Date / /20

Q5) Obtain R.E to accept strings of a's and b's of length ≤ 2

Soln: Strings of a's and b's of length ≤ 2 can be written as:

$\epsilon, a, b, ab, ba, aa, bb$

i.e $\epsilon + a + b + aa + ab + ba + bb$

above strings can be obtained using Regular expression

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

ϵ, a, b

i.e The Regular Expression is

$$(\epsilon + a + b)^2$$

Q6) R.E to accept strings of a's and b's of length ≤ 10

→ i.e $(\epsilon + a + b)^{10}$

Q7) a) R.E to accept the language $L(R) = \{w \mid w \in \{0,1\}^*$ with 3 consecutive 0's.

$$(1)^* 000 1^*$$

b) R.E to accept the language $L(R) = \{w \mid w \in \{0,1\}^*$ with atleast 3 consecutive 0's

$$(0+1)^* 000 (0+1)^*$$

Q8) R.E to accept 0's and 1's having no two consecutive zeroes.

Soln: No two consecutive zeros means we can have ϵ, 0, but not two or more zeros together. According to the question there is no restriction for the number of 1's.

— So you can have ϵ, single 0, 0 followed by any one. One, any number of 1's,

Any number of 1's or any number of 01's

$(1+01)^*$

It may have single zero or ϵ

i.e $0+\epsilon$

i.e The R.E is

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

Q9) R.E to accept strings of a's and b's whose second symbol from the right end is a

Soln: That means last symbol can be either an 'a' or a 'b'. ∴ the R.E is

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

Q10) Obtain R.E to accept strings of a's and b's such that third symbol from the right is a and fourth symbol from the right is b.

→ $(a+b)^* b a (a+b)(a+b)$

Q11) Obtain R.E to accept the words with two or more letters but beginning and ending with the same letter where $\Sigma = \{a, b\}$

→ String beginning and ending with the same let

a

$a(a+b)^* a$

String beginning and ending with the same letter

$b(a+b)^* b$

∴ R.E is $\boxed{a(a+b)^* a + b(a+b)^* b}$

Before Q12) Solve Q15)

Q12) Obtain R.E to accept strings of a's and b's whose length is either even or multiples of 3 or both.
(multiples of 2)

Soln: R.E whose length is even is

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

R.E whose length is multiple of 3 can be obtained as

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

∴ R.E is $((a+b)(a+b))^* + ((a+b)(a+b)(a+b))^*$

Q13) a) Obtain R.E for the set of all strings that do not end with 01 over $\{0,1\}^*$

→ So the strings with length 2 and that do not end with 01 are 00, 10, 11

0 1 ✗ 1 1 if last 1 then 0 0 1

$\boxed{(0+1)^*(00+10+11)}$

b) Do not end with ab

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

Q14) Obtain R.E to accept strings of a's and b's to accept even no/. of a's followed by odd no/. of b's.

$(aa)^* b (bb)^*$

(OR) Question can be Obtain R.E for the following language assuming $\Sigma = \{0,1\}$

$L = \{a^{2n} b^{2m+1} \mid n \geq 0, m \geq 0\}$

Note: Even a's followed by Even b's (or) $L = \{a^{2n} b^{2m} \mid n \geq 0, m \geq 0\}$

$\boxed{(aa)^*(bb)^*}$

Q15) R.E for language $L = \{w : |w| \bmod 3 = 0\}$ assuming $\Sigma = \{a, b\}$

→ $((a+b)(a+b)(a+b))^*$

OR Question can be obtain R.E for strings of a's and b's whose lengths are multiples of 3.

Q16) R.E for the language $L = \{w : n_a(w) \bmod 3 = 0$ where $w \in (a,b)^*\}$

— We are interested only in no. of a's. To be divisible by 3 the numbers of a's can be 0, 3, 6, 9 ...

Strings accepted are ε, aaa, aaab, baaa, baaab abaa, aabab. . . .

No. of a's should be multiple of 3.

$b^*ab^*ab^*ab^*$

∴ R.E will become $(b^*ab^*ab^*ab^*)^*$

Q17) Obtain R.E for $L = \{vuv : u, v \in \{a,b\}^*$ and $|v| = 2\}$

Sol?: Since $|v| = 2$, we have strings of length 2 for v.

i.e $v = aa + ab + ba + bb$

u can have any combinations of a's or b's. i.e $(a+b)^*$

i.e R.E is

$$aa(a+b)^*aa + ab(a+b)^*ab + ba(a+b)^*ba + bb(a+b)^*bb.$$

Q18) Obtain R.E for strings of a's & b's containing not more than three a's. [means atmost 3 a's]

Soln: That means we can have zero a's, 1a, 2a's or 3a's.

For no. of a's not more than 3 a's is given by

→ $(\epsilon+a)(\epsilon+a)(\epsilon+a)$

For this we can place any no. of b's represented as b^* as

$$\boxed{b^*(\epsilon+a)b^*(\epsilon+a)b^*(\epsilon+a)b^*}$$

Q19) Obtain R.E for the language $L=\{a^n b^m \mid m+n \text{ is even}\}$

Soln: For m+n to be even we have

1) m should be even and n should be even. That means we must have even no. of a's followed by even no. of b's.

R.E is $(aa)^*(bb)^*$

2) m should be odd and n should be odd. That means we must be odd no. of a's followed by odd no. of b's.

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

∴ The R.E is $\boxed{(aa)^*(bb)^* + a(aa)^* b(bb)^*}$

Q20) Obtain regular expression for the language

$L=\{a^n b^m : n \geq 4, m \leq 3\}$

- That is to accept strings of a's and b's with atleast 4 a's followed by atmost 3 b's.

