}{0} \underset{v}{00} \underset{w}{00}$ $|uv| \leq 2$, $|v| \geq 1$

$uv^i w = 0(0)^2 00$

$= 00000 \notin L$, is not regular.

⑤ State and prove a language of balanced parantheses is not regular.

$L = \{(), (()), ()(), (()())\}$

$z = \underset{u}{(} \underset{v}{(} \underset{w}{)})$ $|uv| \leq 2$, $|v| \geq 1$

$uv^i w = (()^2))$

$= (((())) \notin L$. is not regular.

Applications of R.E.

* Data validation
* Text Processing
* String processing
* Data Separating.
* Data Wrangling.

Closure Properties of Regular language:

→ If R is Regular language. If we apply any operation on 'R' then the result is also a regular language then we can conclude those operations is called closure properties.

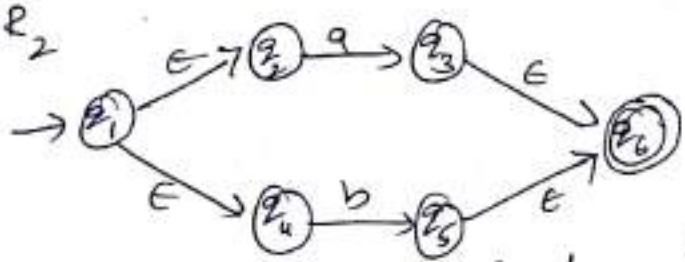
R.L are closed under

1. Union
2. Concatenation
3. Kleenclosure
4. Compliment
5. Intersection
6. Transpose / Reversal

Union:- Let R_1 and R_2 are two regular languages then $R_1 \cup R_2$ is also a Regular language.

Consider $R_1 = \{a\}$, $R_2 = \{b\}$

Union = $R_1 + R_2$



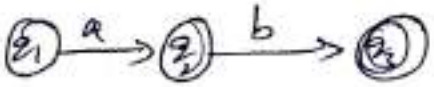
∴ Regular Languages are closed under union.

Concatination:-

Let R_1 & R_2 are two Regular languages then $R_1 \cdot R_2$ is also a regular language.

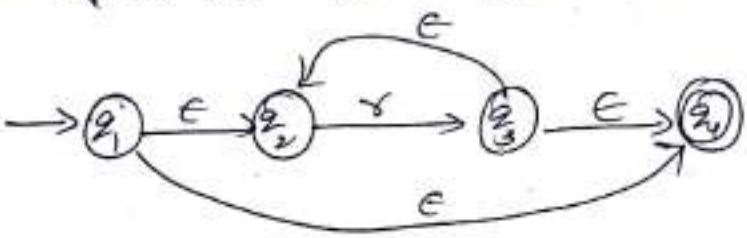
Consider $R_1 = \{a\}$, $R_2 = \{b\}$

$R_1 \cdot R_2 =$



Kleen closure.

Consider r_1 is R.L. then kleen closure of r^* is r^* is RL.



Complement:

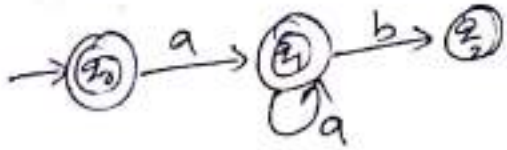
If R is a R.L then the complement of R^1 is also a regular language.

→ Consider a language L accepted by FA then a complement of L is going to accept $\bar{L} = \Sigma^* - L$

→ To Construct FA for Complement use following rules.

1. change all final states into non final states and non-final state into final state.
2. There is no change in the initial state.

Construct FA for not accepting substring ab



(5) Reversal / Transpose

If L is RL then the reversal of L is L^R is also a regular language.

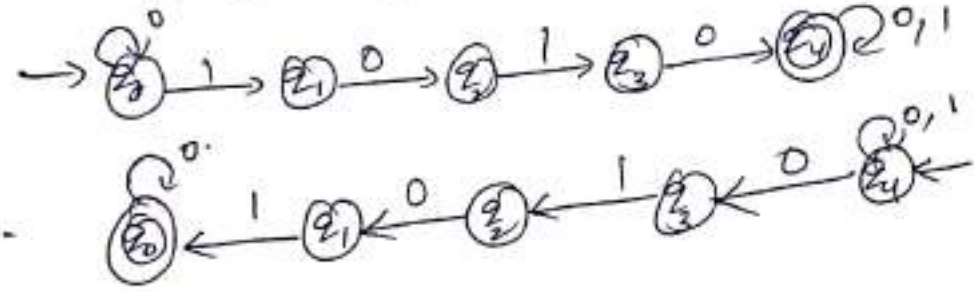
→ To construct FA for L^R use the following rules

(1) Make initial state as final state

(2) Reverse all the direction on edges / transitions

(3) Consider a new state P_0 as start with final state.

Construct FA which contains a substring 1010



Intersection

If L_1 & L_2 are two regular languages $R_1 \cap R_2$ is also a regular language.

$\therefore R_1 = (a+b)^*$, $R_2 = (a+b)^*b$

$R_1 \cap R_2 = (a+b)^*b$

Applications of R.E.

* Pattern Matching
* Lexical Analyzer
* Data validation
* Find & replace

Mail R.E is

→ letter.(letter+digit)* @letter.+ [com+in]