Lecture Notes
on
Theory of Automata and Formal Languages

Unit-1

Dharmendra Kumar
(Associate Professor)
United College of Engineering and Research, Prayagraj

May 10, 2023

Contents

Chapter 1

Basic Concepts

1.1 Alphabet

A finite set of symbols is said to be alphabet. We will denote it by set Σ .

1.2 String

This is the sequence of symbols from alphabet.

Example: If $\Sigma = \{$a,b$\}$, then abab, aaabab are strings on Σ.

1.2.1 Operations defined on strings

Concatenation

The concatenation of two strings w and v is the string obtained by appending the symbols of v to the right end of w, that is, if
w = $a_1a_2.........a_n$
and v = $b_1b_2........b_m$
Then concatenation of w and v, denoted by wv, is
wv = $a_1a_2........a_nb_1b_2........b_m$

Reverse of a string

The reverse of a string is obtained by writing the symbols in reverse order. If w is a string then its reerse is denoted by w^R.

Example: If w = $a_1a_2.........a_n$ then
$w^R = a_na_{n-1}.........a_2a_1$

Length of a string

The length of a string is the number of symbols in the strings. If w is a string then it is denoted by $\mid w \mid$.
The string with length 0 is said to be empty string. And it is denoted by ϵ. It is also said to be null string.

1.2.2 Properties of empty string(ϵ)

1. $\mid \epsilon \mid = 0$
2. ϵw = w = wϵ, $\forall$ string w.

1.2.3 Substring

Any string of consecutive characters in string w is said to be a substring of w. If
w = vu
Then the substrings v of u are said to be prefix and suffix of w respectively.

1.2.4 Properties of strings

1. $\mid uv \mid = \mid u \mid + \mid v \mid$
2. w^n = wwwwwww............w(upto n times)
3. $w^0 = \epsilon$, $\forall w$

1.2.5 Kleene Closure(*-closure)

If Σ is an alphabet, then Σ^* denote the kleene closure of Σ.
Σ^* is the set of all the strings obtained by concatenating zero or more symbols from Σ.
$\Sigma^* = \bigcup\limits_{i=0}^{\infty} \Sigma^i = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup$ ....

1.2.6 Positive Closure(+-closure)

If Σ is an alphabet, then Σ^+ denote the kleene closure of Σ.
Σ^+ is the set of all the strings obtained by concatenating one or more symbols from Σ.
$\Sigma^+ = \bigcup\limits_{i=1}^{\infty} \Sigma^i = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup$ ....
$= \Sigma^*$ - $\{\epsilon\}$

Example: Consider $\Sigma = \{$a, b$\}$. Find Σ^2 and Σ^3.

Chapter 2

Language and Grammar

2.1 Grammar

2.1.1 Definition

A grammar G is defined as a quadruple G=(V,Σ,S,P), where

V $\rightarrow$ Finite set of variables or non-terminal symbols
Σ $\rightarrow$ Finite set of terminal symbols
S $\in$ V $\rightarrow$ Starting symbols of the Grammar
P $\rightarrow$ Finite set of production rules

And P is defined as following:-
P $\subseteq (V \cup \Sigma)^* X (V \cup \Sigma)^*$
(u,v)$\in$ P is denoted by
u $\rightarrow$ v, where u,v $\in (V \cup \Sigma)^*$
u always contains at least one variable.

2.1.2 Direct derivation or derivation in one step

Let $\alpha, \beta \in (V \cup \Sigma)^*$. α directly derives β if $\alpha \rightarrow \beta \in P$. It is denoted by $\alpha \Rightarrow \beta$.

2.1.3 Derivation in many steps

Let $\alpha, \beta \in (V \cup \Sigma)^*$. α derives β if there exists production rules $\alpha \rightarrow A_1$, $A_1 \rightarrow A_2$,$A_2 \rightarrow A_3$,.............$A_n \rightarrow \beta$, such that
$\alpha \Rightarrow A_1 \Rightarrow A_2 \Rightarrow A_3 \Rightarrow$ .... $\Rightarrow A_n \Rightarrow \beta$. It is denoted by $\alpha \overset{*}{\Rightarrow} \beta$.

2.1.4 Sentential Form

Let $\alpha \in (V \cup \Sigma)^*$. The string α is said to be in the sentential form if
$S \overset{*}{\Rightarrow} \alpha$, where S is the starting symbol.

2.1.5 Language generated by a Grammar

The set of all the sentental forms consisting of only terminal symbols is said to be language generated Grammar. That is, L(G) = { x$\in \Sigma^*$! $S \overset{*}{\Rightarrow} x$ }

2.1.6 Equivalent Grammars

Two grammars G_1 and G_2 are said to be equivalent grammar if the languages generated by both grammars are the same. That is,

L(G_1) = L(G_2).

2.2 Examples

1. Determine the languages generated by the following grammars:-
   (a) $S \to aS/a$
   (b) $S \to 0S1/\epsilon$
   (c) $S \to aCa$, $C \to aCa/b$
   (d) $S \to aS/bS/a/b$
   (e) $S \to 0SA2$, $S \to 012$, $2A \to A2$, $1A \to 11$
2. Determine the languages generated by the following grammars:-
   (a) $S \to 0S1/0A1$, $A \to 1A/1$
   (b) $S \to aA$, $A \to bS$, $S \to \epsilon$
   (c) $S \to 0S1/0A/0/1B/1$, $A \to 0A/0$, $B \to 1B/1$
   (d) $S \to 0SBA/01A$, $AB \to BA$, $1B \to 11$, $1A \to 10$, $0A \to 00$
   (e) $S \to 0S1/0A1$, $A \to 1A0/10$
   (f) $S \to 0A/1S/0/1$, $A \to 1A/1S/1$
3. Construct the grammars which generates the following languages:-
   (a) L(G) = { $a^n ba^m$! m,n $\geq$ 1}
   (b) L(G) = The set of all palindromes over {a,b}
   (c) L(G) = { wcw^T ! w $\in \{a, b\}^*$ }
   (d) L(G) = { $a^n b^n c^i$! n $\geq$ 1 and i$\geq$ 0}
   (e) L(G) = { $a^j b^n c^n$! n $\geq$ 1 and j$\geq$ 0}
   (f) L(G) = { $a^n b^n c^n$! n $\geq$ 1}
4. Construct the grammars which generates the following languages:-
   (a) L(G) = { w $\in \{a, b\}^*$! The number of a's in w is divisible by 3}
   (b) L(G) = { w $\in \{a, b\}^*$! w has an equal number of a's and b's}
   (c) L(G) = { w $\in \{a, b\}^*$! $n_a(w) > n_b(w)$}
   (d) L(G) = { w $\in \{a, b\}^*$! $n_a(w) \neq n_b(w)$}
   (e) L(G) = { $0^m 1^n 0^n 1^m$! m,n $\geq$ 1 }
   (f) L(G) = { $0^n 1^{2n}$! n $\geq$ 1}

(g) L(G) = $\{ 0^n1^n ! \ n \geq 1\} \cup \{ 1^m0^m ! \ m \geq 1\}$
(h) L(G) = $\{ 0^n1^m0^n ! \ m,n \geq 1\} \cup \{ 0^n1^m2^m ! \ m,n \geq 1\}$

5. Find grammars for $\Sigma = \{a,b\}$ that generates the sets of
   (a) All strings with exactly one a.
   (b) All strings with at least one a.
   (c) All strings with no more than three a's.
   (d) All strings with at least three a's.
6. Find grammars for the following languages on $\Sigma = \{a\}$
   (a) L = $\{ w ! \mid w \mid \bmod 3 = 0\}$
   (b) L = $\{ w ! \mid w \mid \bmod 3 > 0\}$
   (c) L = $\{ w ! \mid w \mid \bmod 3 \neq \mid w \mid \bmod 2\}$
   (d) L = $\{ w ! \mid w \mid \bmod 3 \geq \mid w \mid \bmod 2\}$
7. Find grammars for the following languages over $\Sigma = \{a,b\}$
   (a) L = $\{ w ! \ n_a(w) = n_b(w)+1\}$
   (b) L = $\{ w ! \ n_a(w) > n_b(w)\}$
   (c) L = $\{ w ! \ n_a(w) = 2n_b(w)\}$
   (d) L = $\{ w ! \mid n_a(w) - n_b(w) \mid = 1\}$

2.3 Chomsky Hierarchy

According to Chomsky's classification, all the grammars are divided into following four categories:-

Type 0 Grammar(Unrestricted Grammar)
If there is no restriction on the production rules, then grammar is said to be type 0 grammar or unrestricted grammar.

Type 1 Grammar(Context Sensitive Grammar)
A grammar is said to be type 1 grammar or context sensitive grammar if every production rules are of the following form:-
$\phi_1 A \phi_2 \to \phi_1 \psi \phi_2,$ where $\phi_1, \phi_2, \psi \in (V \cup \Sigma)^*$ and A $\in$ V.

Type 2 Grammar(Context Free Grammar)
A grammar is said to be type 2 grammar or context free grammar if every production rules are of the following form:-
$A \to \psi,$ where $\psi \in (V \cup \Sigma)^*$ and A $\in$ V.

Type 3 Grammar(Regular Grammar)
A grammar is said to be type 3 grammar or regular grammar if every production rules

are of the following form:-

$A \to aB$ or $A \to a$, where $a \in \Sigma$ and A, B $\in$ V.