R.E for Atleast 4 a's can be given as

$$aaaa(a)^*$$

R.E for Atmost 3 b's can be given as

$$(\epsilon+b)(\epsilon+b)(\epsilon+b)$$

$\therefore$ R.E is $aaaa(a)^*(\epsilon+b)(\epsilon+b)(\epsilon+b)$ or $aaaa^+(\epsilon+b)(\epsilon+b)$

Q21) Obtain R.E for $L = \{a^n b^m : n \leq 4, m \geq 2\}$

Almost 4 a's followed by atleast 2 b's.

$$(\epsilon+a)(\epsilon+a)(\epsilon+a)(\epsilon+a)\,bbb^*$$

Q22) a) Obtain R.E for $L = \{w : w$ has atleast one pair of consecutive zeros $/ w \in \{0,1\}^*\}$

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

b) Exactly one pair of consecutive zero's.

1^*001^*

Extra Questions:

1) Set of all strings such that number of 0's is odd
2) Set of all strings that do not contain 101
3) Set of all strings of 0's and 1's not containing 101 as substring
4) Set of all strings of 0's and 1's whose number of 0's is divisible by five and whose number of 1's is even.
5) Set of all strings of 0's and 1's containing even number of zero's.
6) Set of all strings with no pair of consecutive zeros.

Precedence of ~~any~~ regular Expression operators:

1) The star closure operator (denoted by *) has the highest precedence
2) The concatenation operator (also called dot operator) has the next highest precedence
3) The union operator (+ operator) has the least precedence.

→ The precedence of all the above operators can be changed using paranthesis.

Finite Automata & Regular Expressions:

Language defined by FA(DFA, NFA, ∈-NFA) is same as the language defined by Regular Expression.

To show that Regular expression defines same language we must show that:

1) Every language defined by one of these automata (DFA, NFA, ∈-NFA) is also defined by a regular expression. For this proof, we can assume the language is accepted by some DFA.

i.e From FA to obtain R.E we use DFA. to obtain R.E

2) Every language defined by regular expression is defined by one of these automata. For that part of the proof, the easiest is to show that there is an NFA with ∈-transitions accepting the same language.

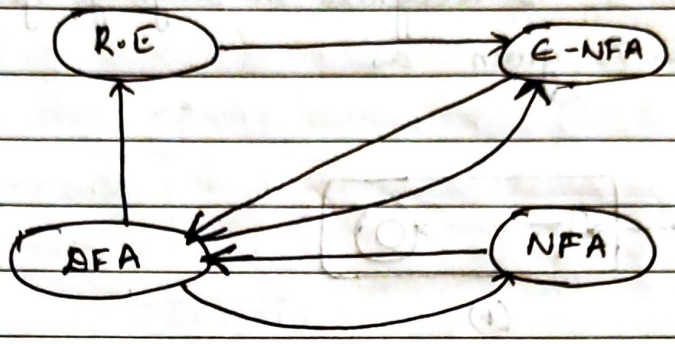
i.e From the R.E to obtain FA, the easiest way to obtain ∈-NFA from R.E

Date / /20

Relation b/w DFA, NFA, ϵ-NFA and Regular expression

OR

Equivalences b/w four different notations for regular language can be given as



Converting Regular Expression to Finite Automata

Theorem: Every language defined by Regular expression is defined by a finite Automata.

(OR)

Proof: Prove that there exists a finite automaton to accept the language $\mathcal{L}(R)$ corresponding to the regular expression R.

⇒ Let R be regular expression. The there exists FA $E=(Q, \Sigma, \delta, q_0, F)$ which accept $\mathcal{L}(R)$

Proof: Suppose $\mathcal{L}=\mathcal{L}(R)$ for a Regular expression. We show that $\mathcal{L}=\mathcal{L}(E)$ for some ϵ-NFA E with

a) Exactly one accepting state

b) No arcs into the initial state

c) No arcs out of the accepting state. $\mathcal{L}(R)$ $\mathcal{L}(R)$

Schematic Representation of FA accepting $\mathcal{L}(R)$ is → ◎ or → q_f

Basis: There are three parts to the basis

By definition, ϕ, ϵ and a are regular expression

Automata for Construction of ϕ is given below.

1) ϕ is the language of the automaton, since there are no paths from start state to accepting state

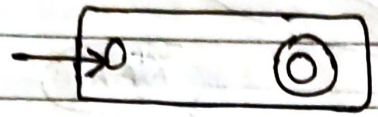
Automaton for regular expression ϵ is shown below.

2) {ϵ} is the language of the automaton, since there only path from the start state to final state is

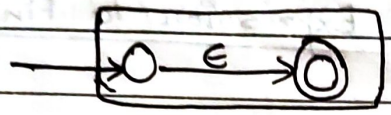
* write Diagram continuously for each step

3) {a} is the language of the automaton consists of the string a. Automaton for a regular expression 'a' is given in fig(3) of basis

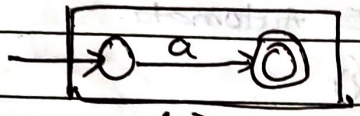
Automata to recognize the language for the regular expression is given as



(1)



(2)



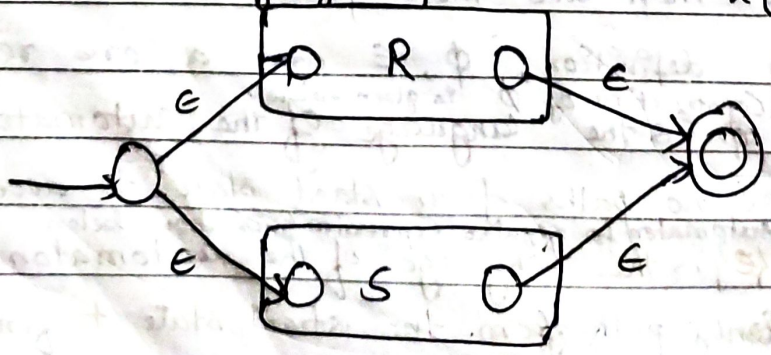
(3)

Fig(a). Shows the basis of the construction of an automaton from a regular Expression.

