THEORY OF COMPUTATION

IN C.S.

FINITE AUTOMATA AND REGULAR EXPRESSION 1

PREVIOUS YEARS QUESTIONS

PART-A

Q.1 What is Moore Machine? *[R.T.U. 2019]*

Ans. Moore Machine : Moore Machine is an FSM whose outputs depend on only present state. It can be described by a 6 tuple $(Q, \Sigma, O, \delta, X, g_0)$ where :

- Q is a finite set of states.
- Σ is a finite set of symbols called input alphabets.
- O is a finite set of symbols called output alphabets.
- δ is input transition function where $\delta : Q \times \Sigma \to Q$
- X is the output transition function where $X : Q \times \Sigma \to 0$
- g_0 is initial state from where any input is processed ($g_0 \in Q$). State diagram is as shown below :

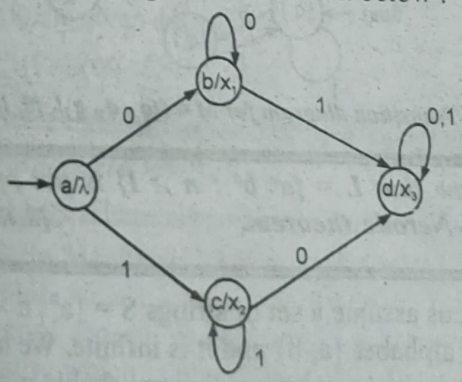


Fig.

Q.2 What is NDFA? *[R.T.U. 2019]*

Ans. Non-deterministic Finite Automaton (NDFA) : In NDFA, for a particular input symbol, the machine can move to any combination of the states in the machine. In other words, the exact state to which the machine moves cannot be determined. Hence, it is called **Non-deterministic Automaton.** As it has finite number of states, the machine is called **Non-deterministic Finite Machine** or **Non-deterministic Finite Automaton.**

Formal Definition of an NDFA

An NDFA can be represented by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where :

- Q is a finite set of states.
- Σ is a finite set of symbols called the alphabets.
- δ is the transition function where $\delta : Q \times \Sigma \to 2^Q$

(Here the power set of Q (2^Q) has been taken because in case of NDFA, from a state, transition can occur to any combination of Q states.)

- q_0 is the initial state from where any input is processed ($q_0 \in Q$).
- F is a set of final state/states of Q ($F \subseteq Q$).

Q.3 What is finite automata? *[R.T.U. 2019]*

Ans. Finite Automaton : A finite automaton can be represented by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

(i) Q is a finite non-empty set of states.

(ii) Σ is a finite non-empty set of inputs called input alphabets.

(iii) δ is a function which maps $Q \times \Sigma$ into Q and is usually called direct transition function. This is the function which describes the changes of states during the transition. This mapping is usually represented by a transition table or a transition diagram.

(iv) $q_0 \in Q$ is the initial state.

(v) $F \subseteq Q$ is the set of final states. It is assumed here that there may be more than one final state.

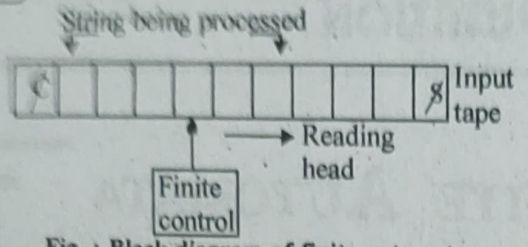


Fig. : Block diagram of finite automaton

(vi) Input Tape : The input tape is divided into squares, each square containing a single symbol from the input alphabet Σ. The end square of the tape contain end-markers ¢ at the left end and $ at the right end. Absence of end-markers indicates that the tape is of infinite length. The left-to-right sequence of symbol between the end markers is the input string to be processed.

(vii) Reading Head : The head examines only one square at a time and can move one square either to the left or to the right. For further analysis, we restrict the movement of R-head only to the right side.

(viii) Finite Control : The input to the finite control will be usually : symbol under the R-head, say a, or the present state of the machine, say q, to give the following outputs : (a) A motion of R-head along the tape to the next square (In some a null move *i.e.* R-head remaining to the same square is permitted); (b) The next state of a finite state machine given by δ (q, a).

Initial states are q_0 and q_1.

Final state is q_3.

For checking the string acceptability, we start the string from initial state. If we reach the final state after completing the string, then we say that this string is accepted by transition system or not.

Q.4 What are graphs?

[R.T.U. 2019]

Ans. Graph : A graph is a picture designed to express words, particularly the connection between two or more quantities.

A simple graph usually shows the relationship between two numbers or measurements in the form of a grid. If this is a *rectangular graph* using Cartesian coordinate system, the two measurements will be arranged into two different lines at right angle to one another. One of these lines will be going up (the *vertical* axis). The other one will be going right (the *horizontal* axis). These lines (or *axes*, the plural of axis) meet at their ends in the lower left corner of the graph.

Both of these axes have *tick marks* along their lengths. So each measurement is indicated by the length of the associated tick mark along the particular axis.

A graph is a kind of chart or diagram. However, a chart or a diagram may not relate one quantity to other quantities. *Flowcharts* and *tree diagrams* are charts or diagrams are not graphs.

Q.5 ***$M = (\{q_1, q_2, q_3\}, \{0, 1\}, \delta, q_1, \{q_3\})$ is nondeterministic finite automaton, where δ is given by***

$\delta(q_1, 0) = \{q_2, q_3\}$,
$\delta(q_1, 1) = \{q_1\}$
$\delta(q_2, 0) = \{q_1, q_2\}$,
$\delta(q_2, 1) = \phi$
$\delta(q_3, 0) = \{q_2\}$,
$\delta(q_3, 1) = \{q_1, q_2\}$

Construct an equivalent DFA.

[R.T.U. 20...

Ans. $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \delta, q_1, \{q_3\})$

Where :

$\delta(q_1, 0) = \{q_2, q_3\}$, $\delta(q_1, 1) = \{q_1\}$
$\delta(q_2, 0) = \{q_1, q_2\}$, $\delta(q_2, 1) = \phi$
$\delta(q_3, 0) = \{q_2\}$, $\delta(q_3, 1) = \{q_1, q_2\}$

The DFA equivalent of this NFA can be obtained as follows :

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

The transition diagram associated with this DFA is shown in following figure.

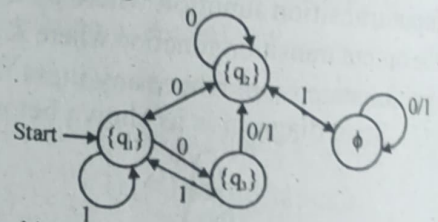


Fig. : Transition diagram for $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \delta, q_1, \{q_3\})$

Q.6 ***Show that $L = \{a^n b^n : n \geq 1\}$ is not regular using Mayhill-Nerode theorem.***

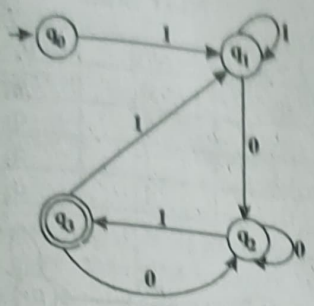
[R.T.U. 2013, 2012]

Ans. Let us assume a set of strings $S = \{a^n \mid n > 0\}$. The set S is over alphabet {a, b} and it is infinite. We have to show that its strings are pairwise distinguishable with respect to language L.

Assume that a^i and a^j be arbitrary two different members of the set S, where i and j are positive integers and $i \neq j$.

Let b^j as a string to be appended to a^i and a^j. It then becomes $a^i b^j$ and $a^j b^j$. From these two strings $a^i b^j$ is not L while $a^j b^j$ is in L. hence we can say that a^i and a^j are arbitrary strings of S and are distinguishable with respect to L. Thus S satisfies conditions of Myhil-Nerode theorem. This proves that L is non-regular.