The languages generated by the corresponding grammars are said to be unrestricted

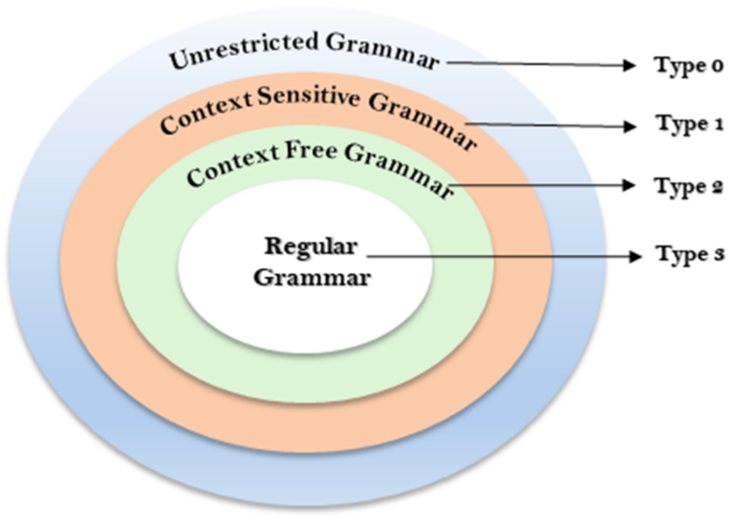
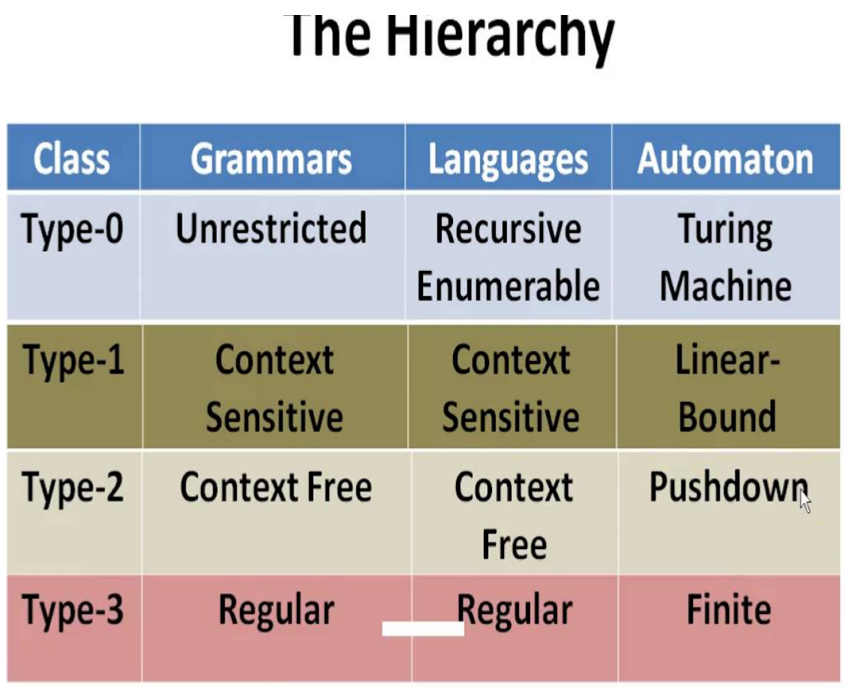


Fig: Chomsky Hierarchy

grammar, context sensitive grammar, context free grammar and regular grammar.



The Hierarchy

Class	Grammars	Languages	Automaton
Type-0	Unrestricted	Recursive Enumerable	Turing Machine
Type-1	Context Sensitive	Context Sensitive	Linear-Bound
Type-2	Context Free	Context Free	Pushdown
Type-3	Regular	Regular	Finite

2.3.1 Exercise

1. Determine the highest type of grammar in the following grammars:-

   (a) $S \to 0S1/0A1$, $A \to 1A/1$

(b) $S \to 0S1/0A/0/1B/1, \qquad A \to 0A/0, \qquad B \to 1B/1$
(c) $S \to 0SBA/01A, \qquad AB \to BA, \qquad 1B \to 11, \qquad 1A \to 10, \qquad 0A \to 00$
(d) $S \to 0S1/0A1, \qquad A \to 1A0/10$
(e) $S \to 1S/0A/0/1, \qquad A \to 1A/1S/1$

2. Construct context free grammars to generate the following languages:-

   (a) L = $\{0^m1^n\ !\ m \neq n\ and\ m, n \geq 1\}$
   (b) L = $\{a^lb^mc^n\ !$ one of l,m,n equals 1 and the remaining two are are equal $\}$
   (c) L = $\{0^m1^n\ !\ 1 \leq m \leq n\}$
   (d) L = $\{a^lb^mc^n\ !\ l + m = n\}$
   (e) The set of all strings over {0,1} containing twice as many 0's as 1's.

3. Construct regular grammars to generate the following languages:-

   (a) L = $\{a^{2n}\ !\ n \geq 1\}$
   (b) The set of all strings over {a,b} ending in a.
   (c) The set of all strings over {a,b} beginning with a.
   (d) L = $\{a^lb^mc^n\ !\ l, m, n \geq 1\}$
   (e) L = $\{(ab)^n\ !\ n \geq 1\}$

2.4 AKTU Examination Questions

1. Construct the CFG for the language L = $\{a^{2n}b^n \mid n \geq 3\}$.

2. Design the CFG for the following language:

   (a) L = $\{0^m1^n \mid m \neq n\ and\ m, n \geq 1\}$
   (b) L = $\{a^lb^mc^n \mid l + m = n\ and\ l, m \geq 1\}$

3. Define alphabet, string and language.

4. Define Chomsky hierarchy.

5. Define and give the difference between positive closure and Kleene closure.

6. Determine the grammar for language L = $\{a^nb^m \mid n \neq m\}$. Also explain the type of this language.

7. Identify the language generated by context free grammar $S \to (S)/SS/(\ )$

8. Describe Chomsky hierarchy of languages with proper example.

Chapter 3

Finite Automata

3.1 Finite Automata

- A finite automata (FA) is the most restricted model of automatic machine. A finite automata is an abstract model of a computer system.
- Finite automata are used to recognize patterns.
- It takes the string of symbol as input and changes its state accordingly. When the desired symbol is found, then the transition occurs.
- At the time of transition, the automata can either move to the next state or stay in the same state.
- Finite automata have two states, Accept state or Reject state. When the input string is processed successfully, and the automata reached its final state, then it will accept.

3.1.1 Finite Automata Model

Finite automata can be represented by input tape and finite control.

Input tape: It is a linear tape having some number of cells. Each input symbol

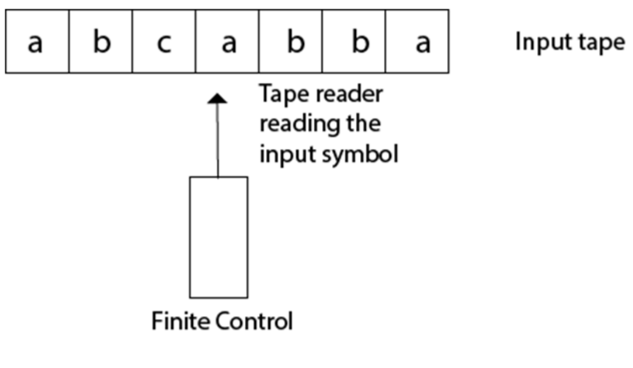


Fig :- Finite automata model

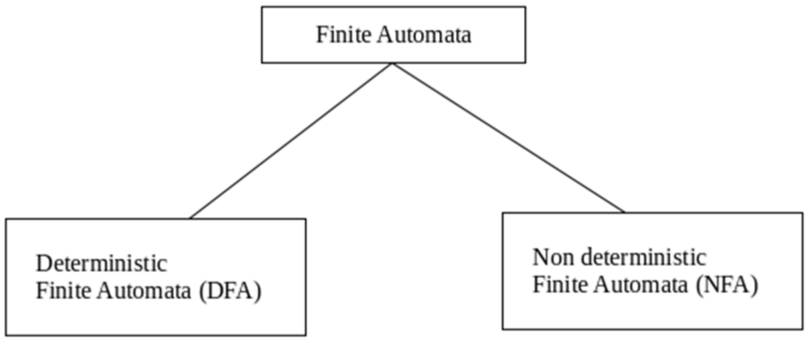
is placed in each cell.

Finite control: The finite control decides the next state on receiving particular input from input tape. The tape reader reads the cells one by one from left to right, and at a time only one input symbol is read.

3.1.2 Types of Finite Automata

There are two types of finite automata:

1. **Deterministic Finite Automata (DFA):** In the DFA, the machine goes to one state only for a particular input character. DFA does not accept the null move.
2. **Non-deterministic Finite Automata (NFA):** In the NFA, the machine goes to one or more state for a particular input character. NFA can accept the null move.


Finite Automata
Deterministic Finite Automata (DFA)
Non deterministic Finite Automata (NFA)

3.1.3 Application of Finite Automata (FA)

We have several applications based on finite automata and finite state machine. Some are given below;

- A finite automata is highly useful to design Lexical Analyzers.
- A finite automata is useful to design text editors.
- A finite automata is highly useful to design spell checkers.
- A finite automata is useful to design sequential circuit design (Transducer).

3.2 Deterministic Finite Automata (DFA)

3.2.1 Definition

A deterministic finite automata M is a 5-tuple, M = (Q, Σ, δ, q_0, F), where

Q $\rightarrow$ Finite set of states
Σ $\rightarrow$ Finite set of input symbols

$q_0 \in Q \rightarrow$ Initial state
F$\subseteq Q \rightarrow$ Set of final states
and $\delta \rightarrow$ Transition function

It is defined as following:-
$$\delta : \text{Q}\text{x}\Sigma \rightarrow \text{Q}$$

Extended Transition Function
It is denoted by $\hat{\delta}$. It is defined as following:-
$$\hat{\delta} : \text{Q}\text{x}\Sigma^* \rightarrow \text{Q}$$

Properties of $\hat{\delta}$

1. $\hat{\delta}(q, \epsilon) =$ q
2. $\hat{\delta}(q, a) = \delta(q, a), \qquad \forall a \in \Sigma$
3. $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a), \qquad \forall a \in \Sigma \; and \; \forall w \in \Sigma^*$

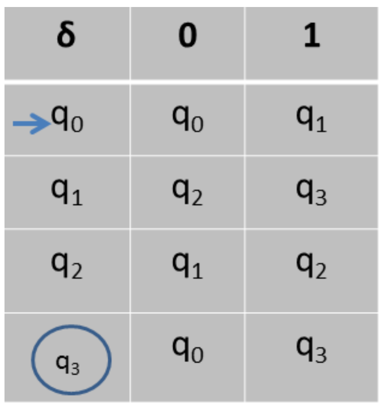
3.2.2 Language Accepted by DFA

Language accepted by DFA M is denoted by L(M). It is defined as following:-
$$\text{L(M)} = \{\text{x} \in \Sigma^* \; ! \; \hat{\delta}(q_0, x) \in F\}$$