Induction: There are 3 parts in induction.

Let R be a regular expression denoting language $\mathcal{L}(R)$ and S be a regular expression denoting language $\mathcal{L}(S)$ The expression is R+S.

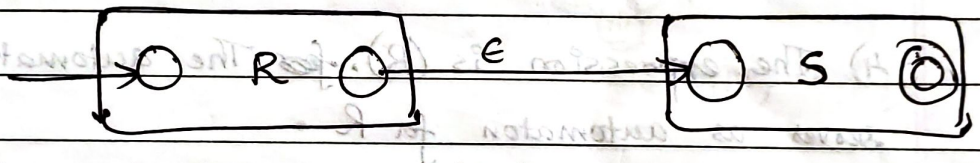
1) The ϵ-NFA automata for accepting either $\mathcal{L}(R)$ or $\mathcal{L}(S)$ which can be represented as $\mathcal{L}(R+S)$ is shown below. The language of the automaton is $\mathcal{L}(R) \cup \mathcal{L}(S)$



Date / /20

Starting at new start state of the new automaton we go to the start state of either the automaton for R or the automaton for S. We then reach the accepting (final) states of one of these automata, following a path labeled by some string in $\mathcal{L}(R)$ or $\mathcal{L}(S)$. Once we reach the accepting state of the automaton for R or S we can follow one of the ϵ-arcs to the accepting state of new automaton.

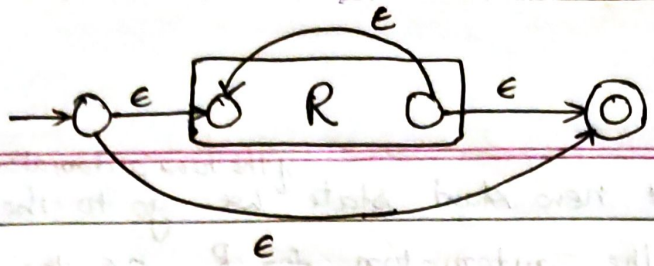
2) The expression is R.S or RS. The automaton for the concatenation is shown below. which accepts $\mathcal{L}(R)$ followed by $\mathcal{L}(S)$. The language for the automaton is $\mathcal{L}(R)\mathcal{L}(S)$.



The start state of the first automaton, i.e automaton for R becomes the start state of whole and the accepting (final) state of second automaton i.e automaton for S becomes the final state of the whole automata

→ The only paths from start to final state go first through the automata for R where it must follow a path labeled by a string in $\mathcal{L}(R)$ and then through the automato for S where it follows a path labeled by a string in $\mathcal{L}(S)$.

3) The expression is R^*. The automaton for closure of R is shown below. The automaton allows us to go either.



The automaton allows us to go either

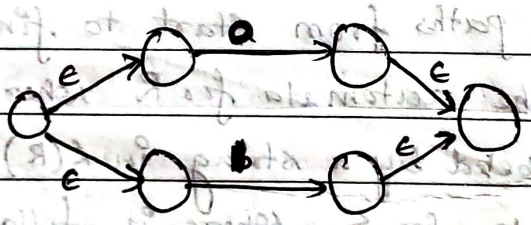
a) Directly from the start state to the accepting state along a path labeled ϵ. That path lets us accept ϵ, which is in $\mathcal{L}(R^*)$.

b) To the start state of the automaton for R, through that automaton one or more times and then to the accepting state. This set of path allows us to accept string $\mathcal{L}(R)$, $\mathcal{L}(R)\mathcal{L}(R)$, $\mathcal{L}(R)\mathcal{L}(R)\mathcal{L}(R)$ etc... thus covering all strings in $\mathcal{L}(R^*)$ except ϵ, which was covered by the direct arc from to the final state.

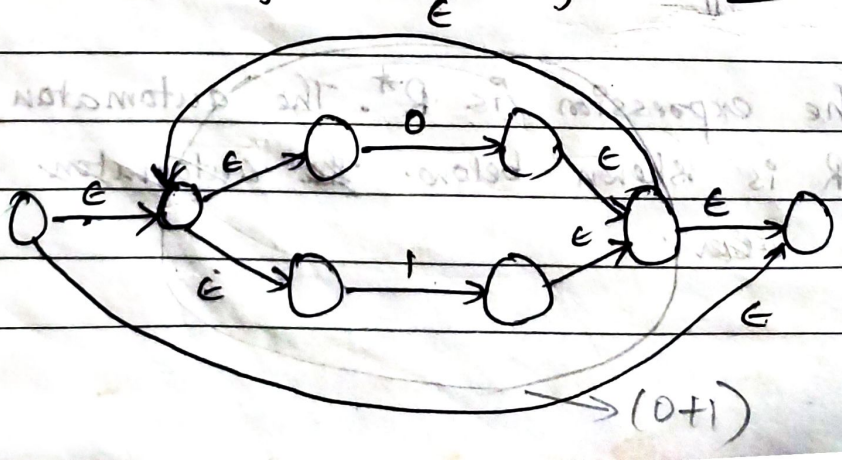
4) The expression is (R). The automaton for (R) serves as automaton for R.