Q.7 Find the regular expression corresponding to the given automaton.



[Raj. Univ. 2007]

Ans. We get the following equations :

$$q_0 = \wedge$$

$$q_1 = q_0 1 + q_1 1 + q_3 1$$

$$q_2 = q_1 0 + q_2 0 + q_3 0$$

$$q_3 = q_2 1$$

Therefore,

$$q_2 = q_1 0 + q_2 0 + q_2 10 = q_1 0 + q_2 (0+10)$$

Applying Arden's theorem, we get

$$q_2 = q_1 0(0+10)^*$$

Now $$q_1 = 1 + q_1 1 + q_2 11 = 1 + q_1 1 + q_1 0(0+10)^* 11$$

$$= 1 + q_1 \left(1 + 0(0+10)^* 11\right)^*$$

By Arden's theorem we get,

$$q_1 = 1\left(1 + 0(0+10)^* 11\right)^*$$

$$q_3 = q_2 1 = 1\left(1 + 0(0+10)11\right)^* 0(0+10)^* 1$$

As q_3 is only the final state, the regular expression corresponding to the given diagram is

$$1\left(1 + 0(0+10)^* 11\right)^* 0(0+10)^* 1$$

Q.8 Define finite state machine.

Ans. Finite State Machine (FSM) : Finite state machine model is a simple, widely known and important model for describing control aspects. It is a model of the system to describe an entity which is characterized by its operations and behavior. Such machines are basically appropriate tools to express the requirements and design specifications of software product.

FSM models are represented by a set of states and specifies transitions between the states. The transition from one state to another state is based on the input. So, such models show system states and events which cause transition from one state to another.

A Finite State Machine (FSM) consists of a finite set of symbols that are given below.

$\mathfrak{J}$ = a finite set of states

$\mathfrak{E}$ = a finite set of input

$\mathfrak{S}$ = the starting state

δ = a transition, from (input × state) to 2 state $\mathfrak{S}$,

which can be a partial function, *i.e.* can be undefined for some values of its domains.

δ : Input × state → state

$\delta : \mathfrak{E} \times \mathfrak{J} \to \mathfrak{J}$

These notations are used to make the finite state machine diagram or transition tables. To draw a diagram, first, we draw a circle for each state in the diagram. Then for each state and possible input, we draw a directed arrow to indicate the transition from one state to another.

Q.9 Define in brief Discrete Automaton.

Ans. An automaton is an abstract model of a digital computer. It has a mechanism to read input, which is a string over a given alphabet. In computer science, the term automaton means **discrete automaton** and is defined in some abstract way as shown in Figure.

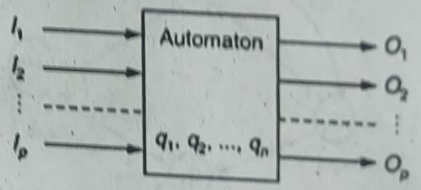


Fig. : Model of a discrete automaton

Q.10 Difference between melay & moore machine.

Ans. Mealy Machine v/s Moore Machine

S. No.	Mealy Machine	Moore Machine
(i)	Output depends both upon present state and present input.	Output depends only upon present state.
(ii)	Generally, it has fewer states than Moore machine.	Generally, it has more states than Mealy machine
(iii)	Output changes at clock edges.	Input change can cause change in output change as soon as logic is done.
(iv)	Mealy machines react faster to inputs.	In moore machines, more logic is needed to decode the output since it has more circuit delays.

Q.11 *Give applications of pumping lemma.*

Ans.(i) Assume L is regular : Let n be the number of state in the FA which accepts L.

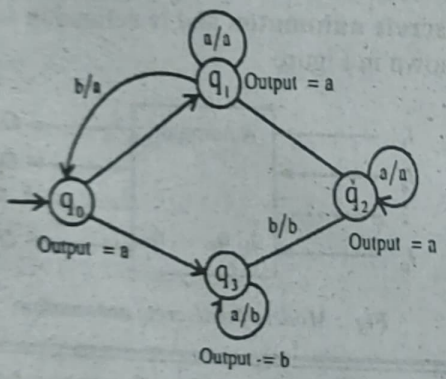
(ii) Choose a string w such that $|w| \geq n$. Using pumping lemma write $w = xyz$ with $|xy| \leq n$ and $|y| > 0$.

(iii) Find suitable integer i such that $xy^i z \notin L$. So it contradicts the assumption we have made in step (i) above. Hence L is not regular.

It is then clear that, the most important aspect of the method is to find i such that $xy^i z \notin L$. Some times we have to prove $xy^i z \notin L$ by considering $|xy^i z|$ or some times we heed to use the structure of strings in L.

PART-B

Q.12 *Consider the Moore Machine shown in figure. What is the output for the input ababa?*



[R.T.U. 2019]

Ans. For the input = ababa

$$\underset{q_0}{a} \xrightarrow{a} \underset{q_1}{a} \xrightarrow{b} \underset{q_0}{a} \xrightarrow{a} \underset{q_1}{a} \xrightarrow{b} \underset{q_0}{a} \xrightarrow{a} \underset{q_1}{a}$$

The ouput for the input string = ababa is aaaaaa

Q.13 *Convert the following Moore Machine into Mealy Machine :*

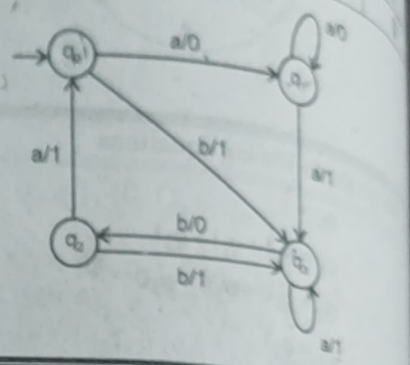
State	Input		Output
	a	*b*	
$\rightarrow q_0$	q_1	q_3	*1*
q_1	q_3	q_1	*0*
q_2	q_0	q_3	*0*
q_3	q_3	q_2	*1*

[R.T.U. 2019]

Ans. Mealy Machine :

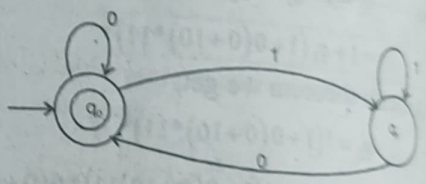
Next State

P.S.	i = a	o/p	i/p = b	o/p
$\rightarrow q_0$	q_1	0	q_3	1
q_1	q_3	1	q_1	0
q_2	q_0	1	q_3	1
q_3	q_3	1	q_2	0



Q.14 *Design a FA which checks whether the given number is even.*

Ans.



Q.15 *Explain the difference between deterministic non-deterministic finite automaton.*

OR

State the difference between deterministic and deterministic finite automata.

Ans. Deterministic and Non-deterministic Finite Acceptors

S. No.	Deterministic Finite Acceptors	Non- deterministic Finite Acceptors
1.	For every symbol of the alphabet, there is only one state transition.	We do not need to specify how does the NFA react according to some symbol.
2.	Cannot use empty string transition.	Can use empty string transition.
3.	Can be understood as one machine.	Can be understood as multiple little machines computing at the same time.
4.	It will reject the string if it end at other than accepting state.	If all of the branches of NFA dies or rejects the string, we can say that NFA reject the string.

	Deterministic Finite Acceptors	Non- deterministic Finite Acceptors
1.	For every symbol of the alphabet, there is only one state transition.	We do not need to specify how does the NFA react according to some symbol.
2.	Cannot use empty string transition.	Can use empty string transition.
3.	Can be understood as one machine.	Can be understood as multiple little machines computing at the same time.
4.	It will reject the string if it end at other than accepting state.	If all of the branches of NFA dies or rejects the string, we can say that NFA reject the string.
5.	The transition function is single valued.	The transition function is multi-valued.
6.	Checking membership is easy.	Checking membership is difficult.
7.	Construction is difficult.	Construction is easy.
8.	Space required is more.	Space required is comparatively less.
9.	Backtracking is allowed.	Not possible in every case.
10.	Can be constructed for every input and output.	Cannot be constructed for every input and output.
11.	There can be more than one final state.	There can only be one final state.

