Internal Assessment Question Paper – 1

M.S. Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of CSE

Programme: B.E **Term: march – June 2025**
Course: Finite Automata & Formal languages **Course Code: CS45** **Credit :2:1:0**
Sem:IV **CIE: I** **Section: A, B, C & D**
Date:26-04-2024 **Time:10.40 am to 11.40 am**
Max Marks: 30 **Duration: 1Hr** **Portions for Test: L1-L14.**

Scheme

<table>
<tr><th>Sl#</th><th></th><th>Question</th><th>Marks</th></tr>
<tr><td>1</td><td>a)</td><td>

i. Define DFA.

A finite automaton can be defined as a tuple:

{ Q, Σ, δ, q_0, F}, where:

Q: Finite set of states

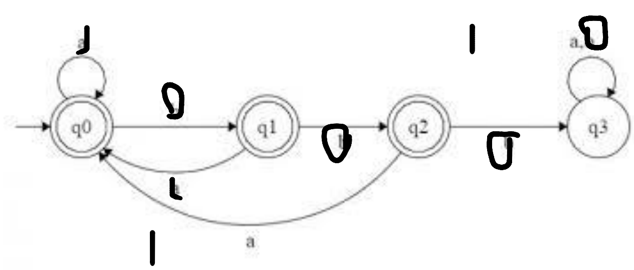
Σ: Set of input symbols

q_0: Initial state

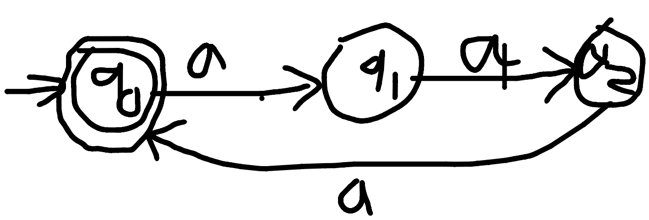
F: Set of final states

δ: Transition function

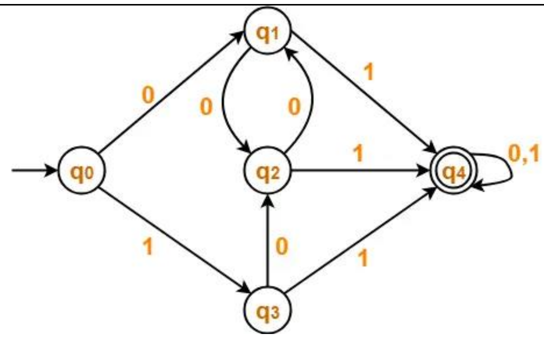
ii. Draw a DFA to accept string of 0’s and 1’s having no 3 consecutive 0’s



Obtain a DFA to accept L={n_a(w)mod 3=0} on $\sum$ ={a}



</td><td>01</td></tr>
<tr><td></td><td>b)</td><td>Minimize the following DFA using table filling algorithm.</td><td>5</td></tr>
</table>



	0	1
$q_0\ q_1$	$q_1\ q_2$	$q_3\ q_4$
$q_0\ q_2$	$q_1\ q_1$	$q_3\ q_4$
$q_0\ q_3$	$q_1\ q_2$	$q_3\ q_4$
$q_0\ q_4$	$q_1\ q_4$	$q_3\ q_4$
$q_1\ q_2$	$q_1\ q_2$	$q_4\ q_4$
$q_1\ q_3$	$q_2\ q_2$	$q_4\ q_4$
$q_1\ q_4$	$q_2\ q_4$	$q_4\ q_4$
$q_2\ q_3$	$q_1\ q_2$	$q_4\ q_4$
$q_2\ q_4$	$q_1\ q_4$	$q_4\ q_4$
$q_3\ q_4$	$q_2\ q_4$	$q_4\ q_4$

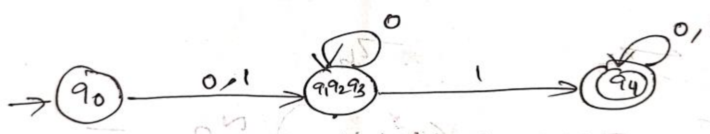
	q_0	q_1	q_2	q_3
q_1	*			
q_2	*			
q_3	*			
q_4	*	*	*	*

