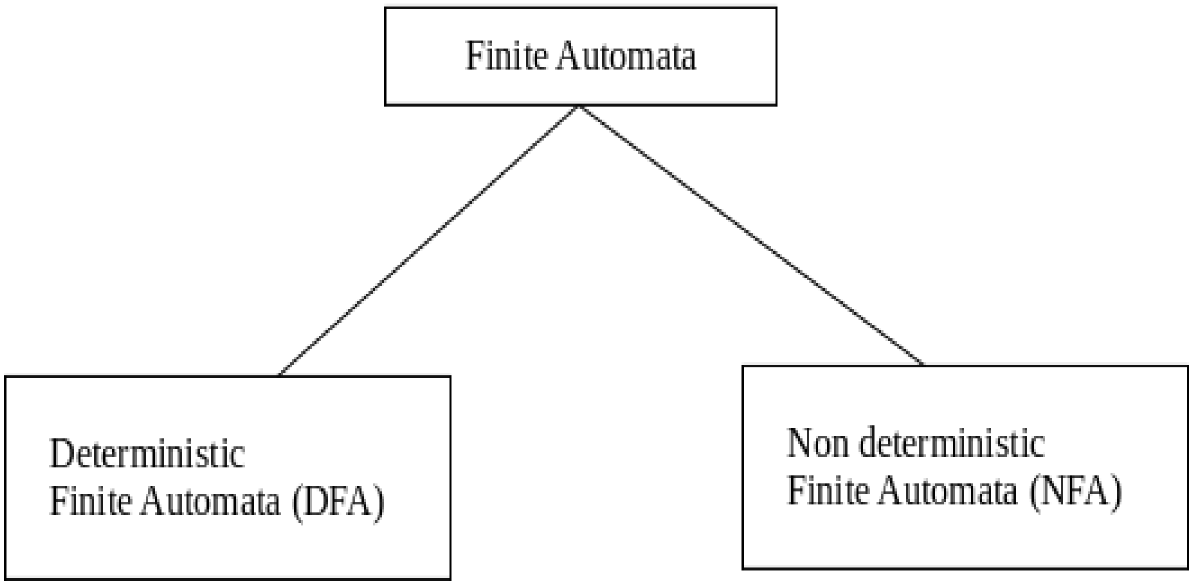
FINITE AUTOMATA

Types of Automata

- There are two types of finite automata:

1. DFA(deterministic finite automata)
2. NFA(non-deterministic finite automata)



Deterministic Finite Automata(DFA)

- The finite automata are called deterministic finite automata if the m/c is read an i/p string one symbol at a time.
- Deterministic refers to the uniqueness of the computation.
- In the DFA, there is only one path for specific i/p from the current state to the next state.
- DFA does not accept the null move i.e. DFA cannot change state without any i/p character.

- DFA can contain multiple final states.
- It is used in lexical Analysis in compiler.

- Acceptance of languages: Reaching to final state
- DFA to accept zero or more “a”
- L={a,aa,aaa, aaaa,..........}

LOVELY PROFESSIONAL UNIVERSITY

Formal Definition of DFA

- A DFA can be represented by a 5-tuple (Q, $\sum$, δ, q_0, F) where –
- **Q** is a finite set of states.
- **$\sum$** is a finite set of symbols called the alphabet.
- **δ** is the transition function where $\delta: Q \times \sum \to Q$
- **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 ($F \subseteq Q$).

LOVELY PROFESSIONAL UNIVERSITY

Graphical Representation of a DFA

- A DFA is represented by digraphs called **state diagram**.
- The vertices represent the states.
- The arcs labeled with an input alphabet show the transitions.
- The initial state is denoted by an empty single incoming arc.
- The final state is indicated by double circles.