Q.16 (a) ***Describe the block diagram of a finite automaton. Consider the transition system given below :***

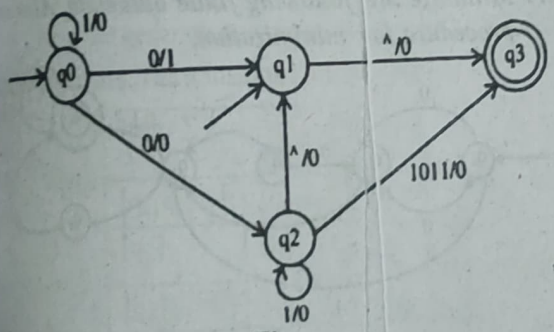


Fig.

Determine the initial states, the final state and the acceptability of 101011 and 111010.

(b) ***Prove that for any transition function δ and for any two input string x and y.***

$\delta(q, xy) = \delta(\delta(q, x), y)$ *[R.T.U. 2016]*

Ans.(a) Finite Automaton : *Refer to Q. 3.*

For string 101011, the path value is $q_0\ q_0\ q_2\ q_3$. Since q_3 is the final state so this string is accepted by above transition system.

For string 111010, there is no path value. So this string is not accepted by the above transition system.

Ans.(b) Properties of Transition Functions

(1) $\delta(q, \wedge) = q$ is a finite automaton. This means the state of the system can be changed only by an input symbol.

(2) For all strings w and input symbols a,

$$\delta(q, aw) = \delta(\delta(q, a), w)$$

$$\delta(q, wa) = \delta(\delta(q, w), a)$$

This property gives the state after the automaton consumers or reads the first symbol of a string aw and the state after the automation consumers a prefix in of the string wa.

Example : Prove that for any transition function δ and for any two input string x and y, ...(1)

$$\delta(q, xy) = \delta(\delta(q, x), y)$$

Proof : By the method of induction on $|y|$, i.e. length of y.

Basis : When $|y| = 1$, $y = a \in \Sigma$,

L.H.S. of equation (1)

$= \delta(q, xa) = \delta(\delta(q, x), a)$ (by property 2)

$=$ R.H.S. of (1)

Assume the result, *i.e.* (1) for all strings x and string y with $|y| = n$. Let y be a string of length $n + 1$. Write $y_1 = y, a$ where $|y| = n$.

L.H.S of (1)

$= \delta(q, xy_1 a) = \delta(q, xy_1 a), x_1 = xy_1$

$= \delta(\delta(q, x_1), a)$ (by property 2)

$= \delta(\delta(q, xy_1), a)$

$= \delta(\delta(\delta(q, x), y_1), a)$

(by induction hypothesis)

R.H.S. of (1) $= \delta(\delta(q, x), y, a)$

$= \delta(\delta(\delta(q, x), y_1), a)$

(by property 2)

Hence, L.H.S. = R.H.S. This proves for any string of length $n + 1$. By the principle of induction (1) is true for all strings.

Q.17 ***Write short note on Finite State Machine (FSM).***

[RTU 2017, 2016, 2013, 12]

OR

Explain Finite State Machine (FSM) Models.

[RTU 2014]

Ans. Finite State Machine (FSM) : *Refer to Q.8.*

Example 1 : Draw a FSM model for the following description :

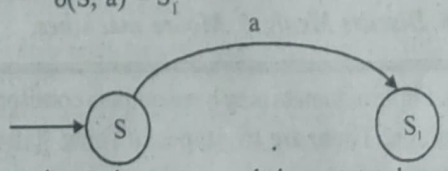
$T = \{S, S_1\}$

$\varepsilon = \{a\}$

$S =$ starting state

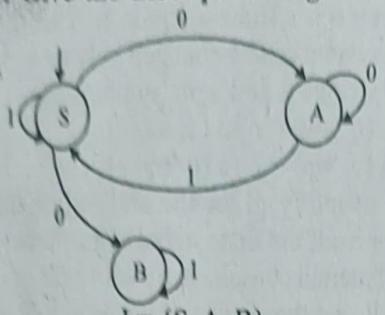
$\delta(S, a) = S_1$

Ans.



When an input does not result in a state change, we can indicate this in the diagram by showing an arrow originating and ending in the same state.

Example 2 : Give the description of given FSM :



Ans. J = {S, A. B}

ε = {0, 1}, S = Starting state

δ (S, 0) = A, δ(S, 1)= S

δ (A, 0) = A, δ(A, 1) = S

δ (B, 0) = B , δ (B, 1) = B

Example 3 : The FSA in given fig. recognizes the following set of words : {bet, ball, beg} :

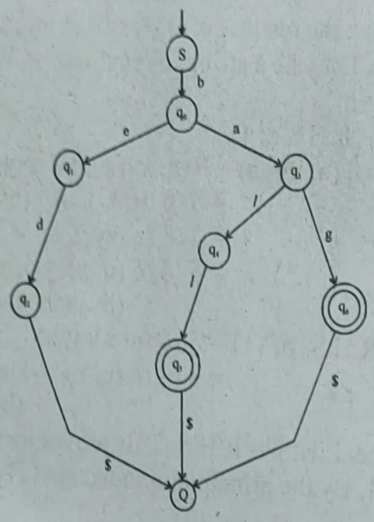


Fig.

Ans. ε = {b, e, d, a, l, g}

J = {s, q_0, q_1, q_2, q_3, q_4, q_5, q_6, Q }

S = Starting state

A = Q (Accepting state)

δ(s, b) = q_0, δ (q_3, 1) = q_4

δ (q_0, e) = q_1 , δ (q_3, g) = q_6

δ (q_0, a) = q_3, δ (q_4, 1) = q_5

δ (q_1,d) = q_2, δ(q_5, $) = Q

δ (q_2, $) = Q δ ($q_6$, $) = Q

and f = {q_2, q_5, q_6}

Q.18 *Discuss Mealy & Moore machines.* *[R.T.U. 2015]*

Ans. Finite automata may have outputs corresponding to each transition. There are two types of finite state machine that generate output

(i) Mealy Machine

(ii) Moore Machine

Mealy Machine:

Mealy Machine is an FSM whose output depends present state as well as present input

It can be described by a six tuple (Q, Σ, O, δ, X, where :

- Q is finite set of states
- Σ is finite set of symbols called input alphabe
- O is finite set of symbols called output alphab
- δ is input transition function where δ : Q X Σ
- X is output transition function where X : Q
- g_0 is initial state from where any input is proce (g_0, ∈Q). State diagram of Mealy machine is sh below :

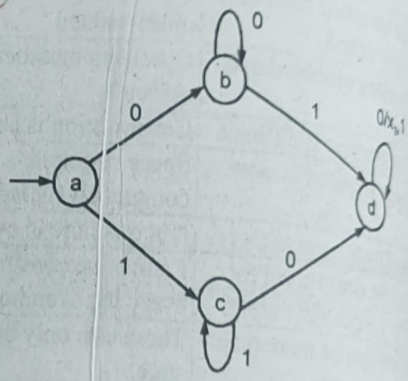


Fig.