	0	1
$q_1 q_2$	$q_1\ q_2$	$q_4\ q_4$
$q_1 q_3$	$q_2\ q_2$	$q_4\ q_4$
$q_2\ q_3$	$q_2\ q_2$	$q_4\ q_4$

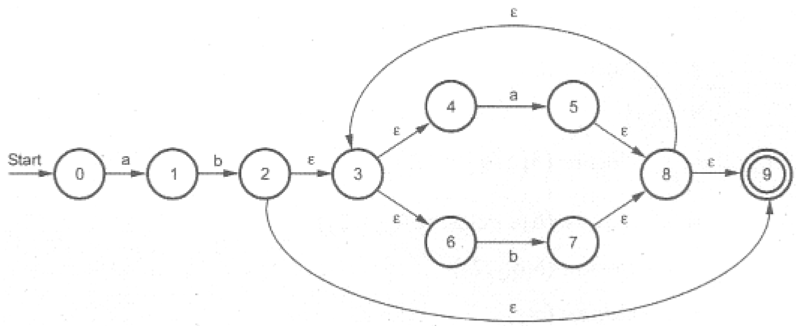
combining

	0	1
$q_1 q_2 q_3$	$q_1 q_2$ ~~q_3~~	$q_4\ q_4$
q_0	q_1	q_3
q_4	q_4	q_4

	0	1
$q_1 q_2 q_3$	$q_1 q_2 q_3$	q_4
$\rightarrow q_0$	$q_1 q_2 q_3$	$q_1 q_2 q_3$
$\rightarrow q_4$	q_4	q_4



c) Convert the following **ε**-NFA to DFA



Solution: We will first obtain ε - closure of every state. The ε - closure is basically an ε - transition from one state to other. Hence

ε - closure (0) = {0}
ε - closure (1) = {1}
ε - closure (2) = {2, 3, 4, 6, 9}
ε - closure (3) = {3, 4, 6}
ε - closure (4) = {4}
ε - closure (5) = {5, 8, 3, 4, 6, 9}
= {3, 4, 5, 6, 8, 9} sorted it!
ε - closure (6) = {6}
ε - closure (7) = {7, 8, 3, 4, 6, 9}
= {3, 4, 6, 7, 8, 9}
ε - closure (8) = {8, 3, 4, 6, 9} **5**
= {3, 4, 6, 8, 9}
ε - closure (9) = {9}

Now we will obtain δ' transitions for each state and for each input symbol.

0---A
δ' (0, a) = ε - closure (δ (δ` (0, ε), a))
= ε - closure (δ (ε - closure(0), a))
= ε - closure (δ (0, a))
= ε - closure (1)
{1}----------------------------------- ---------------B
δ ' (0, b) = ε - closure (δ (δ` (0,ε), b))
= ε - closure (δ (ε - closure (0), b))
= ε - closure (δ (0, b))
= ε - closure (φ)
= φ

δ' (1,a) = ε - closure (δ (1, a))
= ε - closure (φ)
= φ
δ ' (1, b) = ε - closure (δ (1, b))
= ε - closure (2)
= {2, 3, 4, 6, 9} ---------------------------------------C
δ ' ({2, 3, 4, 6, 9} , a)
= ε - closure (δ (2, a) U δ (3, a) U δ (4, a) U δ (6, a) U δ (9, a))

= ε - closure (φ U φ U 5 U φ U φ)
= ε - closure (5)
= {3, 4, 5, 6, 8, 9}-------------------------------------D
δ' ({2, 3, 4, 6, 9}, b) =
= ε - closure (δ(2, b) U δ(3, b) U δ(4, b) U δ(6, b) U δ(9, b))
= ε - closure (φ U φ U φ U 7 U φ)
= ε - closure (7)
= {3, 4, 6, 7, 8, 9}-----------------------------------E