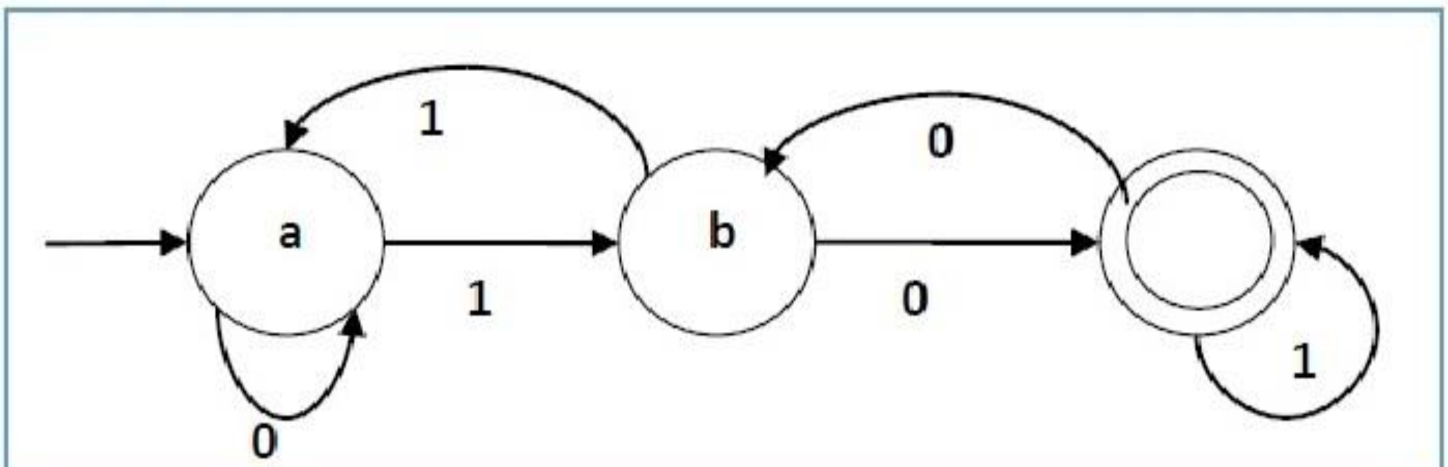
Example

- Let a deterministic finite automaton be →
- $Q = \{a, b, c\}$,
- $\sum = \{0, 1\}$,
- $q_0 = \{a\}$,
- $F = \{c\}$, and

- No of States = | Min String |+1

Present State	Next State for Input 0	Next State for Input 1
a	a	b
b	c	a
c	b	c

Its graphical representation would be as follows –



Practice Questions on DFA

1. Construct DFA accepting all strings over {0,1} that starts with 1 and ends with 0.

2. Construct DFA accepting all strings over {a,b} ending with ‘ab’.

- 3.Construct DFA accepting all strings over {a,b} ending with ‘abb’.

Non Deterministic Finite Automata(NDFA)

- The FA are called NFA, when there exist many paths for specific i/p from the current state to the next state.

- It is easy to construct NFA than DFA for a given regular language.

- Every NFA is not DFA, but each NFA can be translated into DFA.

- NFA is defined in the same way as DFA but with two exceptions:

  1. It contain multiple next state

  2. It contains ∈ transitions.

Formal Definition of an NDFA

- An NDFA can be represented by a 5-tuple (Q, ∑, δ, q_0, F) where –
- **Q** is a finite set of states.
- **∑** is a finite set of symbols called the alphabets.
- **δ** is the transition function where $\delta: Q \times \sum \rightarrow 2^Q$
- (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 ($F \subseteq Q$).

Graphical Representation of an NDFA: (same as DFA)

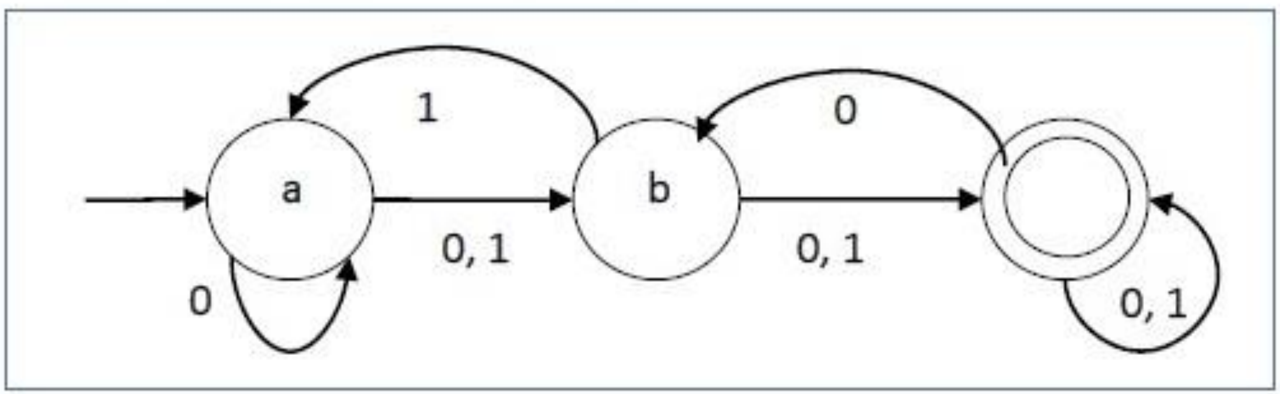
- An NDFA is represented by digraphs called **state diagram.**
- The vertices represent the states.
- The arcs labeled with an input alphabet show the transitions.
- The initial state is denoted by an empty single incoming arc.
- The final state is indicated by double circles.

Example

- Let a non-deterministic finite automaton be →
- Q = {a, b, c}
- ∑ = {0, 1}
- q_0 = {a}
- F = {c}

Present State	Next State for Input 0	Next State for Input 1
a	a, b	b
b	c	a, c
c	b, c	c

Its graphical representation would be as follows −



DFA vs NDFA

DFA	NDFA
The transition from a state is to a single particular next state for each input symbol. Hence it is called *deterministic*.	The transition from a state can be to multiple next states for each input symbol. Hence it is called *non-deterministic*.
Empty string transitions are not seen in DFA.	NDFA permits empty string transitions.
Backtracking is allowed in DFA	In NDFA, backtracking is not always possible.
Requires more space.	Requires less space.
A string is accepted by a DFA, if it transits to a final state.	A string is accepted by a NDFA, if at least one of all possible transitions ends in a final state.