Moore Machine: *Refer to Q.1.*

Difference between Mealy and Moore Machine : *to Q. 10.*

Q.19 *Minimize the following finite automata. Also wr procedure for minimization.*

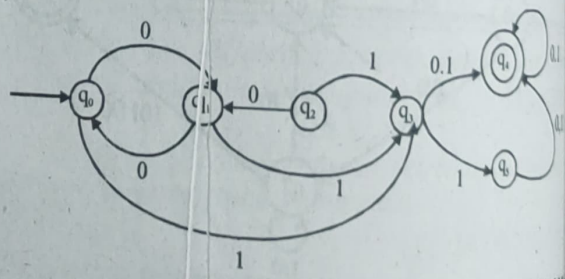


Fig. *[R.T.U. 201*

Ans. Step 1 : Firstly we make state transition table for abo given state transition diagram is given by as follows.

State / Σ	Input	
	0	1
q_0	q_1	q_3
q_1	q_0	q_3
q_2	q_1	q_3
q_3	q_4	$[q_4, q_5]$
q_4	q_4	q_4
q_5	q_4	q_4

Step 2 : Now we obtain π_0 by use of grouping

$$\pi_0 = \{\{q_4\}, \{q_0, q_1, q_2, q_3, q_5\}\}$$

Step 3 : Partition set of non final states in such a way that equivalent states are grouped together in separate sets. This is π_1 equivalence criteria used here is:

Two states are said to be equivalent if their state transitions corresponding to same input belong to same set.

Step 4 : The step 3 is repeated for π_2, π_3 and so on until we get

$$\pi_k = \pi_{k+1}$$

where k is any integer and π_{k+1} is required solution.

Step 5 : In this step state transition table is generated in such a way that the set obtained in π_{k+1} are considered to be as one state, the rest of state transitions in the state transition table goes according to original state transition table.

Step 6 : State transition diagram is generated from state transition table.

after applying step 3 we get π_1

$$\pi_1 = \{\{q_4\}, \{q_0, q_1, q_2\}, \{q_3\}, \{q_5\}\}$$

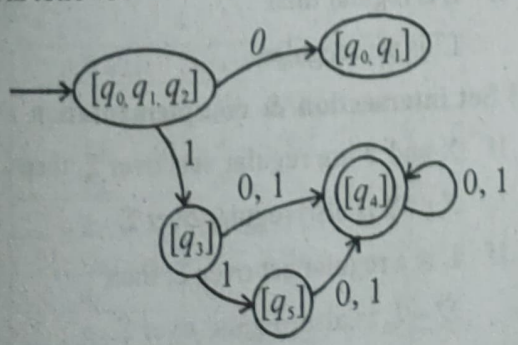
$$\pi_2 = \{\{q_4\}, \{q_0, q_1, q_2\}, \{q_3\}, \{q_5\}\}$$

we get $\pi_2 = \pi_1$ so we stop here.

Now we generate state transition table for minimizing finite automata. The transition table is as follows:

State / Σ	Input	
	0	1
$[q_0, q_1, q_2]$	$[q_0, q_1]$	$[q_3]$
$[q_3]$	$[q_4]$	$[q_4, q_5]$
$[q_4]$	$[q_4]$	$[q_4]$
$[q_5]$	$[q_4]$	$[q_4]$

State transition diagram for above state transition table is given as follows :



Q.20 ***Construct a deterministic finite automata equivalent to following NDFA.***

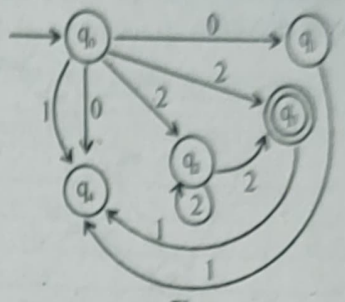
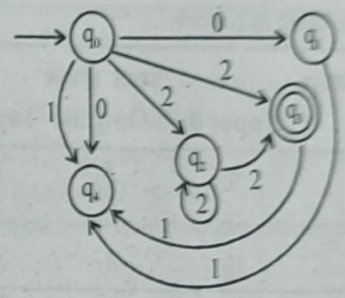


Fig. *[R.T.U. 2013]*

Ans.

Given, NDFA



Step (1) : Find transition table of given NDFA

Table : State table of given NDFA

State (Σ)	0	1	2
→ q_0	q_1, q_4	q_4	q_3, q_2
q_1		q_4	
q_2			q_3, q_2
q_3		q_4	
q_4			

Step (2) : Now,

(i) Start with starting state that is q_0 we get $[q_1, q_4]$, $[q_4]$ and $[q_3, q_2]$ as new states.

(ii) Then we construct δ (transition function) for $[q_4]$ and we do not get any new state.

(iii) Then we construct δ for $[q_1, q_4]$ and we do not get any new state.

(iv) Then we construct δ for $[q_3, q_2]$ and we do not get any new state.

Step (3) : Now, we construct the table for DFA of given NDFA using states generated in step (2).

Table : State table for equivalent DFA of given NDFA

State (Σ)	0	1	2
$[q_0]$	$[q_1, q_4]$	$[q_4]$	$[q_3, q_2]$
$[q_4]$	ϕ	ϕ	ϕ
$[q_1, q_4]$	ϕ	$[q_4]$	ϕ
$[q_3, q_2]$	ϕ	$[q_4]$	$[q_3, q_2]$

Q.21 ***Consider a Mealy machine given by transition diagram. Construct a moore machine equivalent to this mealy machine.***

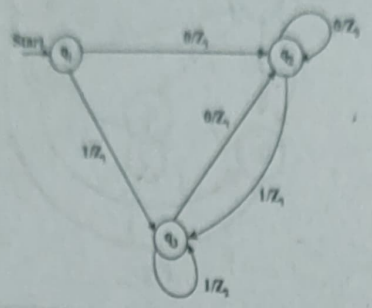


Fig. *[R.T.U. 2013, 2009]*

Ans. First of all we have to convert the transition diagram into transition table as follows :

Present state	Next state			
	Input 0	Output	Input 1	Output
→ q_1	q_2	Z_1	q_3	Z_1
q_2	q_2	Z_2	q_3	Z_1
q_3	q_2	Z_1	q_3	Z_2

Step 1 : We look in the next state column for states q_1, q_2 and q_3.

q_1 is associated with no output, q_2 and q_3 are associated with two different outputs *i.e.* Z_1 and Z_2.

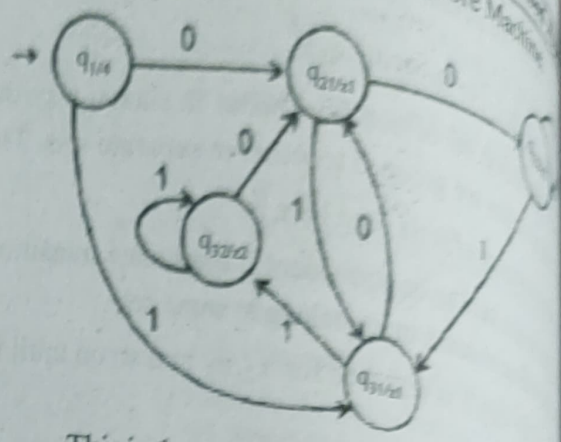
Step 2 : We don't split q_1 now we split q_2 and q_3 into q_{21}, q_{22} and q_{31}, q_{32} respectively. After splitting these state, the transition table we get is as follows :

Present state	Next state			
	Input 0	Output	Input 1	Output
→ q_1	q_{21}	Z_1	q_{31}	Z_1
q_{21}	q_{22}	Z_2	q_{31}	Z_1
q_{22}	q_{22}	Z_2	q_{31}	Z_1
q_{31}	q_{21}	Z_1	q_{32}	Z_2
q_{32}	q_{21}	Z_1	q_{32}	Z_2

Step 3 : Now we rearrange the transition table in such a way that each state in present state column is associated with a single output.

Present state	Next state		Output
	Input 0	Input 1	
→ q_1	q_{21}	q_{31}	0
q_{21}	q_{22}	q_{31}	Z_1
q_{22}	q_{22}	q_{31}	Z_2
q_{31}	q_{21}	q_{32}	Z_1
q_{32}	q_{21}	q_{32}	Z_2

The transition diagram for Moore Machine



This is the required solution.