3.2.3 Representation of DFA

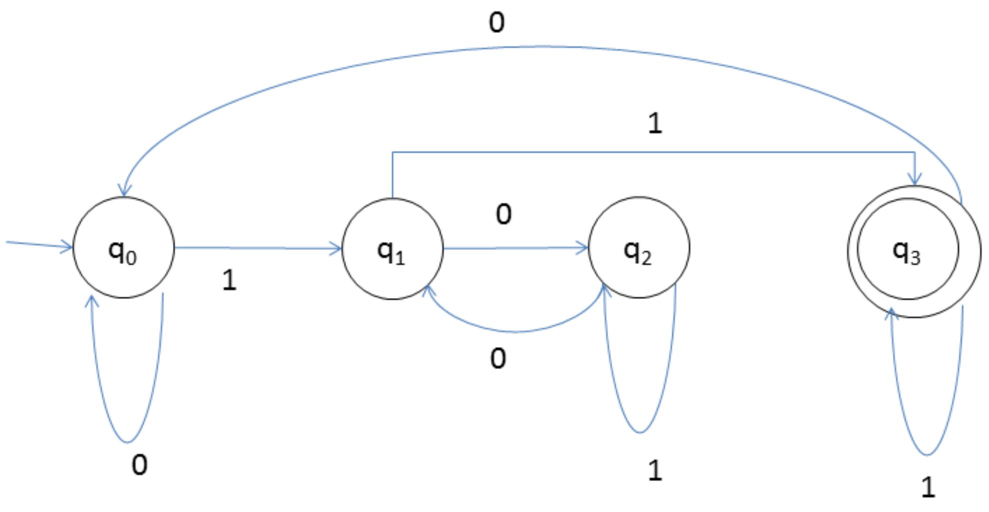
Any DFA can be represented by the following two ways:-
(1) By transition table
(2) By transition diagram

(1)By transition table
.



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

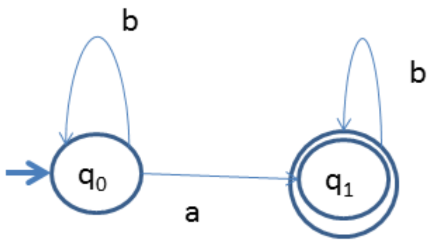
(2) By transition diagram
.



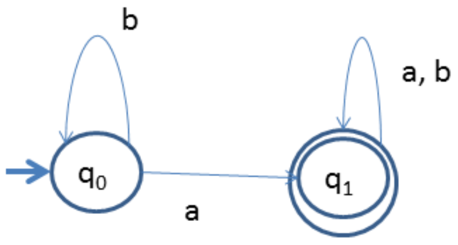
3.2.4 Some Examples

Example: Find the language accepted by following DFA:-

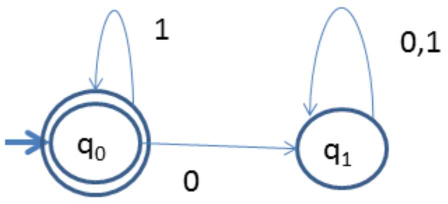
.



Example: Find the language accepted by following DFA:-

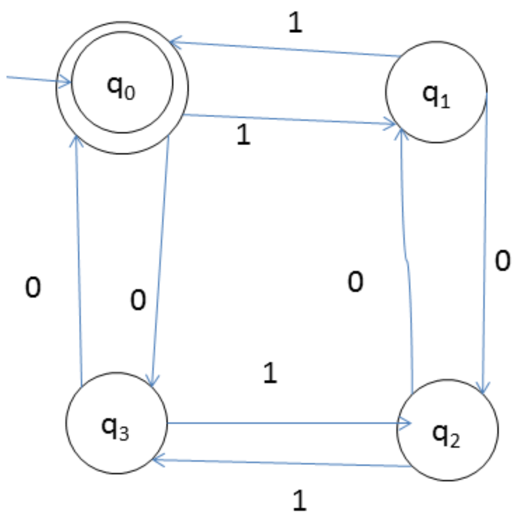


Example: Find the language accepted by following DFA:-

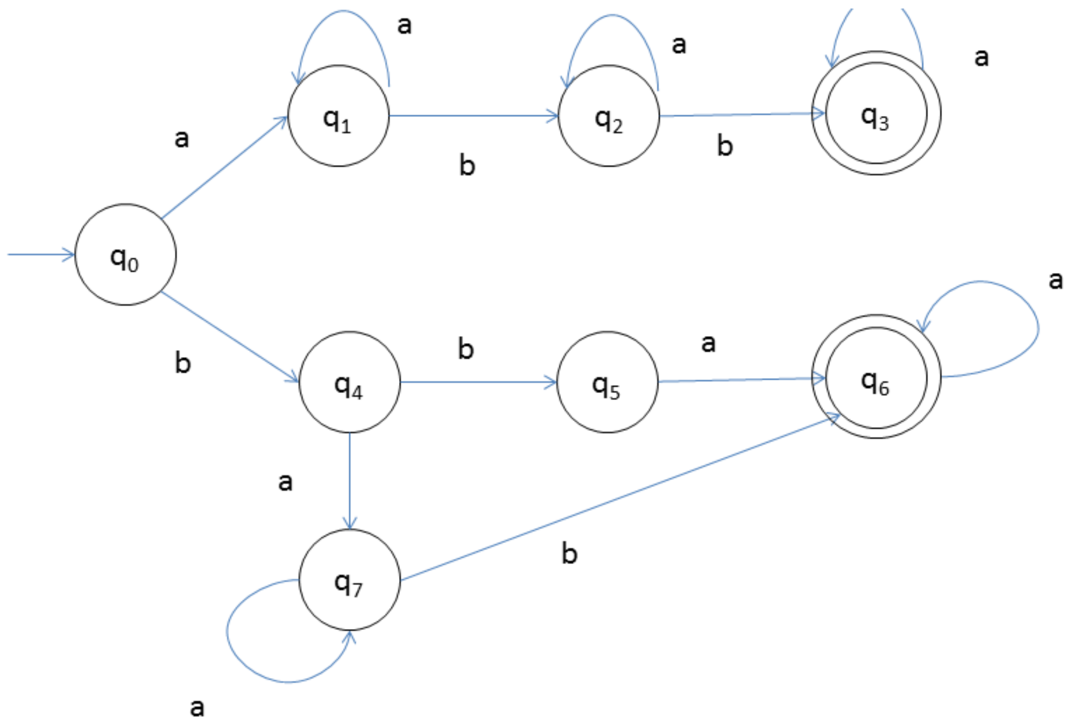


Solution:

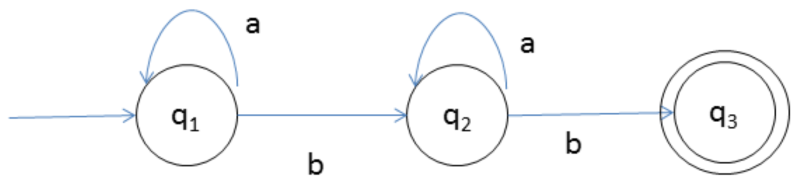
Example: Find the language accepted by following DFA:-



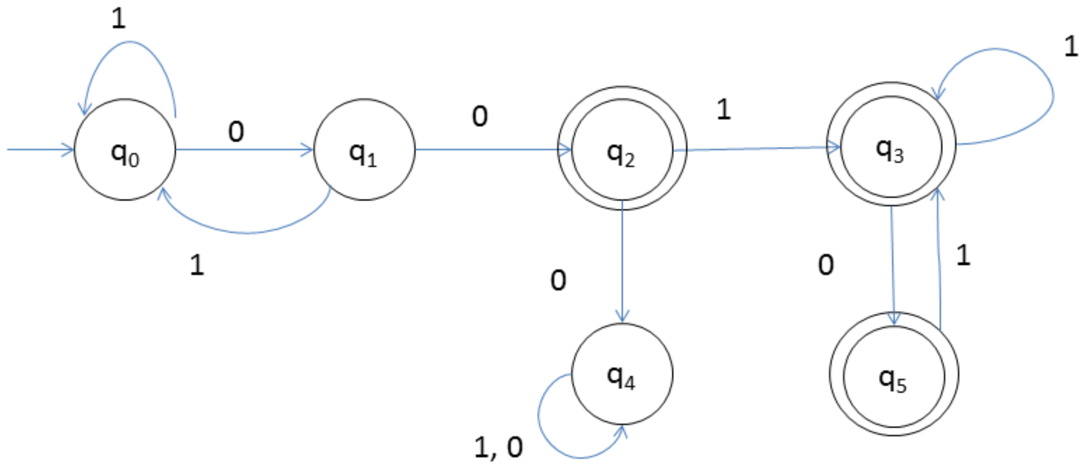
Example: Find the language accepted by following DFA:-



Example: Find the language accepted by following DFAs:-

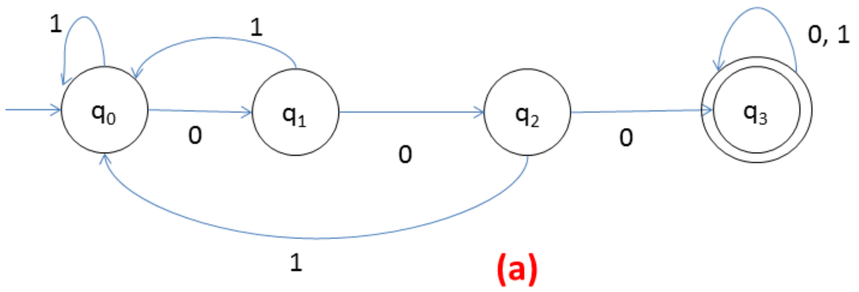


(a)