- Practice Questions on NFA

Question 1: Design NFA for given transition table.

State	0	1
-> q0	q0, q1	q0, q2
q1	q3	∈
q2	q2, q3	q3
* q3	q3	q3

Question 2: Construct NFA , L={Set of all strings that starts with 0} with ∑ = {0, 1}.

Question 3: Construct NFA with ∑ = {0, 1} that accepts all string with 01.

Question 4: Construct NFA with ∑ = {0, 1} that accepts all string of length atleast 2.

Question 5: Design an NFA with ∑ = {0, 1} that accepts all strings ending with 01.

Question 6: Construct NFA with ∑ = {0, 1} in which double ‘1’ is followed by double ‘0’.

Question 7: Construct NFA in which all the string contain a substring 1110.

Question 8: Design an NFA with ∑ = {0, 1} that accepts all string in which the third symbol from the right end is always 0.

Equivalence of DFA and NFA: Conversion from NFA to DFA

- In NFA, when a specific input is given to the current state, the machine goes to multiple states.

- It can have zero, one or more than one move on a given input symbol.

- On the other hand, in DFA, when a specific input is given to the current state, the machine goes to only one state.
-
- DFA has only one move on a given input symbol.

- Let, $M = (Q, \sum, \delta, q0, F)$ is an NFA which accepts the language L(M).

- There should be equivalent DFA denoted by $M' = (Q', \sum', q0', \delta', F')$ such that $L(M) = L(M')$.

Steps for converting NFA to DFA:

- **Step 1:** Initially Q' = φ
- **Step 2:** Add q0 of NFA to Q'. Then find the transitions from this start state.
- **Step 3:** In Q', find the possible set of states for each input symbol. If this set of states is not in Q', then add it to Q'.
- **Step 4:** In DFA, the final state will be all the states which contain F(final states of NFA)

Finite automata:
Mealy and Moore Machines

- Finite automata may have outputs corresponding to each transition.
- There are two types of finite state machines that generate output –
- Mealy Machine
- Moore machine

Mealy Machine

- A Mealy Machine is an FSM whose output depends on the present state as well as the present input.

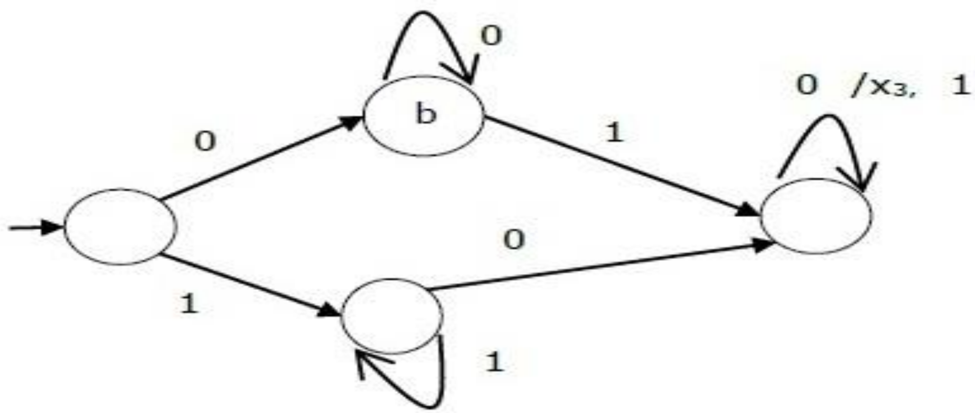
- It can be described by a 6 tuple $(Q, \sum, O, \delta, X, q_0)$ where –
- **Q** is a finite set of states.
- **∑** is a finite set of symbols called the input alphabet.
- **O** is a finite set of symbols called the output alphabet.
- **δ** is the input transition function where $\delta: Q \times \sum \rightarrow Q$
- **X** is the output transition function where $X: Q \times \sum \rightarrow O$
- $\mathbf{q_0}$ is the initial state from where any input is processed ($q_0 \in Q$).

Example

The state table of a Mealy Machine is shown below –

Present state	Next state			
	input = 0		input = 1	
	State	Output	State	Output
→ a	b	x_1	c	x_1
b	b	x_2	d	x_3
c	d	x_3	c	x_1
d	d	x_3	d	x_2

The state diagram of the above Mealy Machine is –



Practice Questions

1. Construct Mealy m/c that produce 1's complement of any binary i/p string.

2. Construct a mealy m/c that points 'a' whenever the sequence '01' is encountered in any i/p binary string.

3. Construct a mealy m/c accepting language consisting of strings where {a,b} and the string should end with either aa or bb.

Moore Machine

- Moore machine is an FSM whose outputs depend on only the present state.