Q.22 ***Write closure property of regular set.***

Ans. The closure properties of regular sets are as follows:

(i) Set union

(ii) Concatenation

(iii) Closure (iteration)

(iv) Transpose

(v) Set intersection

(vi) Complementation

We can understand these all closure properties as follows:

(i) Set Union: The union of two regular expressions R_1 and R_2, written as $R_1 + R_2$ is also a regular expression.

(ii) Concatenation: The concatenation of two regular expressions R_1 and R_2, written as R_1R_2 is also a regular expression.

(iii) Closure (iteration): The closure (or iteration) of a regular expression R, written as R^* is also a regular expression.

(iv) Transpose :

If L is regular then

L^T is also regular

(v, vi) Set intersection & complementation :

If X and Y are regular sets over Σ, then

$X \cap Y$ is also regular over Σ

If L is a regular set over Σ, then

$\Sigma^* - L$ is also regular over Σ.

Q.23 ***Construct a Moore machine equivalent to the Mealy machine M defined by the table given below :***

Present state	Next state			
	a = 0		a = 1	
	State	o/p	State	o/p
q_1	q_1	1	q_2	0
q_2	q_4	1	q_4	1
q_3	q_2	1	q_3	1
q_4	q_3	0	q_1	1

[R.T.U. 2012]

Ans. Convert Mealy machine equivalent to the Moore machine :

Present State	Next State			
	a=0	Output	a=1	Output
$\rightarrow q_1$	q_1	1	q_2	0
q_2	q_4	1	q_4	1
q_3	q_2	1	q_3	1
q_4	q_3	0	q_1	1

Step 1 : We look in the next state column for q_1, q_2, q_3 and q_4

q_1 is associated with one output : 1

q_2 is associated with two outputs : 0 & 1

q_3 is associated with two outputs : 0 & 1

q_4 is associated with one output : 1

Step 2 : Now we split q_1 and q_3 into q_{20}, q_{21} & q_{30}, q_{31} respectively.

Present State	Next State			
	a=0	Output	a=1	Output
$\rightarrow q_1$	q_1	1	q_{20}	0
q_{20}	q_4	1	q_4	1
q_{21}	q_4	1	q_4	1
q_{30}	q_{21}	1	q_{31}	1
q_{31}	q_{21}	1	q_{31}	1
q_4	q_3	0	q_1	1

Step 3 : Now rearranging the transition table in such a way the each state in present state column is associate with a single output.

Present state	Next state		Output
	a=0	a=1	
$\rightarrow q_1$	q_1	q_{20}	1
q_{20}	q_4	q_4	0
q_{21}	q_4	q_4	1
q_{30}	q_{21}	q_{31}	0
q_{31}	q_{21}	q_{31}	1
q_4	q_3	q_1	1

This is Moore machine.

Note : Here, we see that the initial state q_1 is associated with output 1. We add a new starting state q_0 whose transitions are similar to q_1 but output is 0.

So state transition table is transformed as follows :

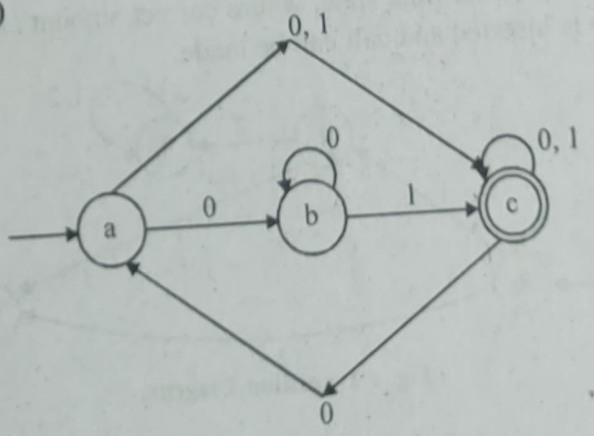
Present state	Next state		Output
	a=0	a=1	
$\rightarrow q_0$	q_1	q_{20}	0
q_1	q_1	q_{20}	1
q_{20}	q_4	q_4	0
q_{21}	q_4	q_4	1
q_{30}	q_{21}	q_{31}	0
q_{31}	q_{21}	q_{31}	1
q_4	q_3	q_1	1

Q.24 ***Let $\Sigma = \{a, b, c\}$***

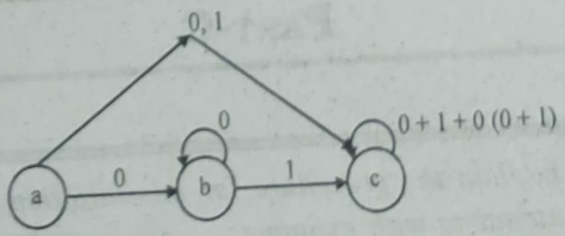
(i) Draw a DFA that rejects all words for which the last two letters match.

(ii) Draw a DFA that rejects all words for which the first two letters match. *[R.T.U. 2010]*

Ans.(i)



(ii)



Q.25 ***A public telephone (PCO) accepts one or rupees two coins. The call can be made only when the total amount inserted is rupees two. Suppose the telephone has two LED : GREEN and RED. The GREEN LED is set when the call is being made. The RED LED is set when the total amount inserted is rupees three or more. This call is possible only when, RED is off. Construct a DFA corresponding to this machine.*** *[R.T.U. 2009]*

Ans. PCO Telephone Light

Input	1	2
q_0	q_1	q_f
q_1	q_f	q_3
q_3	q_3	q_3
q_f	–	–

The States

q_0 = Initial state q_1 = When amount inserted is 1.

q_3 = Red light glowing *i.e.* amount more than 2.

q_f = Green light, call can be made.

In the initial state q_0, when 1 rupee coin is inserted, it goes to state q_1 where it waits for another 1 rupee.

In the q_1, 1 rupee is inserted and it waits for another '1'.

If '1' then 'q_f' *i.e.* call can be made.

If '2' then 'q_3' *i.e.* red light glows as total amount is $1 + 2 = 3$.

In q_3, any rupee entered will continue glowing red light as amount ≥ 3.

q_f is the final state, where correct amount *i.e.* rupees two is inserted and call can be made.

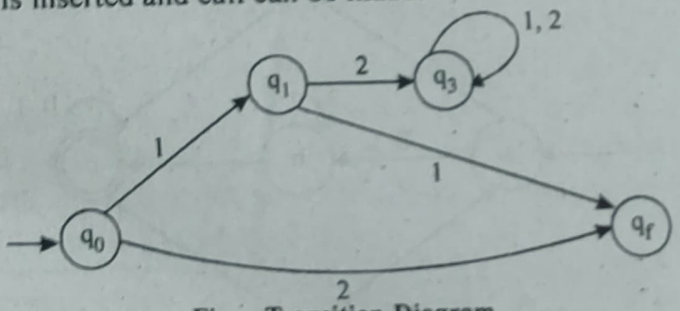


Fig. : Transition Diagram

PART-C

Q.26 Explain the procedure for minimization of finite automata with example. ***[R.T.U. 2019, 13]***

OR

Describe the algorithm to minimize number of states in a finite automaton. ***[Raj. Univ. 2003]***

Ans. Procedure : We construct an automaton with minimum number of states equivalent to given automaton M.

ALGO

Step 1: When we are asked to minimize a finite automaton, we are given a state transition table or a state transition diagram. The state transition diagram must be converted into a state transition table.

Step 2: Now we apply the procedure for partition Q, where partitions are denoted by $\pi_0, \pi_1, \pi_2 \ldots$ and π_0 is obtained by grouping final states in one set and non-final states in another state.

Step 3: Now we leave the set containing the final state as such and partition the other set of non-final states separate status. This is π_1.