δ ' ({3, 4, 5, 6, 8, 9}, a)
= e-closure (δ (3, a) U (4, a) U δ (5, a) U δ (6, a) U δ (8, a) U δ (9,a))
= ε - closure (φ U 5 U φ U φ U φ U φ U φ)
= ε - closure (5)
= {3, 4, 5, 6, 8, 9} -------------------------------------D
δ' ({3, 4, 5, 6, 8, 9}, b)
= ε - closure (δ (3, b) U δ (4, b) U δ (5, b) U δ (6, b) U δ (8, b) U δ (9, b))
= ε - closure (φ U φ U φ U 7 U φ U φ)
= ε - closure (7)
= {3, 4, 6, 7, 8, 9}-------------------------------------E

δ' ({3, 4, 6, 7, 8, 9}, a)
= ε - closure (δ (3, a) U δ (4, a) U δ (6, a) U δ (7, a) U δ (8, a) U δ (9, a))
= ε - closure (φ U 5 U φ U φ U φ U φ)
= ε - closure (5)
= {3, 4, 5, 6, 8, 9}---------------------------------------D
δ ' ({3, 4, 6, 7, 8, 9}, b)
= ε - closure (δ (3, b) U δ (4, b) U δ (6, b) U δ (7, b) U δ (8, b) U δ (9, b))
= ε - closure (φ U φ U 7 U φ U φ U φ)
= ε - closure (7)
= {3, 4, 6, 7, 8, 9}---------------------------------------E
Now we will build the transition table using above calculated δ' transitions.

	a	b
A	ϕ	A
B	ϕ	C
C	D	E
D	D	E
E	D	E

2 a) If D=$(Q_D, \sum, \phi_D, \{q_0\}, F_D)$ is the DFA constructed from NFA N=$(Q_N, \sum, \phi_N, \{q_0\}, F_N)$ by the subset construction. Then show that $L_D = L_N$. **[5]**

Theorem 2.11: If $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ is the DFA constructed from NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ by the subset construction, then $L(D) = L(N)$.

PROOF: What we actually prove first, by induction on $|w|$, is that

$$\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$$

Notice that each of the $\hat{\delta}$ functions returns a set of states from Q_N, but $\hat{\delta}_D$ interprets this set as one of the states of Q_D (which is the power set of Q_N), while $\hat{\delta}_N$ interprets this set as a subset of Q_N.

BASIS: Let $|w| = 0$; that is, $w = \epsilon$. By the basis definitions of $\hat{\delta}$ for DFA's and NFA's, both $\hat{\delta}_D(\{q_0\}, \epsilon)$ and $\hat{\delta}_N(q_0, \epsilon)$ are $\{q_0\}$.

INDUCTION: Let w be of length $n + 1$, and assume the statement for length n. Break w up as $w = xa$, where a is the final symbol of w. By the inductive hypothesis, $\hat{\delta}_D(\{q_0\}, x) = \hat{\delta}_N(q_0, x)$. Let both these sets of N's states be $\{p_1, p_2, \ldots, p_k\}$.

The inductive part of the definition of $\hat{\delta}$ for NFA's tells us

$$\hat{\delta}_N(q_0, w) = \bigcup_{i=1}^{k} \delta_N(p_i, a) \tag{2.2}$$

The subset construction, on the other hand, tells us that

$$\delta_D(\{p_1, p_2, \ldots, p_k\}, a) = \bigcup_{i=1}^{k} \delta_N(p_i, a) \tag{2.3}$$

Now, let us use (2.3) and the fact that $\hat{\delta}_D(\{q_0\}, x) = \{p_1, p_2, \ldots, p_k\}$ in the inductive part of the definition of $\hat{\delta}$ for DFA's:

b) Define Pumping lemma. Show that the given language L={ww^R | w∊(a+b)*}is not regular. **[5]**

The Pumping Lemma is used for proving that a language is **not** regular. Here is the Pumping Lemma.

> If L is a regular language, then there is an integer $n > 0$ with the property that:
>
> **(*)** for any string $x \in L$ where $|x| \geq n$, there are strings u, v, w such that
>
> (i) $x = uvw$,
> (ii) $v \neq \epsilon$,
> (iii) $|uv| \leq n$,
> (iv) $uv^kw \in L$ for all $k \in \mathbb{N}$.

f I let string w be a^mb^m then we know that y will consists of only a's because of the rule |xy| <= m.