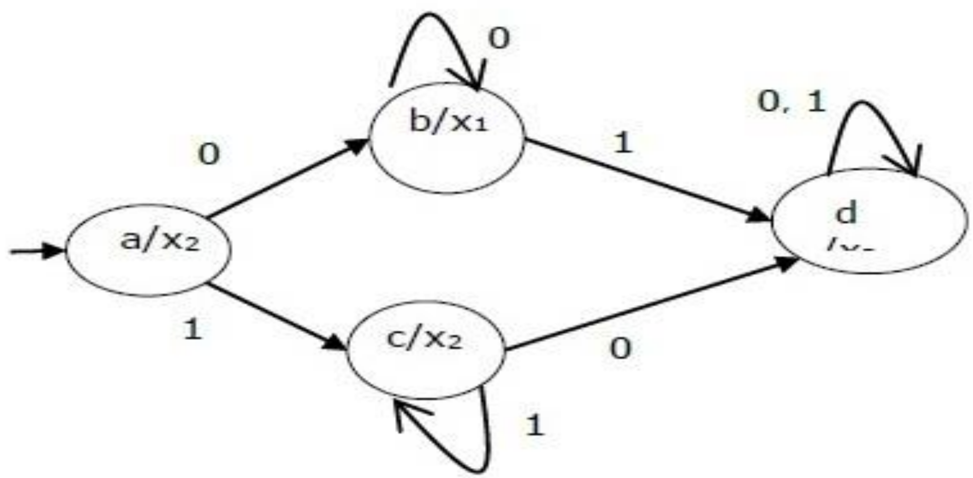
- A Moore machine can be described by a 6 tuple (Q, ∑, O, δ, X, q_0) where –
- **Q** is a finite set of states.
- **∑** is a finite set of symbols called the input alphabet.
- **O** is a finite set of symbols called the output alphabet.
- **δ** is the input transition function where $\delta: Q \times \Sigma \rightarrow Q$
- **X** is the output transition function where $X: Q \rightarrow O$
- $\mathbf{q_0}$ is the initial state from where any input is processed ($q_0 \in Q$).

Example

The state table of a Moore Machine is shown below –

Present state	Next State		Output
	Input = 0	Input = 1	
→ a	b	c	x_2
b	b	d	x_1
c	c	d	x_2
d	d	d	x_3

The state diagram of the above Moore Machine is –



Practice Questions

1. Construct Moore m/c that produce 1's complement of any binary i/p string.

2. Design a moore m/c for a binary i/p sequence such that if it has a substring 101, the m/c o/p is A, if the i/p has substring 110, its o/p is B, otherwise o/p is C.

3. Design moore m/c, where i/p alphabet is $\sum=\{a,b\}$, and the o/p alphabet is O={0,1}. Run the following i/p sequence and find the respective o/p:

(i) aabab (ii) abbb (iii) ababb

State transition table is given as:

States	a	b	o/p
Q0	Q1	Q2	0
Q1	Q2	Q3	0
Q2	Q3	Q4	1
Q3	Q4	Q4	0
Q4	Q0	Q0	0

Conversion from Mealy machine to Moore Machine

- In Moore machine, the output is associated with every state, and in Mealy machine, the output is given along the edge with input symbol.

- To convert Moore machine to Mealy machine, state output symbols are distributed to input symbol paths.

- But while converting the Mealy machine to Moore machine, we will create a separate state for every new output symbol and according to incoming and outgoing edges are distributed.

- **Step 1:** For each state(Qi), calculate the number of different outputs that are available in the transition table of the Mealy machine.

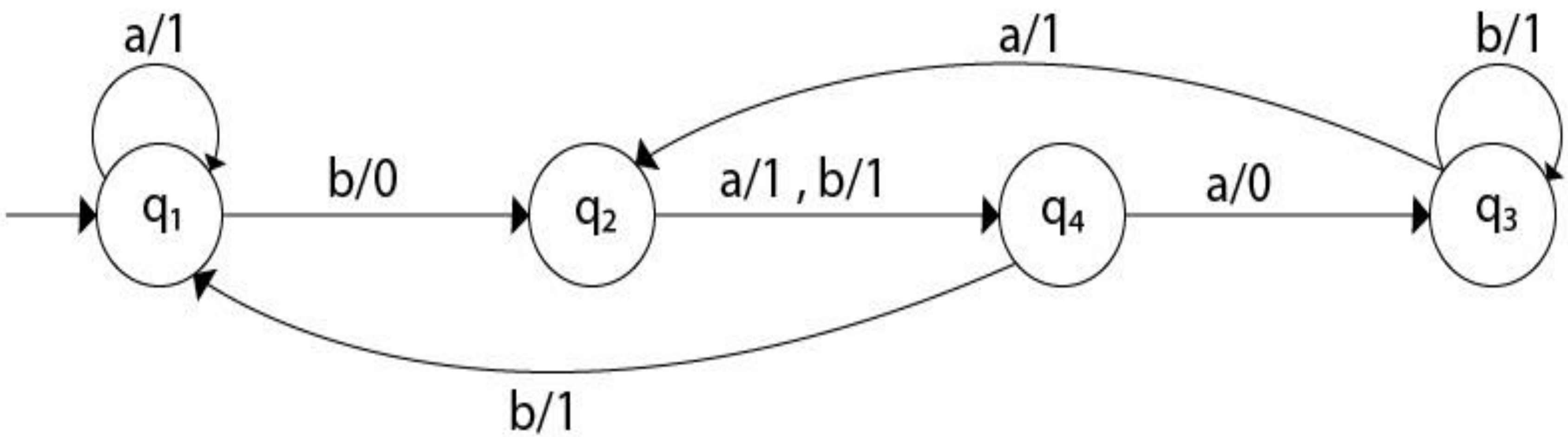
- **Step 2:** Copy state Qi, if all the outputs of Qi are the same. Break qi into n states as Qin, if it has n distinct outputs where n = 0, 1, 2....