6) Convert the regular expression $(0+1)^*.1.(0+1)$ to an ϵ-NFA

1) First construct automaton for $(0+1)$ to accept R.S

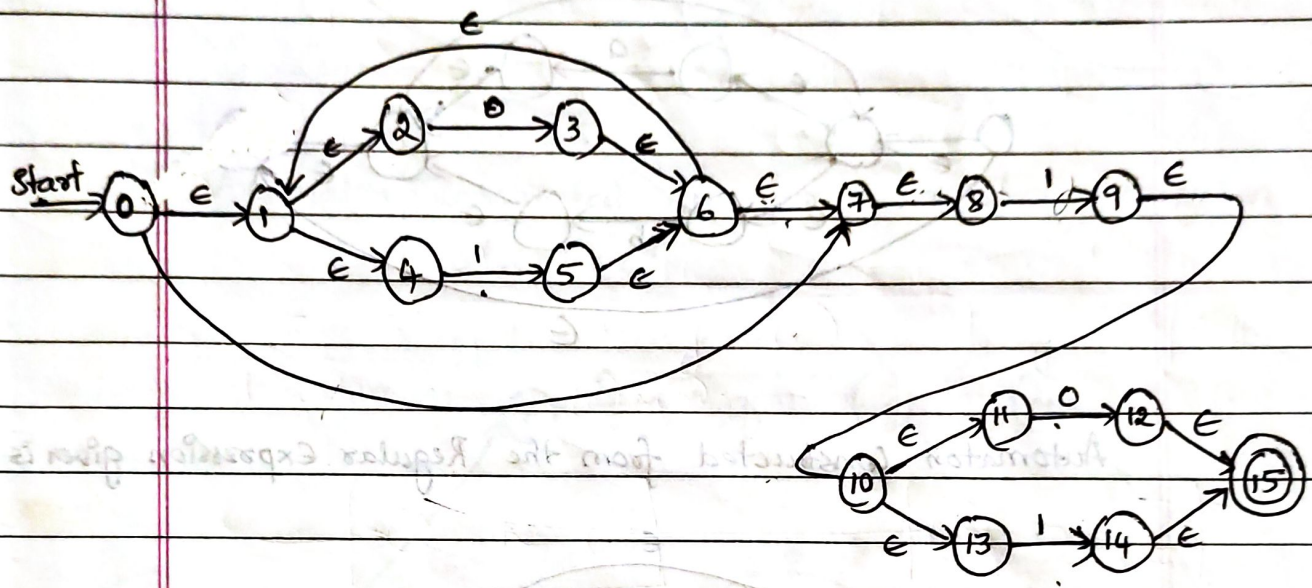


2) Automaton for closure of $(0+1)$ i.e $(0+1)^*$ is



3) Automaton to accept 1 is

Automaton to ~~accept~~ Constructed from the given Regular Expression is



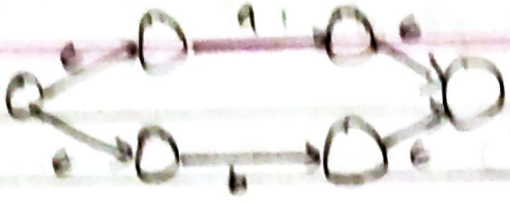
Q2) Obtain R.E to accepts strings of a's and b's starting with the string ab. Obtain ϵ-NFA from the given R.E.