Step 4: The step 3 is repeated for π_2, π_3 and so on we get $\pi_k = \pi_k + 1$. Where k is any integer. $\pi_k + 1$ is the required solution.

Step 5: In this step the state transition table is generated in such a way that the states obtained in $\pi_k + 1$ are considered to be as one state. The rest of the state transition in the transition table goes according to the original state transition table.

Step 6: State transition diagram is generated from the state transition table.

Example: Suppose we have a finite automata as described in following fig:

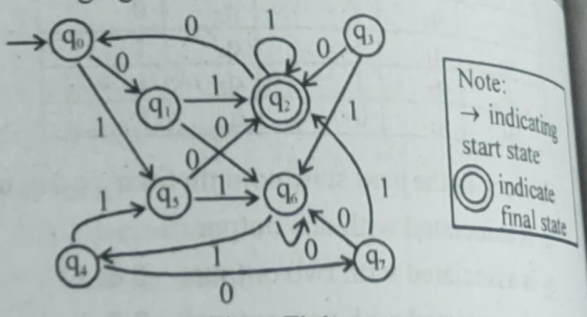


Fig. : Finite automation

Now, we can construct the transition table of the automaton as follows:

Table : Transition table of above automaton

	State (Σ)	0	1
(Start in state) →	q_0	q_1	q_5
	q_1	q_6	q_2
(Final state)	$\boxed{q_2}$	q_0	q_6
	q_3	q_2	q_6
	q_4	q_7	q_5
	q_5	q_2	q_6
	q_6	q_6	q_4
	q_7	q_6	q_2

Now,

Step (1) : Divide all states in two sets, one of final states and another non final states as follows:

$$\pi_0 = \{ \underbrace{\{q_2\}}_{\text{final state(s)}}, \underbrace{\{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}}_{\text{Non final states}} \}$$

Step (2) : Now reconstruct the group by considering any one of the state q_3, q_5, q_7. With the help of transition we find that q_1 is not 1-equivalent to q_3 and q_5 but 1-equivalent to q_7. Hence we form a new group as $\{q_1, q_7\}$.

Similarly, we form other groups

$\{q_2, q_5\}$

$\{q_0, q_4, q_6\}$

So, now we have

$\pi_1 = \{\{q_2\}, \{q_0, q_4, q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$

Note: To form groups, we should follow these steps:

Suppose we are matching q_1 and q_4

Now at 0 q_1 goes to q_6 and at 0 q_4 goes to q_7 } Same group

but at 1 q_1 goes to q_2 and at 1 q_4 goes to q_5 } Different group

So, q_1, q_4 cannot form a group

Now, Suppose we are matching q_1 and q_7

at 0 q_1 goes to q_6 and at 0 q_7 goes to q_6 } Same group

and at 1 q_7 goes to q_2 and at 1 q_1 goes to q_2 } Same group

Step (3) : Now following step (2) to form further group

$\pi_2 = \{\{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$

Step (4) : Similarly,

$\pi_3 = \{\{q_2\}, \{q_0, q_4\}, \{q_6\}, \{q_1, q_7\}, \{q_3, q_5\}\}$

Step (5) : As $\pi_2 = \pi_3$

So we have no need to form further groups.

Step (6) : Reconstruct the transmission table according to π_2 or π_3.

Table : Transition table after minimization

State (Σ)	0	1
$\{q_0, q_4\}$	$\{q_1, q_7\}$	$\{q_3, q_5\}$
$\{q_1, q_7\}$	$\{q_6\}$	$\{q_2\}$
$\{q_2\}$	$\{q_0, q_4\}$	$\{q_2\}$
$\{q_3, q_5\}$	$\{q_2\}$	$\{q_6\}$
$\{q_6\}$	$\{q_6\}$	$\{q_0, q_4\}$

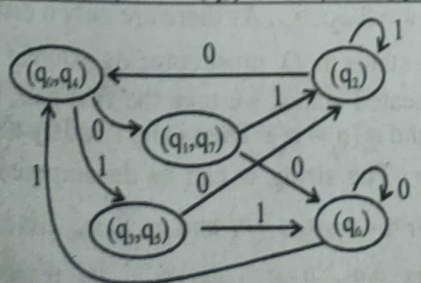


Fig. : Minimized automaton

Q.27 What do you understand by finite automata and regular expression. *[R.T.U. 2015]*

Ans. Finite Automata : *Refer to Q.3.*

Regular Expression : An expression R is a regular expression if R is

1. a for some a in some alphabet Σ,
2. $\in$,
3. ϕ,
4. $(R_1 \cup R_2)$ for some regular expressions R_1 and R_2,
5. $(R_1 . R_2)$ for some regular expressions R_1 and R_2, or
6. $(R_1)^*$ for some regular expression R_1.

When the meaning is clear from the context, () and . can be removed from the expression.

The Language Represented by a Regular Expression

For a regular expression R, L(R) denotes the language R expresses.

1. For each $a \in \Sigma$, $L(a) = \{a\}$.
2. $L(\in) = \{\in\}$
3. $L(\phi) = \phi$
4. $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$
5. $L(R_1 . R_2) = L(R_1) . L(R_2) = \{uv \mid u \in R_1 \text{ and } v \in R_2\}$
6. $L(R_1^*) = \{\in\} \cup \{u_1 \ldots u_k \mid u_1, \ldots, u_k \in R_1\}$

Regular Expression Examples

- $L(a) = \{a\}$ (for a single-element regular expression, you may simply write the element)
- $L(abab \cup bc) = \{abab, bc\}$
- $L((abab \cup abc)^*) = \{\in\} \cup \{w_1 \ldots w_k \mid k \geq 1 \text{ and } w_1, \ldots, w_k \in \{abab, abc\}\}$.
- $L((abab \cup abc)^* \cup c^* \cup abc(abca)^*)$
  $= \{\in\} \cup \{w_1 \ldots . w_k \mid k \geq 1 \text{ and } w_1 \ldots ., w_k \in \{abab, abc\}\}$
  $\cup \{w \mid w \text{ is a repetition of c' s }\}$
  $\cup \{abc, abcabca, abcabcaabca, \ldots\}$

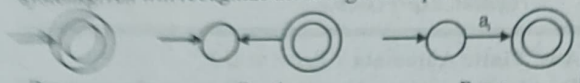
Finite Automata and Regular Expressions :

(1) Transition System and Regular Expression : The following theorem describes the relation between transition system and regular expression.

Theorem : Every regular expression R can be recognized by a transition system i.e. for every string w in the set R, there exists a path from the initial state to a final state with path value w.

Proof : The proof is by the principle of induction on the total number of characters in R. By "Character" we mean elements of $\Sigma, \wedge, \phi, *$ and $+$, for example if $R = \wedge + 10 * 11 * 0$, the characters are $\wedge + 1, 0, *, 1, 1, *, 0$ and the number of character is 9.

Suppose the number of characters in R be 1. Then $R = \wedge$, or $R = \phi$, or $R = a_i$, $a_i \in \Sigma$. The transition systems will recognize these regular expressions.



$R = \wedge$ $R = \phi$ $R = a_i$

Fig. : Transition system recognizing elementary regular sets

Induction Step : Assume the theorem is true for regular expressions with n character or less. We must prove that it is also true for n + 1 characters. Let R have n + 1 characters. Then

$$R = P + Q \text{ or } R = PQ \text{ or } R = P^*$$

where P and Q are regular expressions each having n characters or less. By induction hypothesis, P and Q can be recognized by transition system G and H, respectively.