- **Step 3:** If the output of initial state is 0, insert a new initial state at the starting which gives 1 output.

Practice Questions

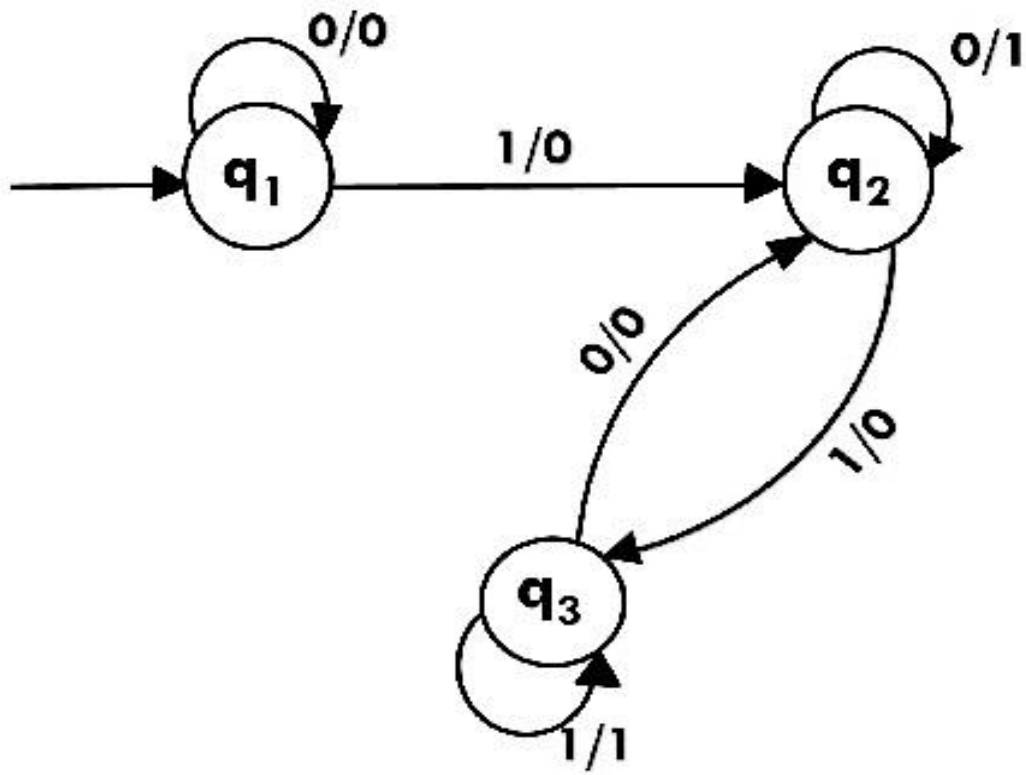
Que:1

Convert the following Mealy machine into equivalent Moore machine.



- Ques:2

Convert the following Mealy machine into equivalent Moore machine.



Conversion from Moore machine to Mealy Machine

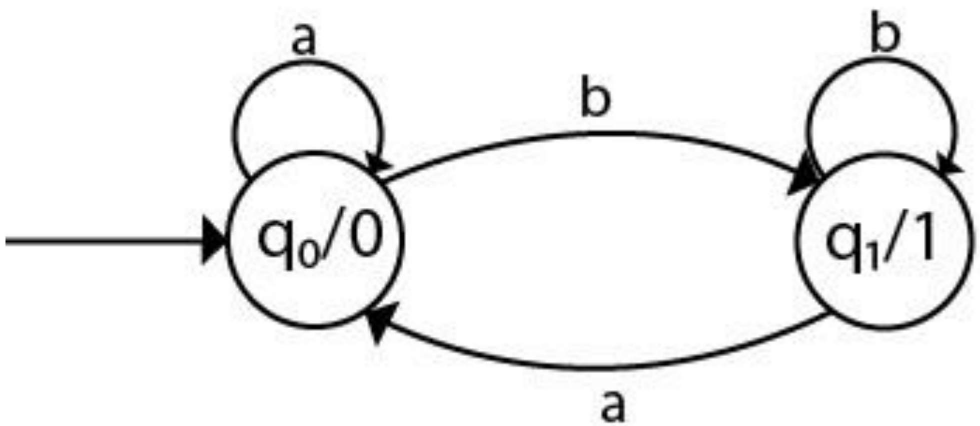
- We cannot directly convert Moore machine to its equivalent Mealy machine because the length of the Moore machine is one longer than the Mealy machine for the given input.