And if I set i=0, then ww^R will have fewer a's on the left side than on the right side. Thus, it proves that this language is not regular.

However, my text book (*An Introduction to Formal Languages and Automata* pg. 118 by Linz) says if I were to choose w = a^2m and let y = aa, then I would fail.

But how so?

To my mind, no matter what x, y, z are, the first a^2m will have fewer a's or more depending on what i is than the second a^2m.

Ex 2:L={ww^R|w ∈{0,1}*}

W=001 100 ,x=001 ,y=1 ,z=00

Now check $xy^kz \in L, \forall k \geq 0$

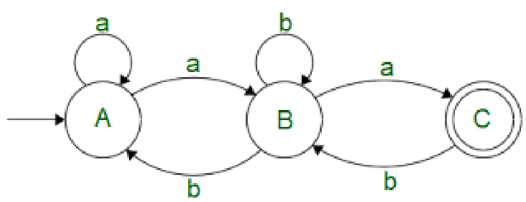
Let k=0 , xy^0z=001 ε 00=00100 ∉ L

Let k=1, xy^1z=001 1 00=001100 ∈ L

Let k=2, xy^2z= 001 11 00 = 0011100 ∉ L

So L is NRL.

c) Obtain regular expression using Kleen's Theorem



K=0

$R_{11}^{(0)}$ =a+**ε**

$R_{12}^{(0)}$ =a

$R_{13}^{(0)} = \phi$

$R_{21}^{(0)}$ =b

$R_{22}^{(0)}$ =b+**ε**

$R_{23}^{(0)}$ =a

$R_{31}^{(0)} = \phi$

$R_{32}^{(0)}$ =b

$R_{33}^{(0)}$ = **ε**

K=1

$R_{11}^{(1)} = R_{11}^{(0)} + R_{11}^{(0)}(R_{11}^{(0)})^* R_{11}^{(0)} = a^*$

$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)}(R_{11}^{(0)})^* R_{12}^{(0)} = a + a^* = a^*$

$R_{13}^{(1)} = R_{13}^{(0)} + R_{11}^{(0)}(R_{11}^{(0)})^* R_{13}^{(0)} = \phi$

$R_{21}^{(1)} = R_{21}^{(0)} + R_{21}^{(0)}(R_{11}^{(0)})^* R_{11}^{(0)} = ba^*$

$R_{22}^{(1)} = R_{22}^{(0)} + R_{21}^{(0)}(R_{11}^{(0)})^* R_{12}^{(0)} = (b+\varepsilon) + b\,a^*a$

$R_{23}^{(1)} = R_{23}^{(0)} + R_{21}^{(0)}(R_{11}^{(0)})^* R_{13}^{(0)} = a$

$R_{31}^{(1)} = R_{31}^{(0)} + R_{31}^{(0)}(R_{11}^{(0)})^* R_{11}^{(0)} = \phi$

$R_{32}^{(1)} = R_{32}^{(0)} + R_{31}^{(0)}(R_{11}^{(0)})^* R_{13}^{(0)} = b$

$R_{33}^{(1)} = R_{33}^{(0)} + R_{31}^{(0)}(R_{11}^{(0)})^* R_{13}^{(0)} = \varepsilon$

5

K=2

$R_{11}^{(2)} = R_{11}^{(1)} + R_{12}^{(1)}(R_{22}^{(1)})^* R_{21}^{(1)} = a^*$

$R_{12}^{(2)} = R_{12}^{(1)} + R_{12}^{(1)}(R_{22}^{(1)})^* R_{22}^{(1)} = a^*((b+\varepsilon) + b\,a^*a)^*$

$R_{13}^{(2)} = R_{13}^{(1)} + R_{12}^{(1)}(R_{22}^{(1)})^* R_{23}^{(1)} = a^*((b+\varepsilon) + b\,a^*a)^*a$

$R_{21}^{(2)} = R_{21}^{(1)} + R_{22}^{(1)}(R_{22}^{(1)})^* R_{21}^{(1)} = ((b+\varepsilon) + b\,a^*a)^*b\,a^*$