→ R.E to accept strings of a's and b's starting with a

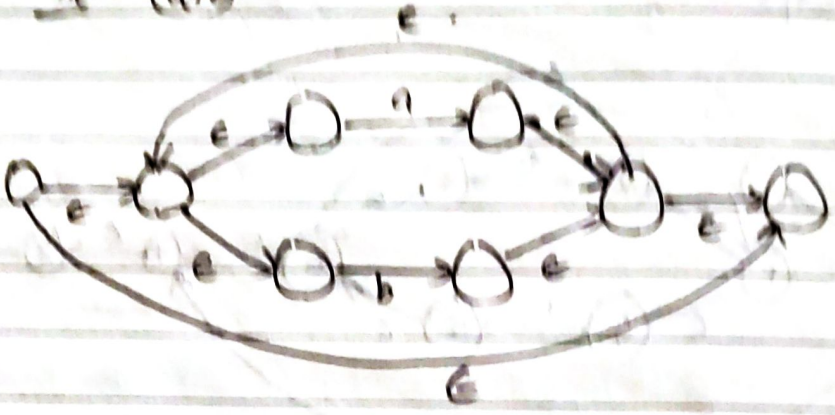
$$ab(a+b)^*$$

1) First ~~construct~~ compute Automata to accept ~~ab~~ ab

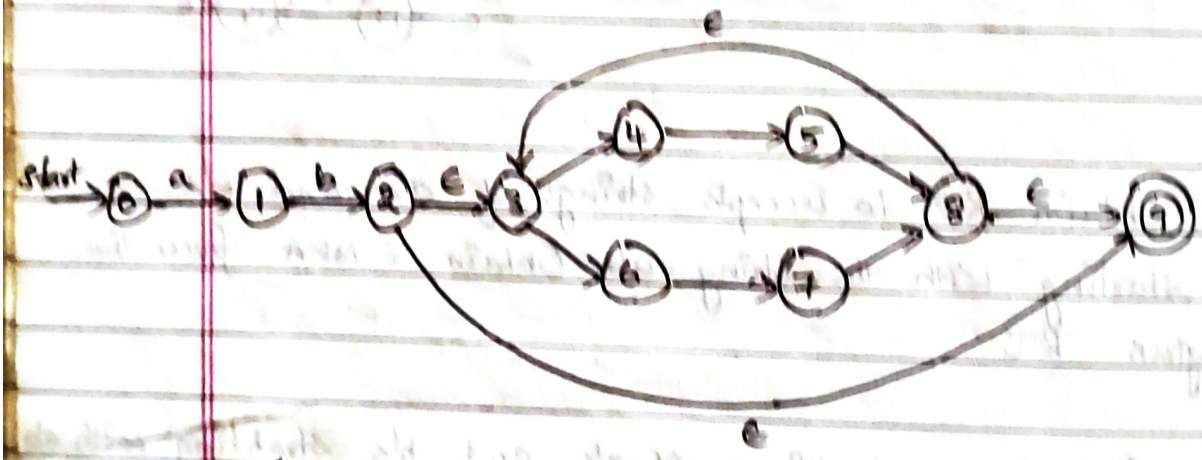
2) Construct Automaton to accept (a+b)



a) Construct Automaton to accept Closure of (a+b)
i.e $(a+b)^*$



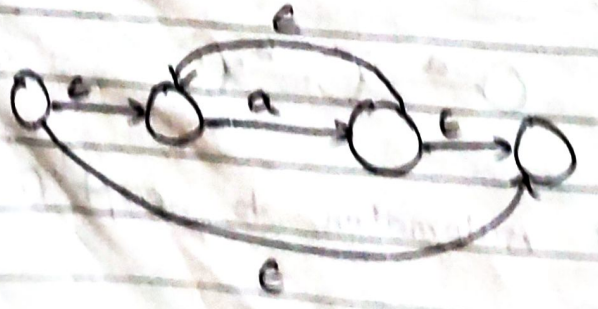
Automaton constructed from the Regular Expression given:



(Better use ε between every concatenation)

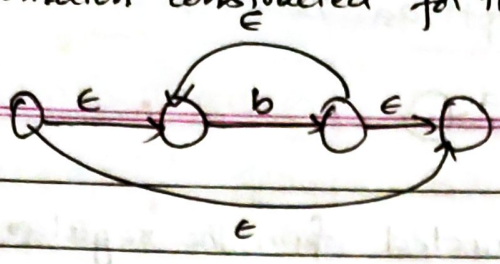
Q3) Obtain RE-NFA for the regular expressions-
$a^* + b^* + c^*$