- To convert Moore machine to Mealy machine, state output symbols are distributed into input symbol paths.

Practice Questions

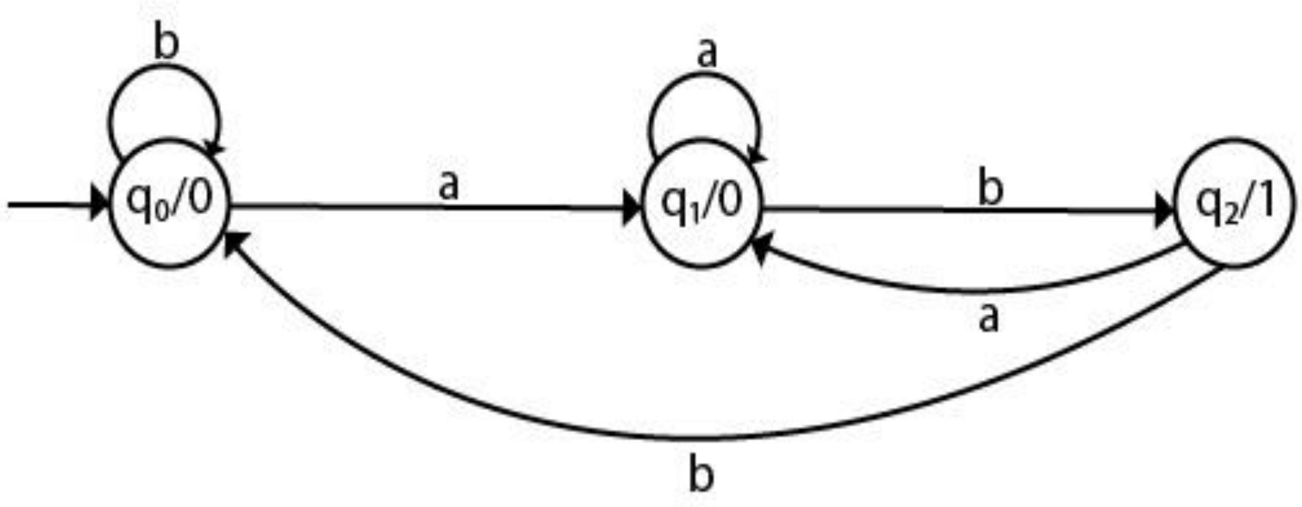
- Ques:1

Convert the following Moore machine into its equivalent Mealy machine.



- Ques:2

Convert the given Moore machine into its equivalent Mealy machine.



- Ques:3

Convert the given Moore machine into its equivalent Mealy machine.

Q	a	b	Output(λ)
q0	q0	q1	0
q1	q2	q0	1
q2	q1	q2	2

- Ques:4 The moore m/c counts the occurrences of sequence ‘abb’ in any i/p binary strings over {a,b}. Convert it into its equivalent mealy m/c.

Minimization of DFA Equivalence Theorem

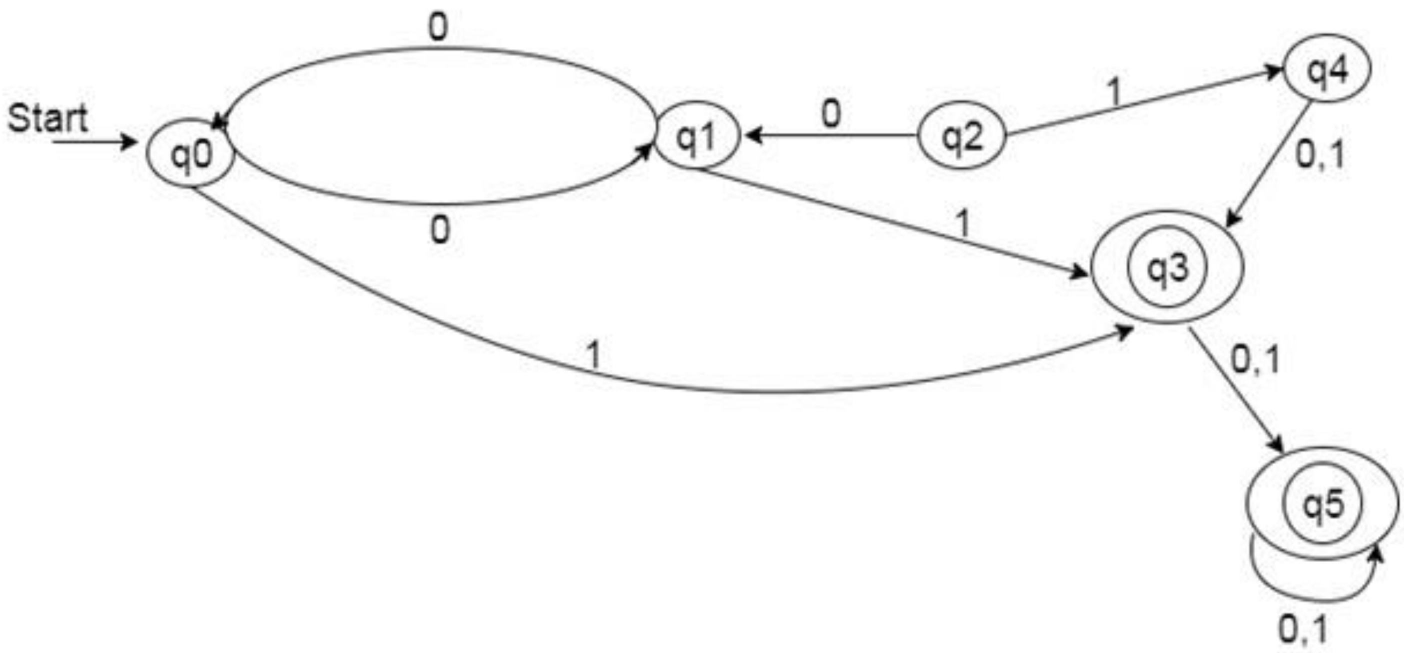
- Minimization of DFA means reducing the number of states from given FA.