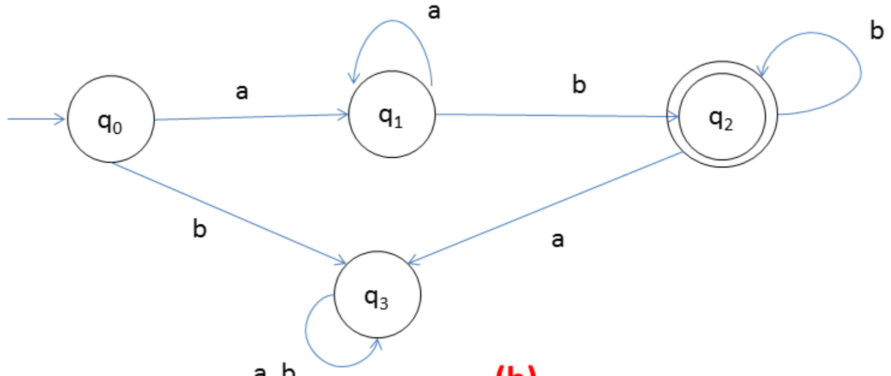


(b)

Example: Find the language accepted by following DFAs:-

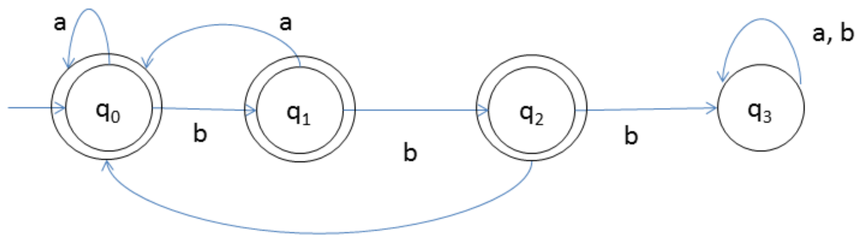


(a)

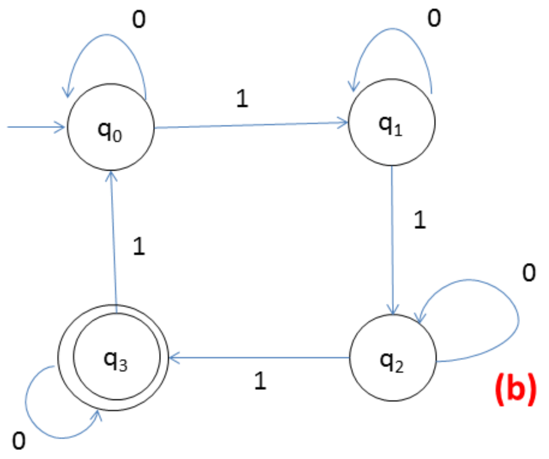


(b)

Example: Find the language accepted by following DFAs:-

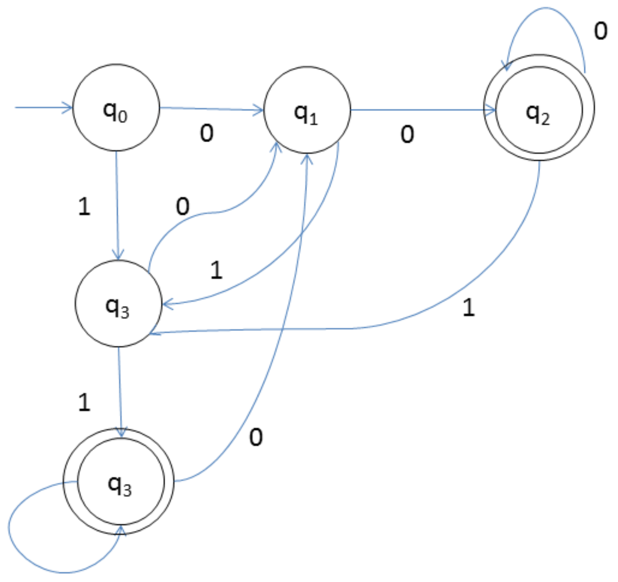


(a)



(b)

Example: Find the language accepted by following DFA:-



Regular language: A language L is said to be regular language iff there exists some deterministic finite automata M such that

L = L(M).

3.2.5 Construction of DFA for a given language

Example: Construct DFA for the following language:-

$L = \{a^m b \,!\, m \geq 0\}$

Solution:

Example: Construct DFA for the following language:-

$L = \{a^m b^n \,!\, m, n \geq 1\}$

Solution:

Example: Construct DFA that recognizes the set of all strings on Σ = {a,b} starting with the prefix ab.

Solution:

Example: Construct DFA that recognizes the set of all strings on Σ = {0,1} except those containing the substring 001.

Solution:

Example: Show that the language

$L = \{awa \,!\, w \in \{a, b\}^*\}$

is regular.

Solution:

Example: If $L = \{awa \,!\, w \in \{a, b\}^*\}$, then show that L^2 is regular. **Solution:**

Note: If L is regular, then L^2, L^3,L^4 .... are also regular.

3.2.6 Exercise

1. For Σ = {a,b}, construct DFA's that accept the sets consisting of
   (a) all strings with exactly one a.
   (b) all strings with at least one a.
   (c) all strings no more than three a's.
   (d) all strings with at least one a and exactly two b's.
   (e) all strings with exactly two a's and more than two b's.
2. Construct DFA for the following sets:-
   (a) The set of all the even binary numbers.
   (b) The set of all the strings of 0 and 1, which start with 1 and ends with 0.

(c) The set of all the strings of 0 and 1 in which number of 1's is odd.
(d) The set of all the binary numbers divisible by 3.
(e) The set of all the positive integers divisible by 3.
(f) The set of all the strings of 0 and 1, which contains even numbers of 0's and 1's.
(g) The set of all the strings of 0 and 1, in which number of 0's is divisible by 5 and number of 1's is divisible by 3.

3. Construct DFA for the following sets:-

   (a) The set of all the strings of 0 and 1 ending with 00.
   (b) The set of all the strings of a and b ending with abb.
   (c) The set of all the strings of 0 and 1 which contains 010 as a substring.
   (d) The set of all the strings of 0 and 1 that contains at least two 0's.
   (e) The set of all the strings of 0 and 1 that contains at most two 0's.
   (f) The set of all the strings of 0 and 1 that have length at most five.
   (g) The set of all the strings of a and b that begins and ends with different letters.
   (h) The set of all the strings of a and b that contains at least two consecutive a's and not contain consecutive b's.

4. For $\Sigma = \{a,b\}$, construct DFA's that accept the sets consisting of

   (a) L $=\{$w ! $\mid w \mid$ mod $3 = 0\}$
   (b) L $=\{$w ! $\mid w \mid$ mod $5 \neq 0\}$
   (c) L $=\{$w ! $n_a(w)$ mod $3 > 1\}$
   (d) L $=\{$w ! $n_a(w)$ mod $3 > n_b(w)$ mod $3\}$
   (e) L $=\{$w ! $(n_a(w) - n_b(w))$ mod $3 > 0\}$
   (f) L $=\{$w ! $\mid n_a(w) - n_b(w) \mid$ mod $3 < 2\}$

3.3 Non-deterministic Finite Automata (NFA)

3.3.1 Definition

A non-deterministic finite automata M is a 5-tuple, M = (Q, Σ, δ, q_0, F), where

Q $\rightarrow$ Finite set of states
$\Sigma \rightarrow$ Finite set of input symbols
$q_0 \in Q \rightarrow$ Initial state
F$\subseteq Q \rightarrow$ Set of final states
and $\delta \rightarrow$ Transition function

It is defined as following:-

δ : Qx$\Sigma \rightarrow$ P(Q)

Where P(Q) is the power set of Q. That is,

$\delta(q,a) \subseteq Q, \qquad \forall q \in Q \textit{ and } a \in \Sigma$

Extended Transition Function
It is denoted by $\hat{\delta}$. It is defined as following:-

$\hat{\delta}$: Q$x\Sigma^* \rightarrow$ P(Q)

Properties of $\hat{\delta}$

1. $\hat{\delta}(q,\epsilon) = \{\text{q}\}$
2. $\hat{\delta}(q,a) = \delta(q,a), \qquad \forall a \in \Sigma$
3. $\hat{\delta}(q,wa) = \bigcup\limits_{p\in\hat{\delta}(q,w)} \delta(p,a), \qquad$ where $q,p \in Q,\ a \in \Sigma \textit{ and } w \in \Sigma^*$

Another Extended Transition Function
It is denoted by $\hat{\hat{\delta}}$. It is defined as following:-

$\hat{\hat{\delta}}$: P(Q)$x\Sigma^* \rightarrow$ P(Q)

Properties of $\hat{\hat{\delta}}$

1. $\hat{\hat{\delta}}(P,a) = \bigcup\limits_{p\in P} \hat{\delta}(p,a)$
2. $\hat{\hat{\delta}}(P,w) = \bigcup\limits_{p\in P} \hat{\delta}(p,w), \qquad$ where $P \subseteq Q,\ a \in \Sigma \textit{ and } w \in \Sigma^*$

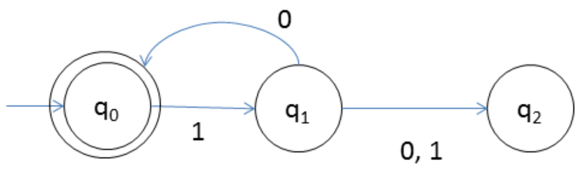
3.3.2 Language Accepted by NFA

Language accepted by NFA M is denoted by L(M). It is defined as following:-

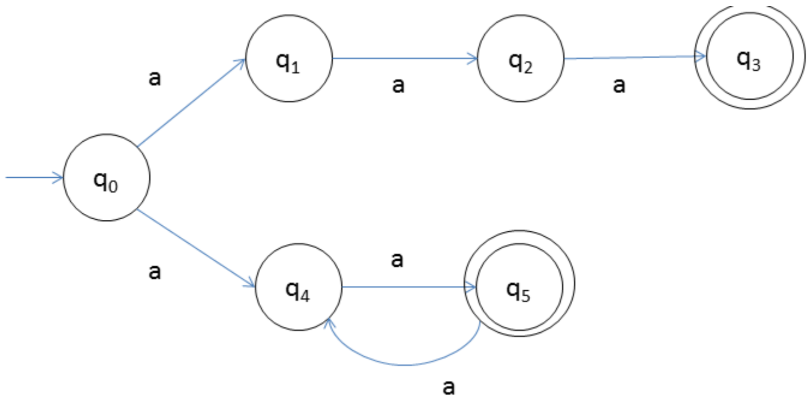
L(M) = $\{\text{x} \in \Sigma^* \ ! \ \hat{\delta}(q_0,x) \cap F \neq \phi\}$