1) Automaton constructed to accept a^* is

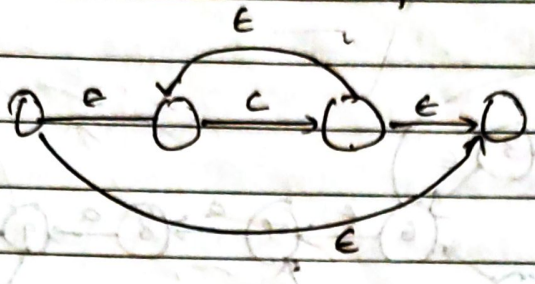


2) Automaton constructed for the regular expression b^* is

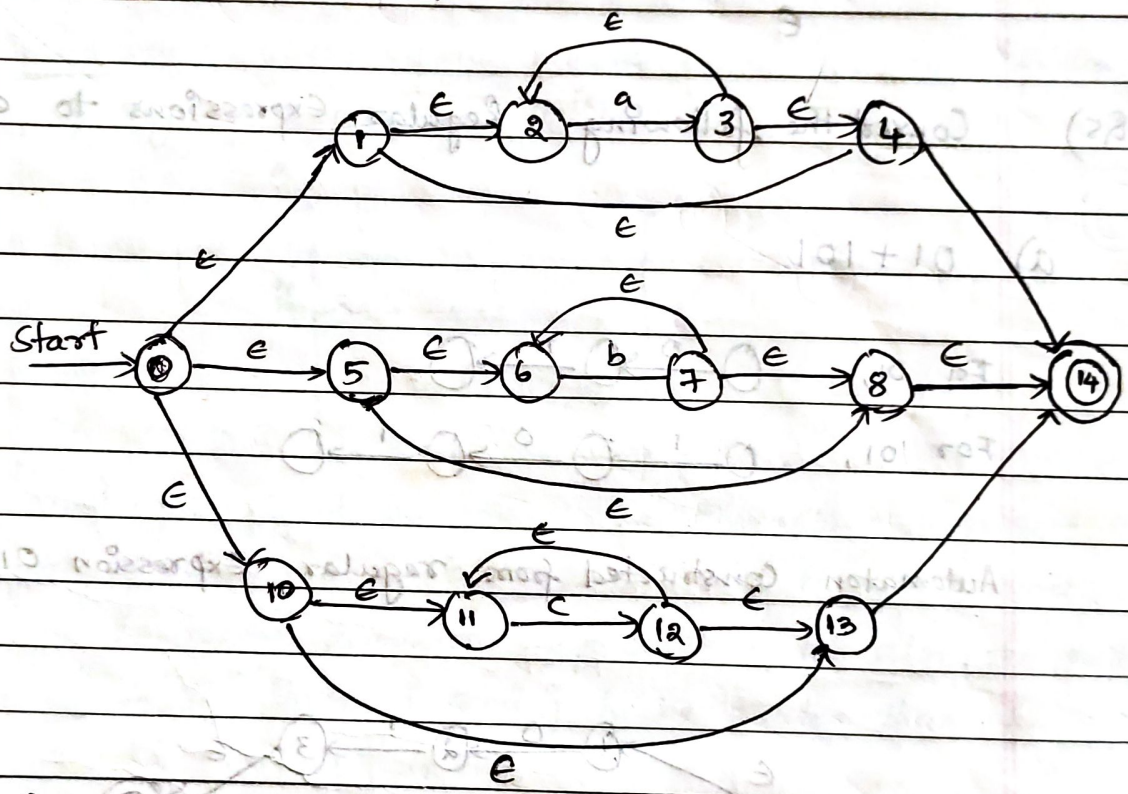
Date / /20



3) Automaton constructed for the regular expression c^* is

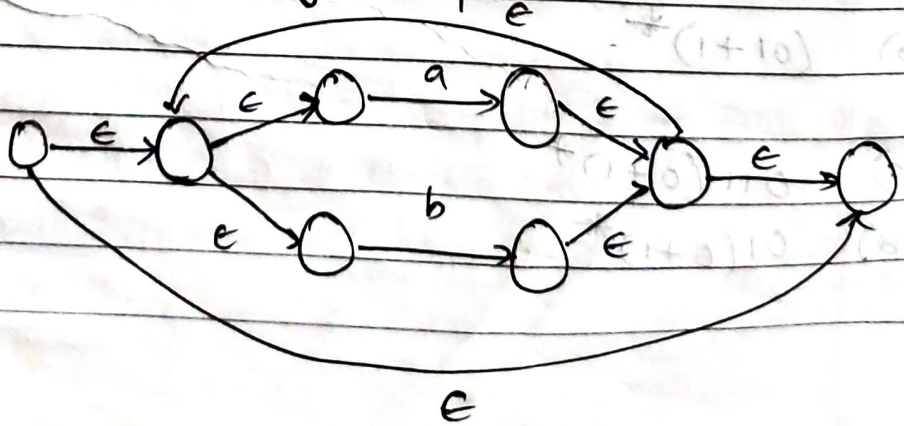


Automaton constructed for the regular expression $a^* + b^* + c^*$



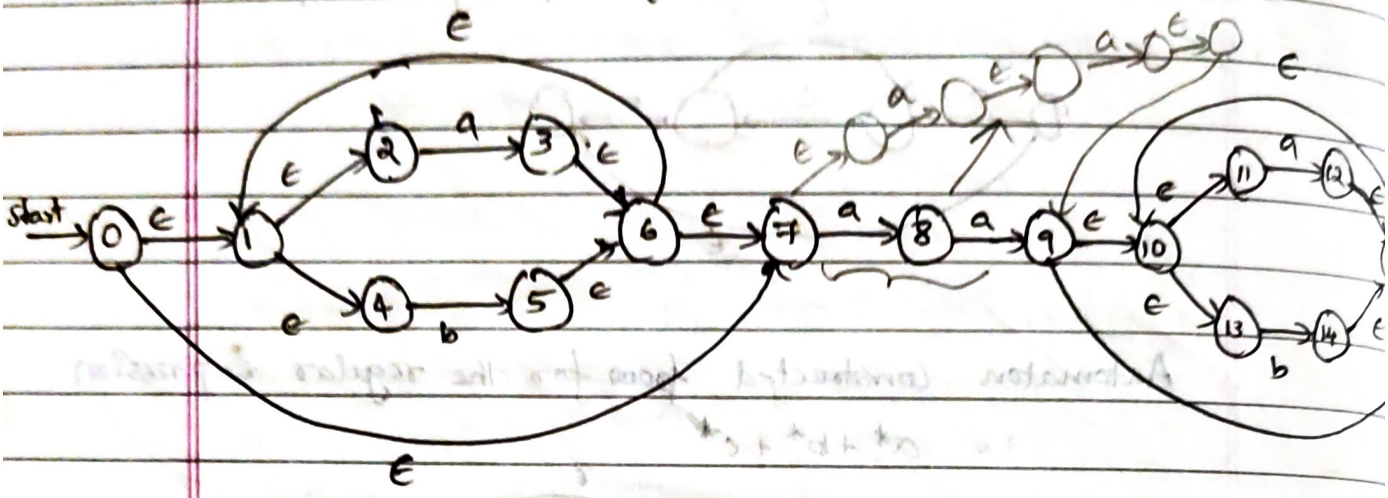
Q4) Convert Regular expression $(a+b)^* aa(a+b)^*$ to ϵ-NFA

1) Automaton for the regular expression $(a+b)^*$ is.