- **Step 1:** Remove all the states that are unreachable from the initial state via any set of the transition of DFA.
- **Step 2:** Draw the transition table for all pair of states.
- **Step 3:** Now split the transition table into two tables T1 and T2. T1 contains all final states, and T2 contains non-final states.

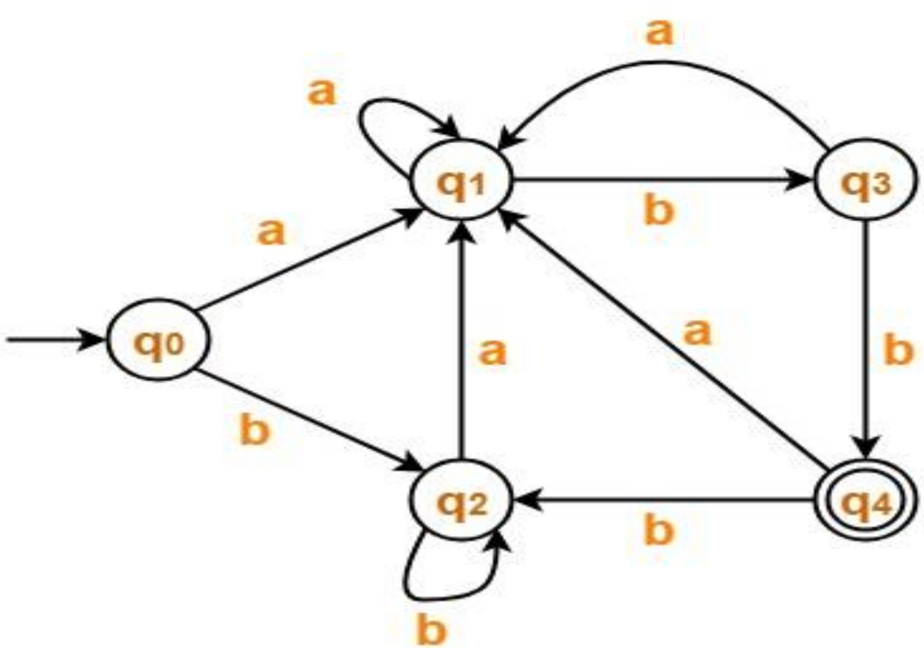
- **Step 4:** Find similar rows from T1 such that:
    1. $\delta(q, a) = p$
    2. $\delta(r, a) = p$
- That means, find the two states which have the same value of a and b and remove one of them.
- **Step 5:** Repeat step 3 until we find no similar rows available in the transition table T1.
- **Step 6:** Repeat step 3 and step 4 for table T2 also.
- **Step 7:** Now combine the reduced T1 and T2 tables. The combined transition table is the transition table of minimized DFA.