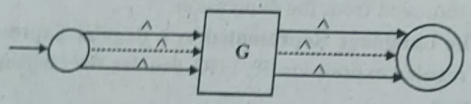


Fig. : Transition system recognizing P

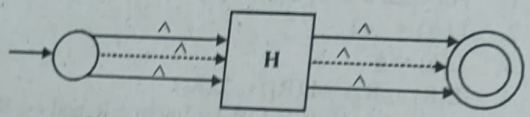


Fig. : Transition system recognizing Q

(2) Transition System Containing ∧-Moves : The transition systems can be generalized by permitting ∧-transitions or ∧-moves which are associated with a null symbol ∧. These transition can occur when no input is applied. But it is possible to convert a transition system with ∧-moves into an equivalent transition system with ∧-moves into an equivalent transition system without ∧-moves we shall give a simple method of doing it with the help of an example.

Suppose we want to replace a ∧-move from vertex v_1 to vertex v_2. Then v e proceed as follows :

Step - 1 : Find all the edges starting from v_2.

Step - 2 : Duplicate all these edges starting from v_1 without changing the edge labels.

Step - 3 : If v_1 is an initial state, make v_2 also an initial state.

Example : Consider a finite automaton with ∧-moves given in Fig. below. Obtain an equivalent automaton without ∧-moves.

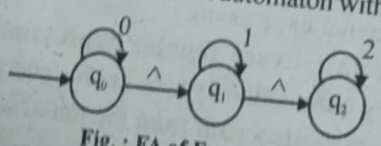
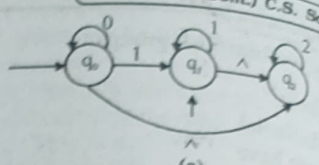
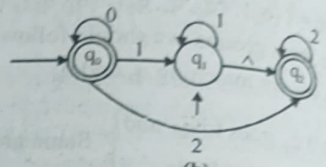


Fig. : FA of Example

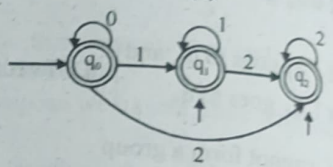
We first eliminate the ∧-moves from q_0 to q_1 to get Fig. (a). q_1 is made an initial state. Then we eliminate the ∧-moves from q_0 to q_2 in Fig. (a) to get Fig. (b). As q_2 is a final state, q_0 is also made a final state. Finally, the ∧-moves from q_1 to q_2 is eliminated in Fig. (c).



(a)



(b)



(c)

Fig. : Transition system without ∧-moves

Q.28 ***State and explain pumping lemma. Prove that the following language $L = \{a^n : n$ is a perfect square$\}$ is not regular.*** [R.T.U. 2012]

Ans. Pumping Lemma for Regular Sets : If we give a necessary condition for an input string to belong to a regular set, the result is called *pumping lemma* as it gives a method of pumping (generating) many input strings from a given string. As pumping lemma gives a necessary condition, it can be used to show that certain sets are not regular.

Theorem : (Pumping Lemma) : Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton with n states. Let L be the regular set accepted by M. Let $w \in L$ and $|w| \geq m$. If $m \geq n$, then there exists x, y, z such that $w = xyz, y \neq \Lambda$ and $xyz \in L$ for each $i \geq 0$.

Proof : Let

$$w = a_1 a_2 \ldots\ldots a_m, \quad m \geq n$$

$$\{\delta(q_0, a_1 a_2 \ldots a_i)\} = q_i$$

for $i = 1, 2, \ldots\ldots m$; $Q_1 = \{q_0, q_1, \ldots\ldots, q_m\}$

That is, Q_1 is the sequence of states in the path with path value $w = a_1 a_2 \ldots a_m$. As there are only n distinct states, at least two states in Q_1 must coincide. Among the various pairs of repeated states, we take the first pair. Let us take them as q_j and $q_k (q_j = q_k)$. Then j and k satisfy the condition $0 \leq j < k \leq n$. The string w can be decomposed into three substrings $a_1 a_2 \ldots a_j, a_{j+1} \ldots a_k$ and $a_{k+1} \ldots a_m$. Let x, y, z denote these strings $a_1 a_2 \ldots a_j, a_{j+1} \ldots a_k, a_{k+1} \ldots a_m$ respectively. As

$1 \le |y| \le n$ and $w = xyz$. The path with the path value w in the transition diagram of M is shown in fig.

The automaton M starts from the initial state q_0. On applying the string x, it reaches $q_i (= q_k)$. On applying the string y, it comes back to $q_i (= q_k)$. So after application y^i for each $i \ge 0$, the automaton is in the same state q_i. On applying z, it reaches q_m, a final state. Hence $xy^iz \in L$. As every state in Q is obtained by applying an input symbol, $y \ne \Lambda$.

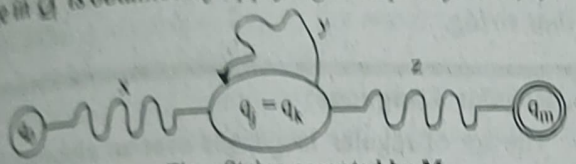


Fig. : String accepted by M

Note : The decomposition is valid only for strings of length greater than or equal to the number of states. For such a string $w = xyz$, we can iterate the substring y in xyz as many times as we like and get strings of the form xy^iz which are longer than xyz and are in L. By considering the path from q_0 to q_k and then the path from q_k to q_m (without going through the loop), we get a path ending in a final state with path value xz. (This corresponds to the case when i = 0)

Proof :

$L = \{a^n : n \text{ is a perfect square}\}$

Let $n = i^2$

(a) Given that $i \ge 1$

For $i = 1, a^{i^2} = a^{1^2} = 0$, length $= 1^2$

$i = 2, a^{i^2} = a^{2^2} =$ aaaa, length $= 2^2$

$i = 3, a^{i^2} = a^{3^2} =$ length $= 3^2$

we can see, the length of each string is a perfect square.

(b) Assume that L is a regular language

(c) Let 'm' be the integer constant of pumping lemma

(d) Let $z = a^{m^2}$, where length of z is

$|z| = m^2$

(e) By pumping lemma, 'z' may be written as,

$z = uvw$.

Where $1 \le (v) \le m$

and uv^iw is in L for $i \ge 0$

(f) As per assumption that L is regular and $z = uvw$ is in L, it is assumed for $i = 1$ that $uv^iw \subset L$

Let $i = 2$

$1 \le (v) \le n$

$m^2 + 1 \le |uvw| \le m + m^2$

as, "uv^2w" is concatenation of "uvw" and 'v' where $|uvw| = |z| = m^2$ by our assumption. Therefore, we have "m^2" on both the sides of $1 \le |v| \le m$

(g) $m^2 + 1 \le |uv^2w| \le m^2 + m$

but $m^2 + m < (m + 1)^2$

Therefore the equation becomes

$m^2 + 1 \le |uv^2w| < (m+1)^2$

i.e. $m^2 < |uv^2w| < (m + 1)^2$

i.e. for i = 2, the length of the string "uv^iw" resides between m^2 and $(m+1)^2$, two consecutives square values. Thus, the $|uv^2w|$ is not a perfect square. Therefore "uv^2w" is not in L. But this is a contradiction to the lemma statement. Then our assumption that "L" is regular must be wrong. Therefore the given language 'L' is not regular.

Q.29 ***Define the finite automata. Explain difference between deterministic and non-deterministic finite automata. Find deterministic (DFA) for the following non-deterministic finite automation (NDFA)***

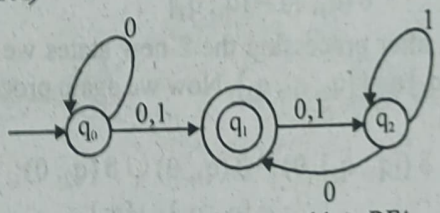


Fig. : NDFA to be converted into DFA

[R.T.U. 2011]

Ans. Finite Automata : *Refer to Q.3.*

Difference between Deterministic and Non-deterministic Finite Automata : *Refer to Q.15.*

Since a state transition diagram is given it must be converted into a state transition table.