Automaton to accept aa is

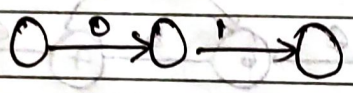
∴ Automaton constructed from the regular Expression $(a+b)^* aa(a+b)^*$ is



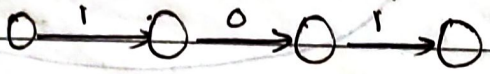
Q5) Convert the following Regular Expressions to ϵ-NFA's

a) 01+101

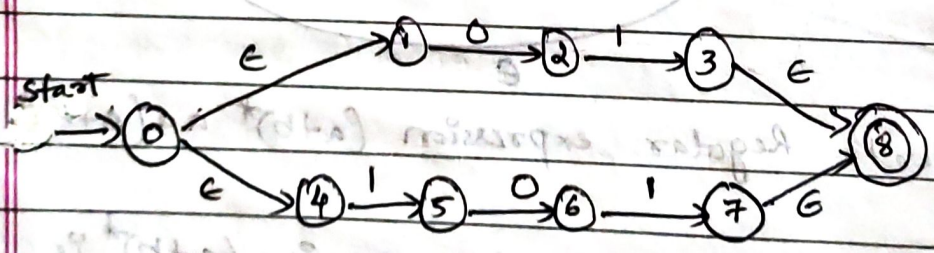
For 01,



For 101,



Automaton constructed from regular Expression 01+101 is



b) $(01+1)^*$

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

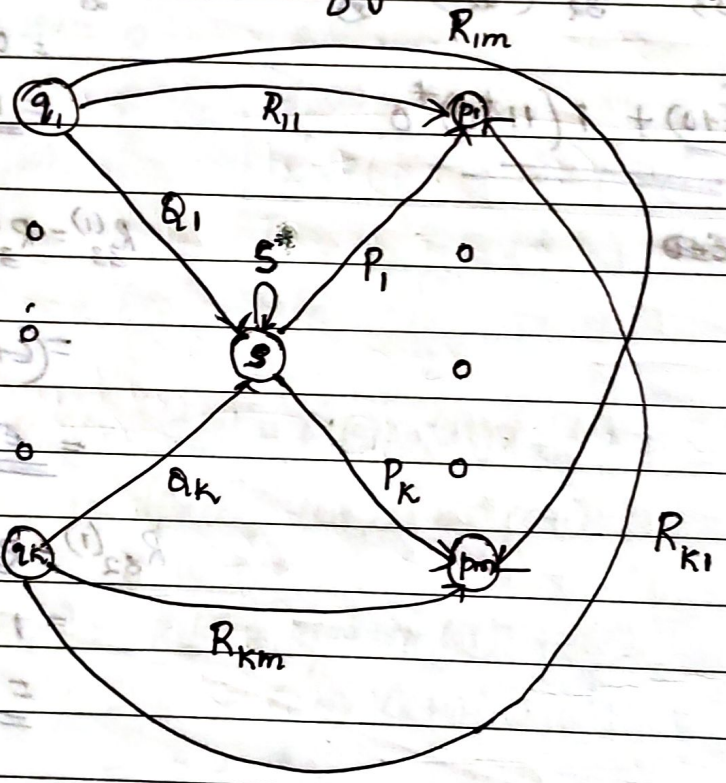
d) $01(0+1)^*$

2) State Elimination Method.

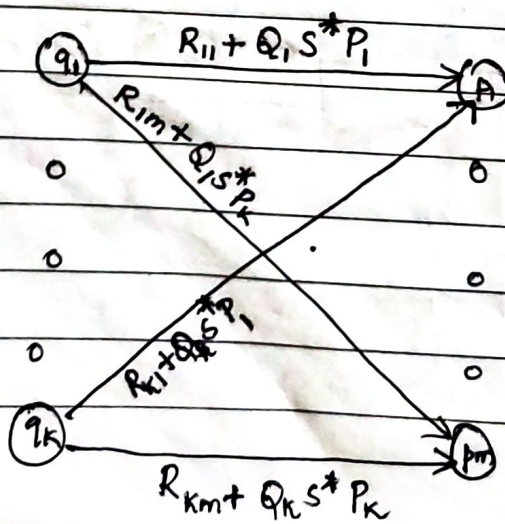
Obtain RE from FA

When we eliminate a state s, all the paths that went through s will no longer exist in the automaton. We must include the labels of paths that went from q to p through state s on the arc directly that goes from q to p state - The labels of the path will have strings instead of single symbol. We represent these strings using regular expression.

Consider the fig below; A State s to be eliminated



After eliminating the state s, the resulting fig is

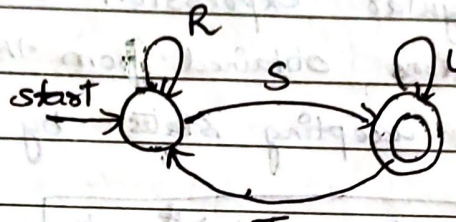


Date / /20

The steps for constructing Regular Expression is as follows:

1) For each accepting state (final state) q, apply the above reduction process to produce equivalent automaton with regular expression labels on the arcs. Eliminate states except q and the start state q_0.

2) If $q \neq q_0$ (start state and final states are not same i.e start state is not the final state)
- Then we will have two-state automaton that looks like



The regular expression can be described in various ways. One is

$$(R + SU^*T)^* SU^*$$

(start to start itself) (& to final state)

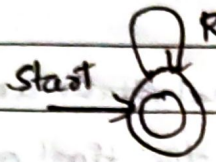
we can go from start state to itself any no. of times, as R^* or SU^*T by following sequence of paths whose labels are either in $\mathcal{L}(R)$ or $\mathcal{L}(SU^*T)$

~~$\mathcal{L}(SU^*T)$~~ SU^*T represents paths that go to the accepting state via path in $\mathcal{L}(S)$ then be in the final state several times using $\mathcal{L}(U)$ and then go to the start state via the paths from $\mathcal{L}(T)$. Then we must go to the accepting state never return to start state by following paths from $\mathcal{L}(S)$. In the accepting state return in same state (final) via paths from $\mathcal{L}(U)$

3) you can even write as

$$R^*S(U^* + TR^*S)^*$$

3) If the start state is itself the accepting state, then we must perform the reduction process to eliminate every state other than the start state. Then we will have only one-state automaton that looks like