3.3.3 Some Examples

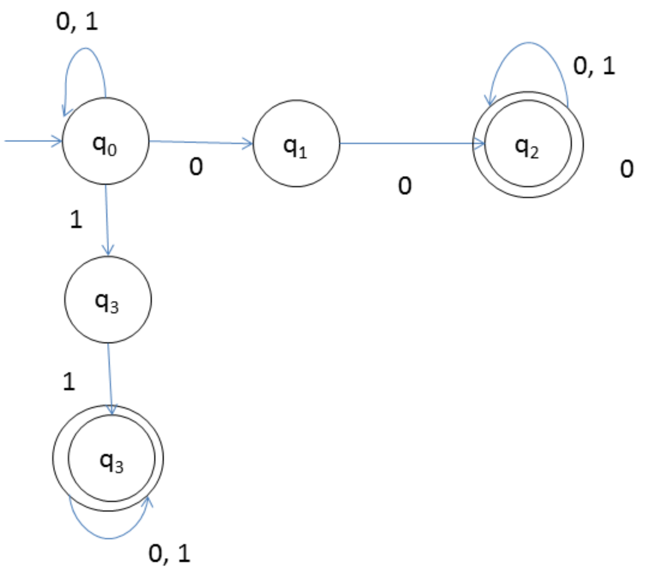
Examples: Find the language accepted by following NFA:-



Examples: Find the language accepted by following NFA:-



Examples: Find the language accepted by following NFA:-



3.3.4 Exercise

1. Design an NFA with no more than five states for the set $\{abab^n$! n≥0$\}\cup\{aba^n$! n≥0$\}$.

2. Construct an NFA with three states that accepts the language $\{ab, abc\}^*$.

3. Find an NFA with three states that accepts the language
   L $= \{a^n \; ! \; n \geq 1\} \cup \{b^m a^k \; ! \; m.k \geq 0\}$

Theorem: For a given NFA M, there exists a DFA M' such that L(M) = L(M').
Proof:
Suppose the given NFA M is

M = (Q,Σ,δ,q_0,F)

Now, we construct a DFA M' as following

$M' = (Q',\Sigma,\delta',q_0',F')$

Where Q' = P(Q), q_0'={q_0}
F' = {P$\subseteq$Q ! P$\cap$F $\neq$ ϕ}
and $\delta' \equiv \hat{\hat{\delta}}$
i.e. $\delta'(P,a) = \hat{\hat{\delta}}(P,a) = \bigcup_{p\in P}\hat{\delta}(p,a) = \bigcup_{p\in P}\delta(p,a)$
Now, we have to show that

L(M) = L(M') .... (1)

To prove the equation (1), first we will prove that

$\hat{\delta}'(q_0',w) = \hat{\delta}(q_0,w)$, $\forall w \in \Sigma^*$................(2)

We will prove this by using induction method on the length of string w.

For $\mid w \mid$= 0, i.e. w = ϵ
$\hat{\delta}'(q_0',\epsilon) = q_0' = \{q_0\} = \hat{\delta}(q_0,\epsilon)$
Therefore, $\hat{\delta}'(q_0',\epsilon) = \hat{\delta}(q_0,\epsilon)$
Therefore, it is proved for w = ϵ.

For $\mid w \mid$= 1, i.e. w = a, where a$\in \Sigma$
$\hat{\delta}'(q_0',a) = \delta'(q_0',a)$

$=\hat{\hat{\delta}}(\{q_0\},a)$

$= \hat{\delta}(q_0,a)$

Therefore, $\hat{\delta}'(q_0',a) = \hat{\delta}(q_0,a)$
Therefore, it is proved for w = a, i.e. $\mid w \mid$= 1.

Suppose equation (2) is true for string w = x.
Therefore, $\hat{\delta}'(q_0',x) = \hat{\delta}(q_0,x)$, ...(3)
Now, we will prove for w = xa.
$\hat{\delta}'(q_0',xa) = \delta'(\hat{\delta}'(q_0',x),a)$

$=\hat{\hat{\delta}}(\hat{\delta}(q_0,x),a)$

$= \bigcup_{p\in\hat{\delta}(q_0,x)} \hat{\delta}(p,a)$

$= \bigcup_{p\in\hat{\delta}(q_0,x)} \delta(p,a)$

$= \hat{\delta}(q_0,xa)$

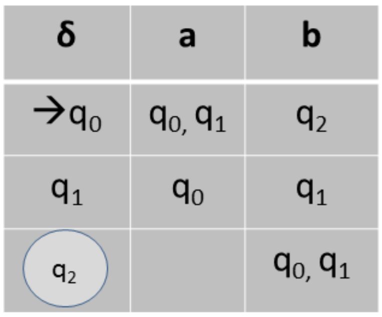
Therefore, $\hat{\delta}'(q_0',xa) = \hat{\delta}(q_0,xa)$
Therefore, it is proved for w = xa.

Therefore, equation (2) is proved.

Now, we prove the equation (1) by using equation (2).
Let w $\in$ L(M') $\Leftrightarrow$ $\hat{\delta}'(q_0', w) \in F'$
$\Leftrightarrow$ $\hat{\delta}'(q_0', w) \cap F \neq \phi$
$\Leftrightarrow$ $\hat{\delta}(q_0, w) \cap F \neq \phi$
$\Leftrightarrow$ $w \in L(M)$
Therefore, L(M) = L(M')
Now, it is proved.

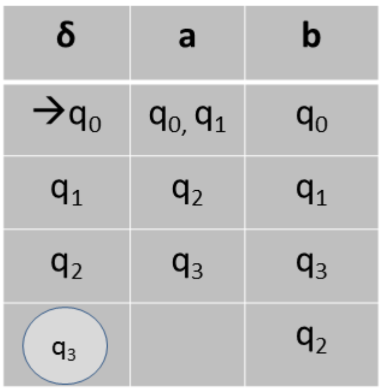
3.3.5 Some Examples

Example: Find DFA equivalent to the following NFA:-
M = $(\{q_0, q_1, q_2\}$,{a,b}, δ, q_0,$\{q_2\})$,
where δ is given by



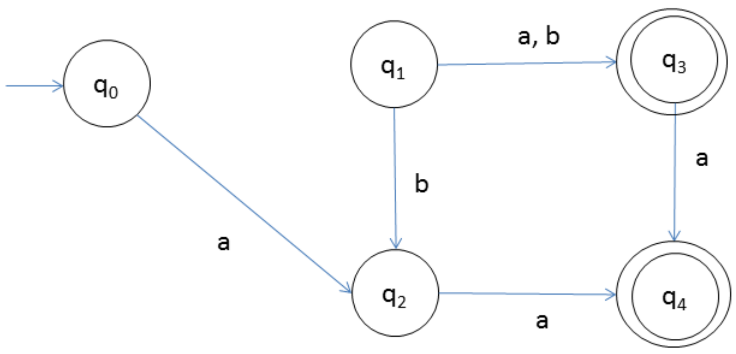
δ	a	b
→q_0	q_0, q_1	q_2
q_1	q_0	q_1
q_2		q_0, q_1

Example: Find DFA equivalent to the following NFA:-
M = $(\{q_0, q_1, q_2, q_3\}$,{a,b}, δ, q_0,$\{q_3\})$,
where δ is given by



δ	a	b
→q_0	q_0, q_1	q_0
q_1	q_2	q_1
q_2	q_3	q_3
q_3		q_2

Example: Construct DFA equivalent to the following NFA:-



Example: Construct DFA equivalent to the following NFA:-

δ	a	b
→q_0	q_1, q_3	q_2, q_3
q_1	q_1	q_3
q_2	q_3	q_2
q_3		

3.4 Myhill-Nerode Theorem

Let L be a language over an alphabet Σ. Define the equivalence relation $\sim_L$ on the set of strings over Σ as follows: for any two strings x and y, $x \sim_L y$ if and only if for all strings z in Σ^*, the concatenated string xz is in L if and only if yz is in L. This relation is said to be right congruence relation.

Theorem:
The Myhill-Nerode Theorem states that:

1. L is a regular language if and only if the equivalence relation $\sim_L$ has a finite index, i.e., there are finitely many equivalence classes under $\sim_L$.

2. If L is a regular language, then the minimal DFA for L has a number of states equal to the index of $\sim_L$, and each state corresponds to an equivalence class under $\sim_L$.

The Myhill-Nerode Theorem provides a useful criterion for determining whether a language is regular or not. If we can prove that the index of $\sim_L$ is finite, then we know that L is regular. Conversely, if we can prove that the index of $\sim_L$ is infinite, then we know that L is not regular.

Proof:
We will prove both directions of the theorem separately.

1. If L is a regular language, then $\sim_L$ has a finite index.

   Proof:
   Suppose L is a regular language. Then there exists a deterministic finite automaton (DFA) $M = (Q, \Sigma, \delta, q_0, F)$ that accepts L, where Q is the set of states, Σ is the input alphabet, δ is the transition function, q_0 is the initial state, and F is the set of accepting states.

   Define an equivalence relation $\sim_M$ on Σ^* as follows: for any two strings x and y in Σ^*, $x \sim_M y$ if and only if the DFA M ends up in the same state after processing x and y, i.e., $\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$, where $\hat{\delta}$ is the extended transition function for M.

   Now, we will show that $\sim_M$ is a refinement of $\sim_L$, i.e., if $x \sim_M y$, then $x \sim_L y$.

   Let x, y, z be strings in Σ^* such that $x \sim_M y$. Then $\hat{\delta}(q_0, x) = \hat{\delta}(q_0, y)$. Therefore, $\hat{\delta}(q_0, xz) = \hat{\delta}(\hat{\delta}(q_0, x), z) = \hat{\delta}(\hat{\delta}(q_0, y), z) = \hat{\delta}(q_0, yz)$, which implies that xz is in L if and only if yz is in L, i.e., $x \sim_L y$.