States	Input	
	0	1
$\rightarrow q_0$	q_0, q_1	q_1
(q_1)	q_2, q_1	q_2, q_1
q_2	q_1	q_2

Step 1 : Input symbol for the resultant DFA

= Input symbols for the given NDFA

= 0 and 1

Step 2 : Start state for the resultant DFA

= Set of start states of given NDFA

= q_0

Step 3 : Since there are three states in the given NDFA, so the number of states in the resultant DFA = 2^3

All the states of resultant DFA = Set of subsets of the set of states of the given NDFA.

$= \phi, q_0, q_1, q_2, [q_0, q_1], [q_0, q_2], [q_1, q_2], [q_0, q_1, q_2]$

Step 4 : Now we will find out the accessible states from the states derived in step 3 *i.e.* which are accessible from the start state of the DFA which is q_0 in this case.

$$\delta(q_0, 0) = [q_0, q_1]$$
$$\delta(q_0, 1) = [q_1]$$

Taking each of the two input symbols once with the start state, we get two new states *i.e.* $[q_0, q_1]$ and q_1. Now we apply the same procedure for the two new states as under

$$\delta([q_0, q_1], 0) = \delta(q_0, 0) \cup \delta(q_1, 0)$$
$$= \delta[q_0, q_1] \cup [q_2, q_1]$$
$$= [q_0, q_1, q_2]$$
$$\delta([q_0, q_1], 1) = \delta(q_0, 1) \cup \delta(q_1, 1)$$
$$= [q_1] \cup [q_2, q_1]$$
$$= [q_2, q_1]$$
$$\delta(q_1, 0) = [q_1, q_2]$$
$$\delta(q_1, 1) = [q_1, q_2]$$

Thus, after processing the 2 new states we get 2 new states $[q_1, q_2]$ and $[q_0, q_1, q_2]$. Now we again process these 2 new states.

$$\delta([q_1, q_2], 0) = \delta(q_1, 0) \cup \delta(q_2, 0)$$
$$= [q_1, q_2] \cup [q_1]$$
$$= [q_2, q_1]$$
$$\delta([q_1, q_2], 1) = \delta(q_1, 1) \cup \delta(q_2, 1)$$
$$= [q_1, q_2] \cup [q_2]$$
$$= [q_1, q_2]$$
$$\delta([q_0, q_1, q_2], 0) = \delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0)$$
$$= [q_0, q_1] \cup [q_1, q_2] \cup [q_1]$$
$$= [q_0, q_1, q_2]$$
$$\delta([q_0, q_1, q_2], 1) = \delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1)$$
$$= [q_1] \cup [q_1, q_2] \cup [q_2]$$
$$= [q_1, q_2]$$

Thus on processing we do not get any new state and so we simply stop processing now. The accessible states are :

$[q_0], [q_1], [q_0, q_1], [q_1, q_2], [q_0, q_1, q_2]$

Step 5 : Final state for the resultant DFA = Those accessible state that consists of at least one of the final state of the given NDFA. Final state of NDFA = $[q_2]$. From accessible state q_2 containing states are (q_1, q_2) and (q_0, q_1, q_2).

So, final state of the resultant DFA

$= (q_1, q_2)$ and (q_0, q_1, q_2)

Q.30 ***Define regular expressions and languages associated with regular expressions. Write the regular expression and finite automata (transition diagram) for following language over alphabets*** $\sum = \{a, b\}$

(i) The set of strings that start with "ab" and end with "bb".

(ii) The set of strings that starts with 'a' and ends with 'b' and contain at least one sequence of "a a a" in that string.

[RTU 2011]

Ans. Regular Language

The set of regular languages over an alphabet Σ is defined recursively as below. Any language belonging to this set is a regular language over Σ.

Definition of set of Regular Languages

Basic Clause : $\{\Lambda\}$ and $\{a\}$ for any symbol $a \in \Sigma$ are regular languages.

Inductive Clause: If L_r and L_s are regular languages, then $L_r \cup L_s$, $L_r L_s$ and L_r^* are regular languages.

Extremal Clause: Nothing is a regular language unless it is obtained from the above two clauses. For example, let $\Sigma = \{a, b\}$.

Then since $\{a\}$ and $\{b\}$ are regular languages, $\{a,b\}$ $(a=\{a\} \cup \{b\})$ and $\{ab\}(=\{a\}\{b\})$ are regular languages. Also since $\{a\}$ is regular, $\{a\}^*$ is a regular language which is the set of strings consisting of a's such a Λ, a, aa, aaa, aaaa etc. Note also that Σ^*, which is the set of strings consisting of a's and b's, is a regular language because $\{a,b\}$ is regular.

Regular Expression

Regular expressions are used to denote regular languages. They can represent regular languages and operations on them succinctly.

The set of regular expressions over an alphabet Σ is defined recursively as below. Any element of that set is a regular expression.

Basic Clause : ϕ, Λ and a are regular expressions corresponding to languages ϕ, $\{\Lambda\}$ and $\{a\}$, respectively, where a is an element of Σ.

Inductive Clauses : If r and s are regular expressions corresponding to language L_r and L_s, then $(r+s)$, (rs) and (r^*) are regular expressions corresponding to languages $L_r \cup L_s$, $L_r L_s$ and L_r^*, respectively.

Extremal Clause : Nothing is a regular expression unless it is obtained from the above two clauses.

Conventions on regular expressions

(1) When there is no danger of confusion, bold face may not be used for regular expression. So for example, (r + s) is used instead of (r + s).

(2) The operation * has precedence over concatenation, which has precedence over union (+). Thus the regular expression (a+(b(c*))) is written as a + bc*,

(3) The concatenation of k r's, where r is a regular expression, is written as r^k. Thus for example $rr = r^2$. The language corresponding to r^k is L_r^k, where L_r is the language corresponding to the regular expression r. For a recursive definition of L_r^k

(4) We use (r^+) as a regular expression to represent L_r^+.

Examples of regular expression and regular languages corresponding to them

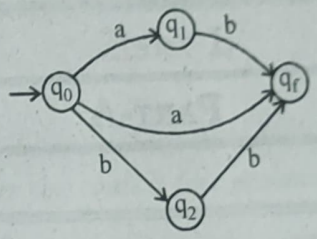
- $(a+b)^2$ corresponds to the language {aa, ab, ba, bb}. That is the set of strings of length 2 over the alphabet {a,b}. In general $(a+b)^k$ corresponds to the set of strings of length k over the alphabet {a,b}. (a+b)* corresponds to the set of all strings over the alphabet {a,b}.
- a^*b^* corresponds to the set of strings consisting of zero or more a's followed by zero or more b's.
- $a^*b^+a^*$ corresponds to the set of strings consisting of zero or more a's followed by one or more b's followed by zero or more a's.
- $(ab)^+$ corresponds to the language {ab, abab, ababab, ...}, that is, the set of strings of repeated ab's.

Note: A regular expression is not unique for a language. That is, a regular language, in general, corresponds to more than one regular expressions. For example (a + b)* and (a*b*)* correspond to the set of all strings over the alphabet {a,b}.

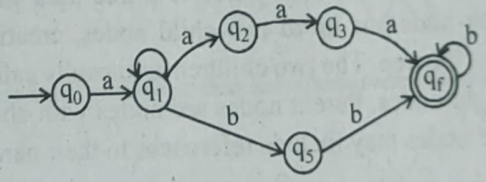
Definition of Equality of Regular Expressions

Regular expressions are **equal** if and only if they correspond to the same language. Thus for example $(a + b)^* = (a^*b^*)^*$, because they both represent the language of all strings over the alphabet {a,b}. **In general, it is not easy to see by inspection whether or not two regular expressions are equal.**

(i) Regular expression is ab + (a + bb), corresponding transition diagram for this expression is



(ii) Regular expression is a(a+b)* (aaa+bb)(b)*, transition diagram is a, b



❑❑❑