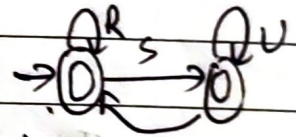


The regular expression denoting the strings it accepts is R^*

4) The desired regular expression is the union (sum) of all the expressions obtained from the reduced automata for each accepting states by rules (2) and (3)

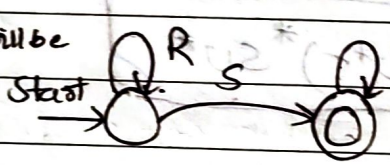
i.e

$$R^* + (R + SU^*T)^*SU^*$$



Note: if in the FA, if T is not there i.e

(2) will be



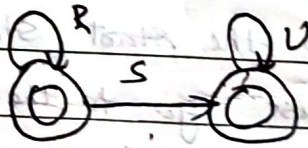
, then Regular expression is

R^*SU^*

(3) will be R^*

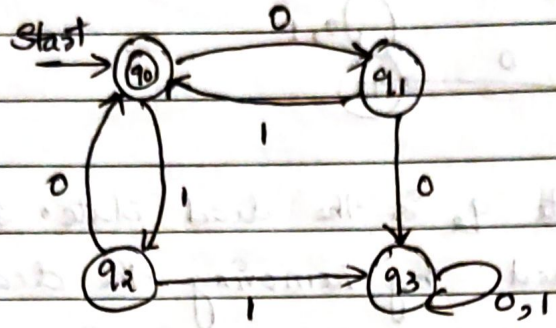
→ If both states are final states then (4) will be.

$$R^* + R^*SU^*$$

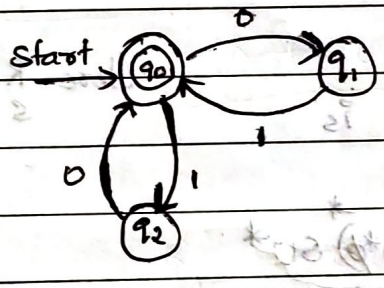


Date / /20

Q1) Obtain Regular expression for the FA shown below.

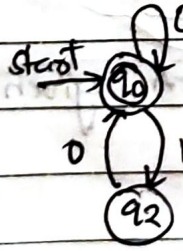


From the finite Automata it is clear that q_3 is the dead state. i.e (Once state q_3 is reached irrespective of the input, the machine stays in q_3 only and there is no way to reach the final state) and so all the edges connected to q_3 can be removed and the resulting figure is given below.



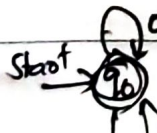
Next we have to elimate the state q_1,

→ When 0 is read from q_0 state, it will go to q_1 and from q_1 if you read 1 you will come back to the state q_0. and the process is repeated. Hence the automaton written with labels in R.E is.



Generalized transition graph is

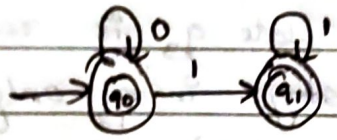
→ Eliminate state q_2,



∴ Regular Expression is $(01+10)^*$

Q2) What is the language accepted by the following FA.

→ In Above FA, state q_2 is the dead state. Hence the automata obtained by removing the dead state q_2 and all the edges connected to it is



Here the start state q_0 is the final state as well as q_1 is the final state. So further elimination is not possible.

Regular expression is

Here R=0 T=ϕ
S=1 U=1

$$R^* + (R + SU^*\phi)^* SU^*$$

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

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

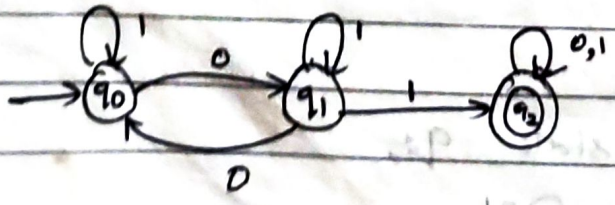
$$\Rightarrow 0^* (\epsilon + 1^+)$$

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

$$= 0^* 1^*$$

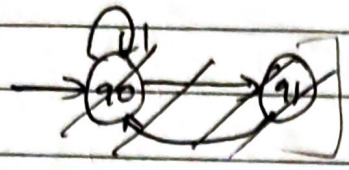
→ In short = $0^* 1^*$

Q3) Obtain regular Expression for the FA given below.

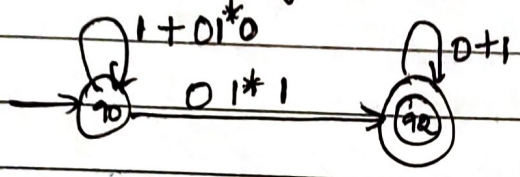


Date / /20

Soln: Since q_2 is the dead state, we need to eliminate the dead state by removing all the edges connected to the state q_2. Hence the FA will be



In the FA given, q_0 is the start state and q_2 is the final state. We need to eliminate the state q_1 and perform the reduction process as:



All the labels of paths are written as Regular Expression.

To obtain the RE from above diagram.

Here $R = 1+01^*0$ $\quad U = (0+1)^*$

$S = 01^*1$

$T = \phi$

$R^* S U^*$

$\therefore$ Regular Expression is $(R + SU^*T)^* SU^*$

$\Rightarrow \left((1+01^*0) + 01^*1(0+1)^*\phi\right)^* 01^*1(0+1)^*$

$\Rightarrow \underline{(1+01^*0)^* 01^*1(0+1)^*}$