   Since $\sim_M$ is a refinement of $\sim_L$ and there are $|Q|$ equivalence classes in $\sim_M$, there can be at most $|Q|$ equivalence classes in $\sim_L$, so the index of $\sim_L$ is finite.

2. If $\sim_L$ has a finite index, then L is a regular language.

   Proof:
   Let's construct a DFA $M = (Q, \Sigma, \delta, q_0, F)$ for L as follows:

   (a) Define the set of states Q to be the set of equivalence classes of $\sim_L$.

   (b) Let the initial state q_0 be the equivalence class of the empty string ϵ.

   (c) For each state $[x]$ (the equivalence class containing x), and for each input symbol a, define the transition function $\delta([x], a) = [xa]$, i.e., the equivalence class containing the string xa.

   (d) Let the set of accepting states F consist of all equivalence classes $[x]$ such that x is in L.

   Now, we will show that M accepts L.

   Let x be a string in Σ^*. M accepts x if and only if $\hat{\delta}(q_0, x)$ is in F. Since $\hat{\delta}(q_0, x) = [x]$, M accepts x if and only if $[x]$ is in F, which is true if and only if x is in L. Thus, M accepts L, and so L is a regular language.

By proving both directions of the theorem, we have shown that L is a regular language if and only if $\sim_L$ has a finite index.

Example: Is language $L_1 = \{w \in \{0, 1\}^* \mid w \text{ has an even number of 0's}\}$ regular or not?

Solution:
Consider the language L_1 over the alphabet $\Sigma = \{0, 1\}$. We will show that L_1 is a regular language by proving that the equivalence relation $\sim_{L_1}$ has a finite index.

Let's analyze the equivalence classes:

1. Strings with an even number of 0s: For example, ϵ, 00, 0101, 1110, etc. All these strings have the same future with respect to L_1, as appending any string with an even number of 0s will still result in a string in L_1. Thus, they form an equivalence class [E].

2. Strings with an odd number of 0s: For example, 0, 100, 110, 1011, etc. All these strings have the same future with respect to L_1, as appending any string with an even number of 0s will not result in a string in L_1. Thus, they form an equivalence class [O].

Since we have found only two equivalence classes, $\sim_{L_1}$ has a finite index, and thus, L_1 is a regular language.

Example: Is language $L_2 = \{0^n 1^n \mid n \geq 0\}$ regular or not?
Solution:
Consider the language L_2 over the alphabet $\Sigma = \{0, 1\}$. We will show that L_2 is not a regular language by proving that the equivalence relation $\sim_{L_2}$ has an infinite index.

Let n be a non-negative integer. Consider the strings 0^n and $0^{(n+1)}$. We will show that $0^n \sim_{L_2} 0^{(n+1)}$ does not hold. Let $z = 1^n$. Then, $0^n 1^n$ is in L_2, but $0^{(n+1)} 1^n$ is not in L_2. Thus, 0^n and $0^{(n+1)}$ are not equivalent under $\sim_{L_2}$.

Since n can be any non-negative integer, we have an infinite number of distinct equivalence classes $\{[0^n] \mid n \geq 0\}$. Therefore, the equivalence relation $\sim_{L_2}$ has an infinite index, and L_2 is not a regular language.

Example: Is language of strings with no consecutive 0s regular or not?
Solution:
Let $L_1 = \{w \in \{0, 1\}^* | \text{no two consecutive 0s appear in w}\}$.

We will show that there is a finite number of equivalence classes under the right congruence relation for the language L_1.

Consider the strings $x = \epsilon$ (the empty string) and $y = 0$. For any string z, if z starts with 1 or is empty, then $xz \in L_1$ and $yz \in L_1$. If z starts with 0, then $xz \in L_1$, but $yz \notin L_1$. Therefore, x and y belong to different equivalence classes.

Consider the strings $x = 0$ and $y = 1$. For any string z, if z starts with 1 or is empty, then $xz \in L_1$ and $yz \in L_1$. If z starts with 0, then $xz \notin L_1$ and $yz \in L_1$. Therefore, x and y belong to different equivalence classes.

Consider the strings $x = 1$ and $y = 11$. For any string z, if z starts with 1 or is empty, then $xz \in L_1$ and $yz \in L_1$. If z starts with 0, then $xz \in L_1$ and $yz \in L_1$. Therefore, x

and y belong to the same equivalence class.

From the above analysis, we can conclude that there are three equivalence classes under the right congruence relation:

1. One class containing strings that can be followed by either 0 or 1 without violating the language rule (such as ϵ, 1, 11, 101, ...).
2. One class containing strings that end with a single 0 and can be followed only by 1 (such as 0, 10, 110, ...).
3. One class containing strings that end with consecutive 0s and are not in the language (such as 00, 100, 11100, ...).

Since there is a finite number of equivalence classes under the right congruence relation, therefore by the Myhill-Nerode Theorem, the language L_1 is regular.

Example: Is language of strings with a length that is a multiple of 3 regular or not?
Solution:
Let $L_2 = \{w \in \{0,1\}^* || w| \text{ is a multiple of } 3\}$.

We will show that there is a finite number of equivalence classes under the right congruence relation for the language L_2.

Consider the strings $x = \epsilon$ (the empty string), $y = 0$, and $z = 00$. For any string $w \in \Sigma^*$, $xw \in L_2$ if and only if $|w|$ is a multiple of 3, $yw \in L_2$ if and only if $|w|$ is a multiple of 3 minus 1, and $zw \in L_2$ if and only if $|w|$ is a multiple of 3 minus 2. Therefore, x, y, and z belong to different equivalence classes.

From the above analysis, we can conclude that there are three equivalence classes under the right congruence relation:

1. One class containing strings with a length that is a multiple of 3 (such as ϵ, 000, 111, 001100, ...).
2. One class containing strings with a length that is a multiple of 3 minus 1 (such as 0, 11, 0101, ...).
3. One class containing strings with a length that is a multiple of 3 minus 2 (such as 00, 1, 1100, ...).

Since there is a finite number of equivalence classes under the right congruence relation, by the Myhill-Nerode Theorem, the language L_2 is regular.

3.5 Minimization of Finite Automata

Equivalent states
Two states q_1 and q_2 are said to be equivalent if both $\hat{\delta}(q_1, x)$ and $\hat{\delta}(q_2, x)$ are final states or both of them are non-final states for all $x \in \Sigma^*$.

K-equivalent states
Two states q_1 and q_2 are said to be k-equivalent if both $\hat{\delta}(q_1, x)$ and $\hat{\delta}(q_2, x)$ are final states or both of them are non-final states for all $x \in \Sigma^*$ and $\mid x \mid \leq k$.

3.5.1 Construction of Minimum Automata

Step 1: First we find a set Π_0 that consists of two sets. First is the set of final states and second is the set of non-final states. That is,
$\Pi_0 = \{$ F, Q-F $\}$
$\Pi_k \rightarrow$ Set of k-equivalence classes
Q $\rightarrow$ Set of states
F $\rightarrow$ Set of final states

Step 2 (Construction of Π_{k+1} from Π_k):

1. Put all the equivalence classes or sets of Π_k into Π_{k+1} as it as if it consists of single states.

2. Let S be a set belong into Π_k. Let q_i and q_j are the two states belong into S.

3. Compute states q_i and q_j (k+1)-equivalent or not.

4. If they are (k+1)-equivalent then put these two states in the same set of Π_{k+1}, otherwise both states belong into different sets in Π_{k+1}.

5. Similarly, we check all pairs of states in S. And put the states either in same set or in different set in Π_{k+1}.

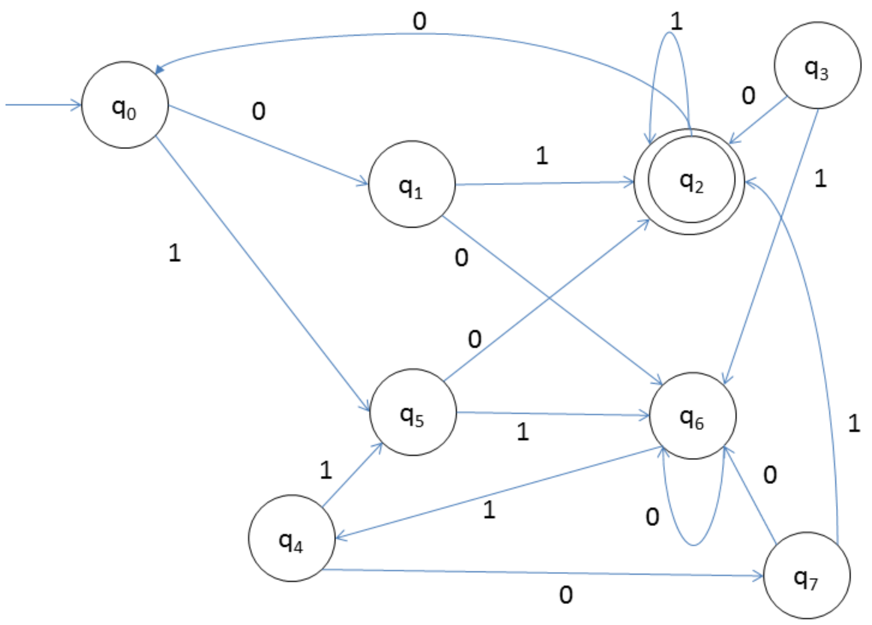
6. Similarly, we apply above procedure for all the sets belong into Π_k.

Step 3: Construct Π_n for n= 1,2,3,4,............., until $\Pi_n = \Pi_{n+1}$.