$R_{22}^{(2)} = R_{22}^{(1)} + R_{22}^{(1)}(R_{22}^{(1)})^* R_{22}^{(1)} = ((b+\varepsilon) + b\,a^*a)^*$

$R_{23}^{(2)} = R_{23}^{(1)} + R_{22}^{(1)}(R_{22}^{(1)})^* R_{23}^{(1)} = {}^*((b+\varepsilon) + b\,a^*a)^*a$

		$R_{31}^{(2)} = R_{31}^{(1)} + R_{32}^{(1)}(R_{22}^{(1)})^* R_{21}^{(1)} = b^*((b+\boldsymbol{\varepsilon}) + b a^* a)^* b a^*$ $R_{32}^{(2)} = R_{32}^{(1)} + R_{31}^{(1)}(R_{11}^{(1)})^* R_{13}^{(1)} = b$ $R_{33}^{(2)} = R_{33}^{(1)} + R_{31}^{(1)}(R_{11}^{(1)})^* R_{13}^{(1)} = \boldsymbol{\varepsilon}$ K=3 $R_{13}^{(3)} = R_{13}^{(2)} + R_{13}^{(2)}(R_{33}^{(2)})^* R_{33}^{(2))} = a^*((b+\boldsymbol{\varepsilon}) + b a^* a)^* a + \boldsymbol{\varepsilon}$	
3	**a)**	Define Regular expression. (i) Strings of a's and b's containing not more than three a's $b^*(a+\boldsymbol{\varepsilon})\ b^*(a+\boldsymbol{\varepsilon})\ b^*(a+\boldsymbol{\varepsilon})\ b^*$ (ii) Obtain a regular expression for $L = \{a^n b^m : n + m \text{ is even}\}$ $(aa)^*(bb)^* + (aa)^* a (bb)^* b$	**5**
	b)	Prove that L=L(A) for some DFA, then there is a regular expression R such that L=L(R). **Theorem 3.4:** If $L = L(A)$ for some DFA A, then there is a regular expression R such that $L = L(R)$. PROOF: Let us suppose that A's states are $\{1, 2, \ldots, n\}$ for some integer n. No matter what the states of A actually are, there will be n of them for some finite n, and by renaming the states, we can refer to the states in this manner, as if they were the first n positive integers. Our first, and most difficult, task is to construct a collection of regular expressions that describe progressively broader sets of paths in the transition diagram of A. Let us use $R_{ij}^{(k)}$ as the name of a regular expression whose language is the set of strings w such that w is the label of a path from state i to state j in A, and that path has no intermediate node whose number is greater than k. Note that the beginning and end points of the path are not "intermediate," so there is no constraint that i and/or j be less than or equal to k. Figure 3.2 suggests the requirement on the paths represented by $R_{ij}^{(k)}$. There, the vertical dimension represents the state, from 1 at the bottom to n at the top, and the horizontal dimension represents travel along the path. Notice that in this diagram we have shown both i and j to be greater than k, but either or both could be k or less. Also notice that the path passes through node k twice, but never goes through a state higher than k, except at the endpoints. To construct the expressions $R_{ij}^{(k)}$, we use the following inductive definition, starting at $k = 0$ and finally reaching $k = n$. Notice that when $k = n$, there is	**5**

no restriction at all on the paths represented, since there *are* no states greater than n.

BASIS: The basis is $k = 0$. Since all states are numbered 1 or above, the restriction on paths is that the path must have no intermediate states at all. There are only two kinds of paths that meet such a condition:

1. An arc from node (state) i to node j.

2. A path of length 0 that consists of only some node i.

If $i \neq j$, then only case (1) is possible. We must examine the DFA A and find those input symbols a such that there is a transition from state i to state j on symbol a.

a) If there is no such symbol a, then $R_{ij}^{(0)} = \emptyset$.

b) If there is exactly one such symbol a, then $R_{ij}^{(0)} = \mathbf{a}$.

c) If there are symbols $a_1, a_2, \ldots, a_k$ that label arcs from state i to state j, then $R_{ij}^{(0)} = \mathbf{a}_1 + \mathbf{a}_2 + \cdots + \mathbf{a}_k$.