Practice Questions

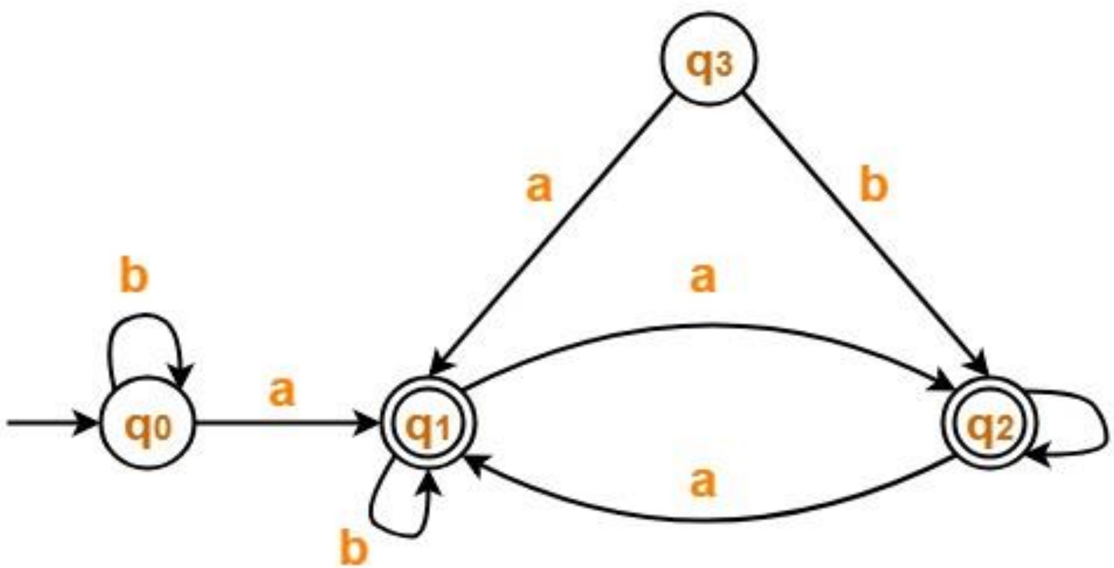
- Ques:1 Minimize the given DFA:



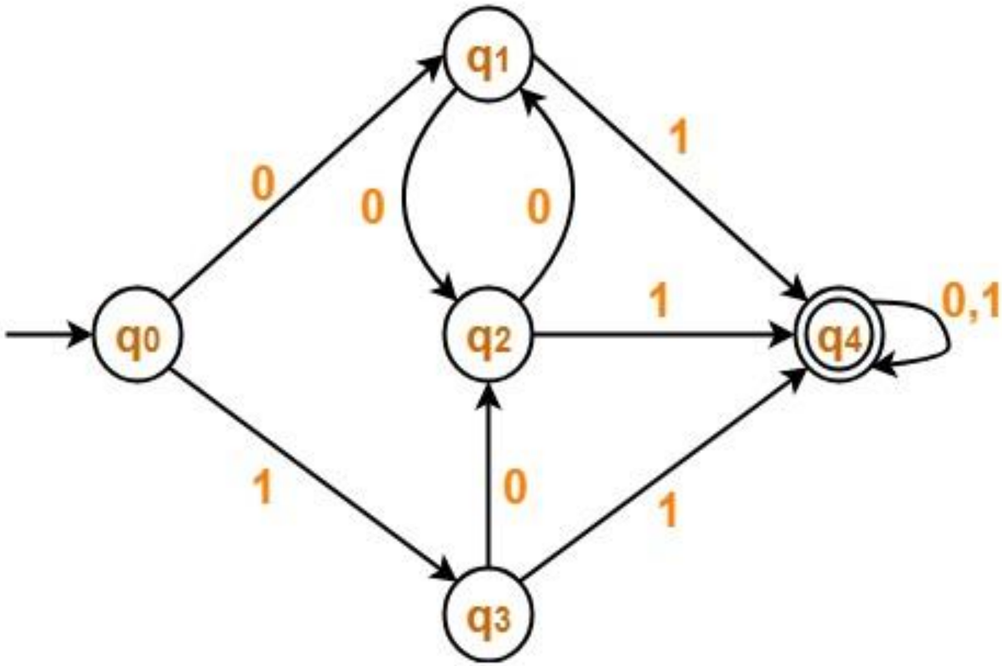
- Ques:2 Minimize the given DFA:



- Ques:3 Minimize the given DFA:



- Ques:4 Minimize the given DFA:



Minimization of DFA
Table Filling Method
(MyHill Nerode Theorem)

- Suppose there is a DFA D < Q, Σ, q0, δ, F > which recognizes a language L. Then the minimized DFA D < Q', Σ, q0, δ', F' > can be constructed for language L as:

  Step 1: We will divide Q (set of states) into two sets. One set will contain all final states and other set will contain non-final states. This partition is called P_0.

  Step 2: Initialize k = 1

- **Step 3:** Find P_k by partitioning the different sets of P_{k-1}. In each set of P_{k-1}, we will take all possible pair of states. If two states of a set are distinguishable, we will split the sets into different sets in P_k.

Step 4: Stop when $P_k = P_{k-1}$ (No change in partition)

Step 5: All states of one set are merged into one. No. of states in minimized DFA will be equal to no. of sets in P_k.

- **How to find whether two states in partition P_k are distinguishable ?**

Two states (qi, qj) are distinguishable in partition P_k if for any input symbol a, δ (qi, a) and δ (qj, a) are in different sets in partition P_{k-1}.

POLLING QUESTIONS

1. Number of states require to accept string ends with 10.

   A) 3
   B) 2
   C) 1
   D) can’t be represented.