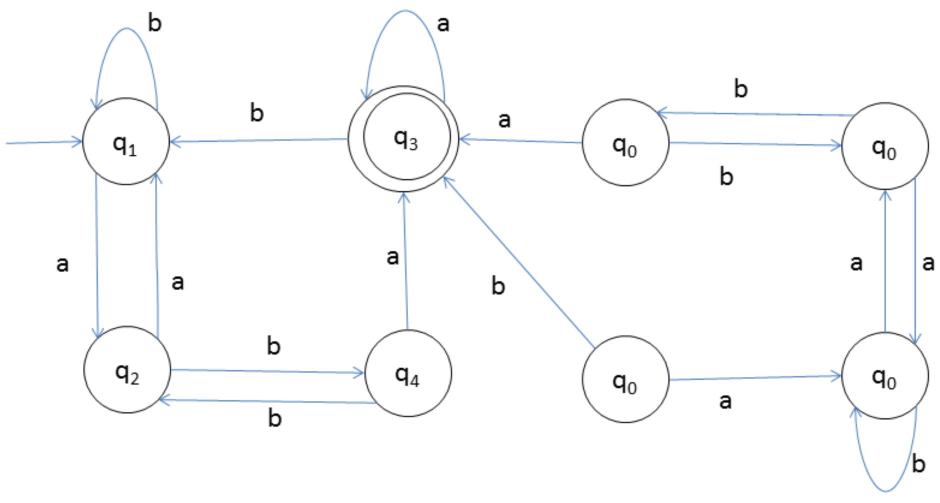
Step 4: For the required minimum state automata, the states are the equivalence classes obtained in step 3 i.e. the elements of Π_n. The state table is obtained by replacing a state q by the corresponding equivalence class [q].

3.5.2 Some Examples

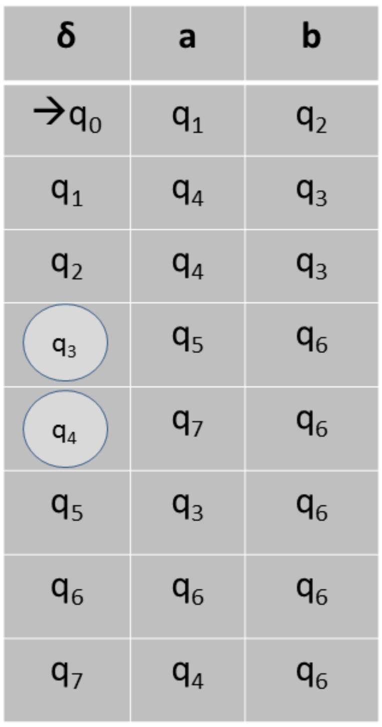
Example: Construct a minimum state automata equivalent to the following finite automata:



Example: Minimize the following automata:-



Example: Minimize the following automata:-



δ	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_4	q_3
q_2	q_4	q_3
q_3	q_5	q_6
q_4	q_7	q_6
q_5	q_3	q_6
q_6	q_6	q_6
q_7	q_4	q_6

3.6 Mealy and Moore Machines(Finite Automata with Outputs)

Moore Machine

Moore machine is a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where
Q$\rightarrow$ Finite set of states
$\Sigma \rightarrow$ Finite set of input symbols
$\Delta \rightarrow$ Finite set of output symbols
$q_0 \in Q \rightarrow$ Initial state
$\delta \rightarrow$ Transition function
It is defined as following:-

δ : Qx$\Sigma \rightarrow$ Q

$\lambda \rightarrow$ Output function
It is defined as following:-

$\lambda : Q \rightarrow \Delta$

Mealy Machine

Mealy machine is a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where
Q$\rightarrow$ Finite set of states
$\Sigma \rightarrow$ Finite set of input symbols
$\Delta \rightarrow$ Finite set of output symbols
$q_0 \in Q \rightarrow$ Initial state
$\delta \rightarrow$ Transition function
It is defined as following:-

$\delta : Qx\Sigma \rightarrow Q$

$\lambda \rightarrow$ Output function

It is defined as following:-

$\lambda : Qx\Sigma \rightarrow \Delta$

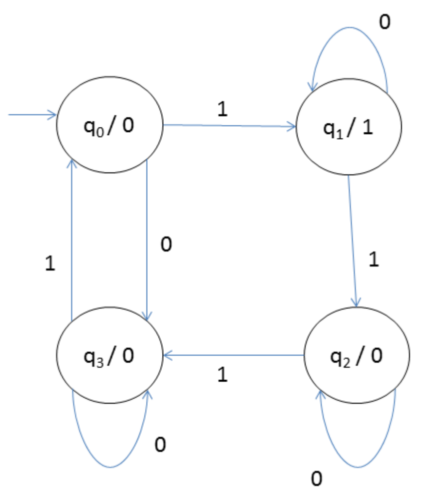
3.6.1 Representation of Moore machine

Moore machine is represented by the following two ways:-

(1) By transition table

Present State	Next State δ		Output λ
	0	1	
$\rightarrow q_0$	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

(2) By transition diagram



Example: Find the output string corresponding to the input string 0111 in the above Moore machine.

$q_0 \rightarrow q_3 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2$

The output string is 00010.

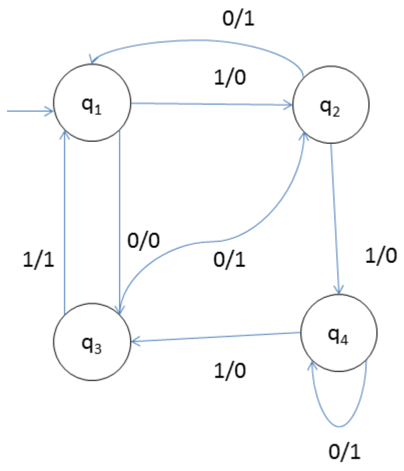
3.6.2 Representation of Mealy machine

Mealy machine is represented by the following two ways:-

(1) By transition table

Present State	0		1	
	δ	λ	δ	λ
$\to q_1$	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

(2) By transition diagram



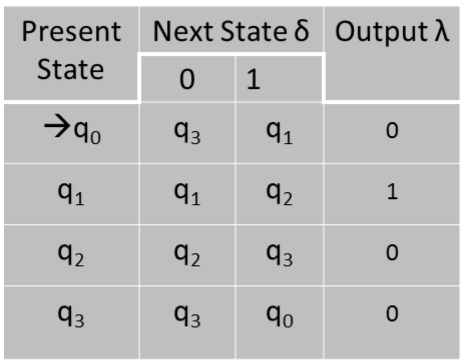
Example: Find the output string corresponding to the input string 0011 in the above Moore machine.
$q_1 \to q_3 \to q_2 \to q_4 \to q_3$
The output string is 0100.

3.6.3 Procedure for transforming a Moore machine into a Mealy machine

(1) The output function λ' for the Mealy machine is determined as following:-
$\lambda'(q, a) = \lambda(\delta(q, a)), \forall q \in Q,\ a \in \Sigma$

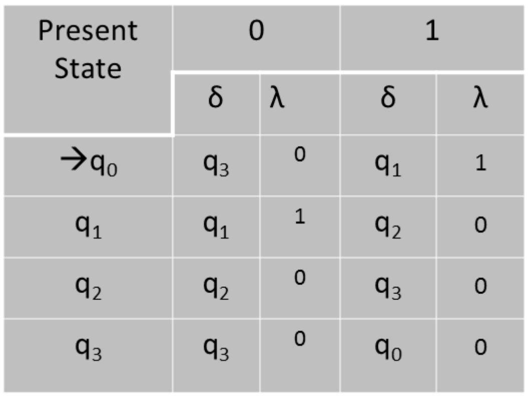
(2) The transition function is the same as that of the given Moore machine.

Example: Construct a Mealy machine which is equivalent to the following Moore Machine



Present State	Next State δ		Output λ
	0	1	
→q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

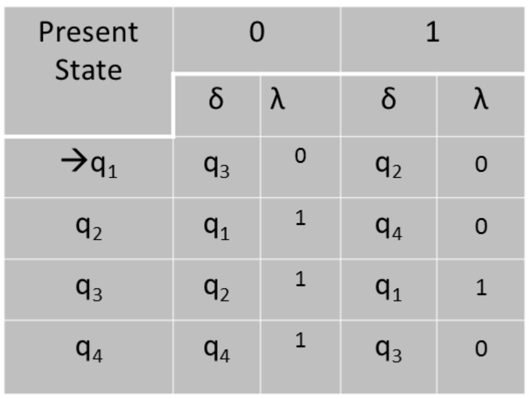
Solution: Mealy machine for the above Moore machine is the following:-



Present State	0		1	
	δ	λ	δ	λ
→q_0	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_0	0

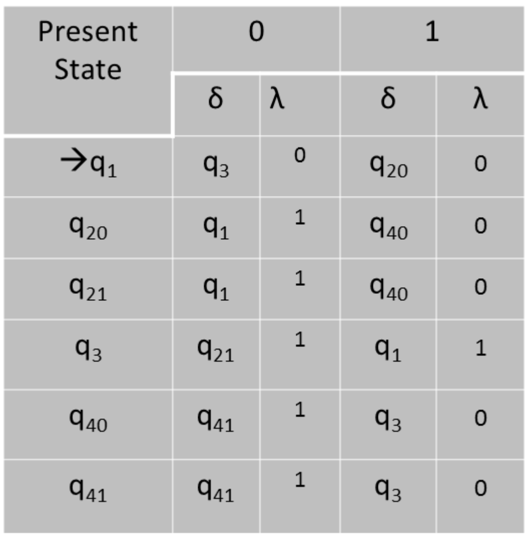
3.6.4 Procedure for transforming a Mealy machine into a Moore machine

Example: Construct a Moore machine which is equivalent to the following Mealy machine



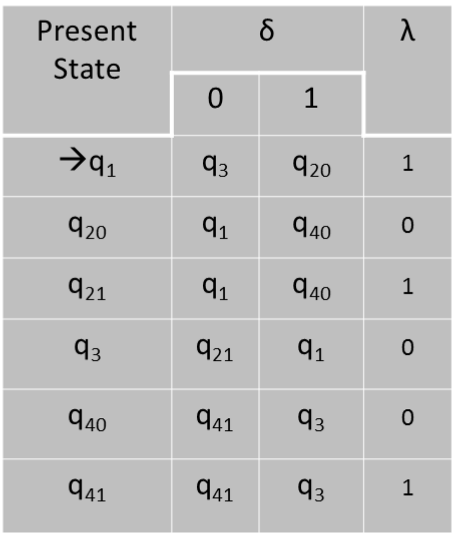
Present State	0		1	
	δ	λ	δ	λ
→q_1	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

Solution: In the first step we split the some states. We split those states that has different outputs. In this example, we split states q_2 and q_4.