However, if $i = j$, then the legal paths are the path of length 0 and all loops from i to itself. The path of length 0 is represented by the regular expression ϵ, since that path has no symbols along it. Thus, we add ϵ to the various expressions devised in (a) through (c) above. That is, in case (a) [no symbol a] the expression becomes ϵ, in case (b) [one symbol a] the expression becomes $\epsilon + \mathbf{a}$, and in case (c) [multiple symbols] the expression becomes $\epsilon + \mathbf{a}_1 + \mathbf{a}_2 + \cdots + \mathbf{a}_k$.

INDUCTION: Suppose there is a path from state i to state j that goes through no state higher than k. There are two possible cases to consider:

1. The path does not go through state k at all. In this case, the label of the path is in the language of $R_{ij}^{(k-1)}$.

2. The path goes through state k at least once. Then we can break the path into several pieces, as suggested by Fig. 3.3. The first goes from state i to state k without passing through k, the last piece goes from k to j without passing through k, and all the pieces in the middle go from k to itself, without passing through k. Note that if the path goes through state k only once, then there are no "middle" pieces, just a path from i to k and a path from k to j. The set of labels for all paths of this type is represented by the regular expression $R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$. That is, the first expression represents the part of the path that gets to state k the first time, the second represents the portion that goes from k to itself, zero times, once, or more than once, and the third expression represents the part of the path that leaves k for the last time and goes to state j.

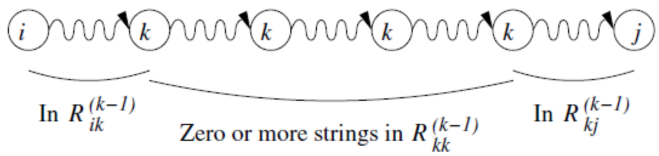


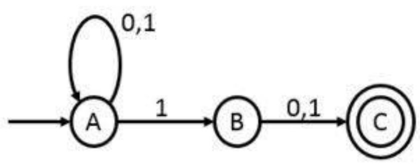
Figure 3.3: A path from i to j can be broken into segments at each point where it goes through state k

When we combine the expressions for the paths of the two types above, we have the expression

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$$

for the labels of all paths from state i to state j that go through no state higher than k. If we construct these expressions in order of increasing superscript,

c) Convert the following NFA to DFA using lazy evaluation method



	0	1
A	A	A,B
B	C	C
C	**ϕ**	**ϕ**

$\delta(A,0)=A$
$\delta(A,1)=\{A,B\}$

$\delta(\{A,B\},0)= \delta(A,0) \cup\delta(B,0)=\{A,C\}$

$\delta(\{A,B\},1)= \delta(A,1) \cup\delta(B,1)=\{A,B,C\}$

$\delta(\{A,C\},0)= \delta(A,0) \cup\delta(C,0)=\{A\}$

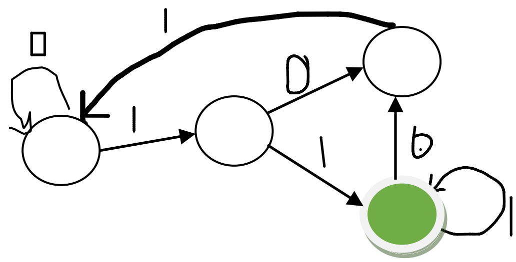
$\delta(\{A,C\},1)= \delta(A,1) \cup\delta(C,1)=\{A,B\}$

$\delta(\{A,B,C\},0)= \delta(A,0) \cup\delta(B,0) \cup\delta(C,0)=\{A,C\}$

$\delta(\{A,B,C\},1)= \delta(A,1) \cup\delta(B,0 \ \cup\delta(C,1)=\{A,B,C\}$

5

	0	1
{A}	{A}	{A,B}
{A,B}	{A,C}	{A,B,C}
{A,C}	{A}	{A,B}
{A,B,C}	{A,C}	{A,B,C}

Course Outcomes meant to be assessed by the IA Test-I:

CO1. Understand and design finite automata for various applications.

CO2. Analyze the need to represent regular languages using finite automata and regular expressions and vice versa.