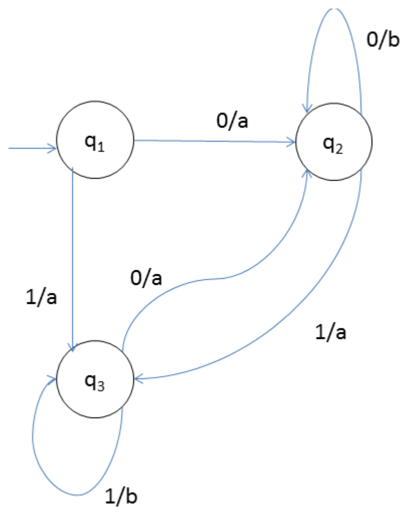
Present State	0		1	
	δ	λ	δ	λ
→q_1	q_3	0	q_{20}	0
q_{20}	q_1	1	q_{40}	0
q_{21}	q_1	1	q_{40}	0
q_3	q_{21}	1	q_1	1
q_{40}	q_{41}	1	q_3	0
q_{41}	q_{41}	1	q_3	0

Now, we convert above Mealy machine into Moore machine:-



Present State	δ		λ
	0	1	
→q_1	q_3	q_{20}	1
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_3	q_{21}	q_1	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

Example: Construct a Moore machine which is equivalent to the following Mealy machine



Example: Construct a Mealy machine which can output EVEN, ODD as the total number of 1's encountered is even or odd. The input symbols are 0 and 1.

Example: Construct a Mealy machine that prints 1's complement of an input binary number.

Example: Design an incremental Mealy machine.

Example: Design a Mealy machine that prints 2's complement of an input binary number.

Example: Design a Moore machine to determine residue mod 3 for binary number.

3.7 Non-deterministic finite automata with null transition($\epsilon-$ $move$)

A non-deterministic finite automata with $\epsilon - move$ M is a 5-tuple, M = (Q, Σ, δ, q_0, F), where

Q $\rightarrow$ Finite set of states
$\Sigma \rightarrow$ Finite set of input symbols
$q_0 \in Q \rightarrow$ Initial state
F$\subseteq Q \rightarrow$ Set of final states
and $\delta \rightarrow$ Transition function

It is defined as following:-

$$\delta : \text{Q}x(\Sigma \cup \{\epsilon\}) \rightarrow \text{P(Q)}$$

$\epsilon - closure(q)$
$\epsilon - closure$ of a state q is the set of all the states reachable from q by using only $\epsilon - move$.

$\epsilon - closure(P)$

Here P⊆Q.
$\epsilon - closure(P) = \underset{p \in P}{\cup} \epsilon - closure(p)$

Extended Transition Function
It is denoted by $\hat{\delta}$. It is defined as following:-
$\hat{\delta}$: Q$x\Sigma^* \rightarrow$ P(Q)

Properties of $\hat{\delta}$

1. $\hat{\delta}(q, \epsilon) = \epsilon - closure(q)$
2. $\hat{\delta}(q, a) = \epsilon - closure(\underset{p \in \epsilon - closure(q)}{\cup} \delta(p, a))$
3. $\hat{\delta}(q, wa) = \epsilon - closure(\underset{p \in \hat{\delta}(q,w)}{\cup} \delta(p, a))$, where $q, p \in Q,\ a \in \Sigma\ and\ w \in \Sigma^*$

Another Extended Transition Function
It is denoted by $\hat{\hat{\delta}}$. It is defined as following:-
$\hat{\hat{\delta}}$: P(Q)$x\Sigma^* \rightarrow$ P(Q)

Properties of $\hat{\hat{\delta}}$

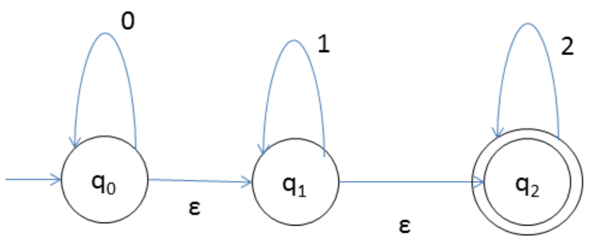
1. $\hat{\hat{\delta}}(P, a) = \underset{p \in P}{\cup} \hat{\delta}(p, a)$
2. $\hat{\hat{\delta}}(P, w) = \underset{p \in P}{\cup} \hat{\delta}(p, w)$, where $P \subseteq Q,\ a \in \Sigma\ and\ w \in \Sigma^*$

3.7.1 Language Accepted by NFA with null transition($\epsilon - move$)

Language accepted by NFA M is denoted by L(M). It is defined as following:-
L(M) = {x ∈ Σ^* ! $\hat{\delta}(q_0, x) \cap F \neq \phi$}

3.7.2 Some Examples

Examples: Consider the following NFA:-



Determine the followings:-

(1) $\epsilon - closure\ of\ q_0,\ q_1,\ and\ q_2$
(2) $\hat{\delta}(q_0, 0)$, $\hat{\delta}(q_0, 00)$, $\hat{\delta}(q_0, 011)$, and $\hat{\delta}(q_1, 12)$.

(3) Language accepted by this NFA.

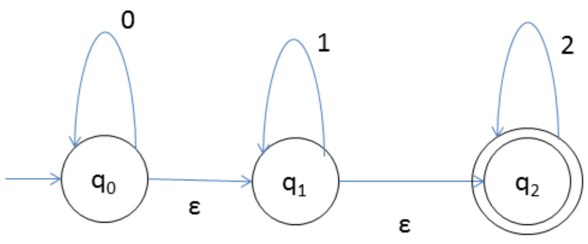
3.7.3 Conversion of NFA with $\epsilon-move$ into NFA without $\epsilon-move$

Suppose the given NFA with $\epsilon-move$ M is the following:-
M = (Q,Σ,δ,q_0,F)
Now, we construct M' as the following:-
M' = (Q,Σ,δ',q_0,F')
Where Q, Σ, q_0 are the same as M and

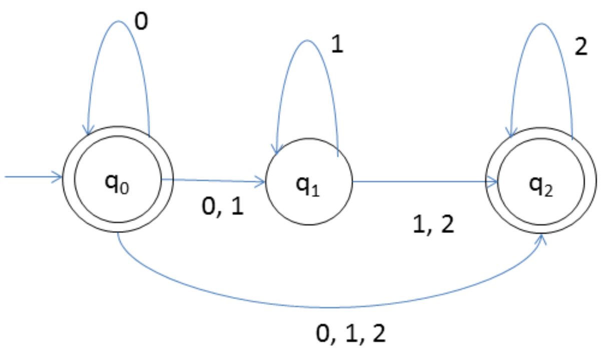
$$F' = \begin{cases} F \cup \{q_0\} \ , & if \quad \epsilon-closure(q_0) \cap F \neq \phi \\ F \ \ , & otherwise \end{cases} \tag{3.1}$$

δ' is defined as following
$\delta'(q,a) = \hat{\delta}(q,a)$, for all q$\in$Q and a$\in$ Σ.

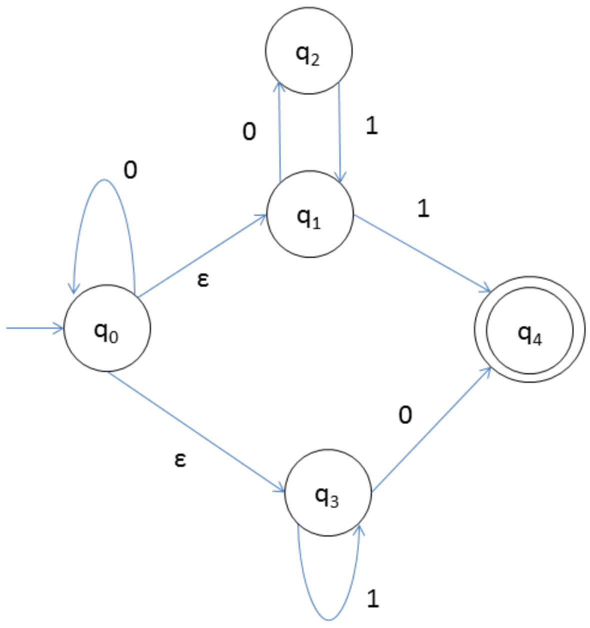
Examples: Consider the following NFA with $\epsilon-move$:-



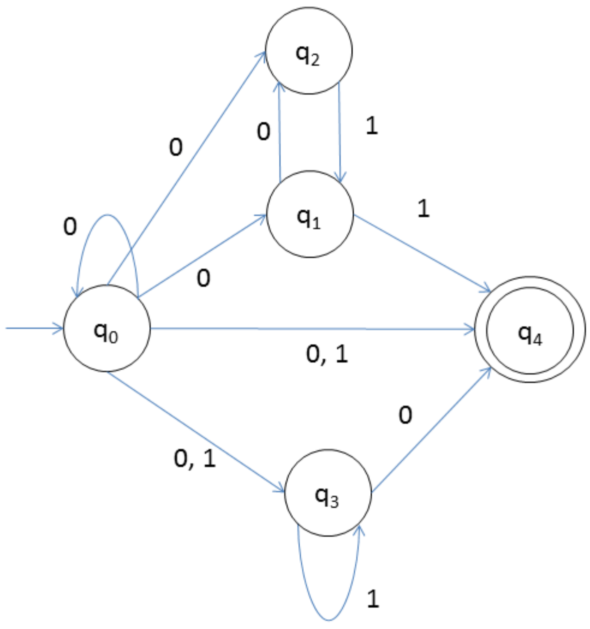
Find NFA without $\epsilon-move$ equivalent to this.
Solution:



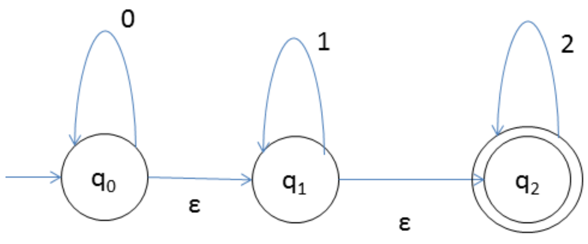
Examples: Consider the following NFA with $\epsilon-move$:-



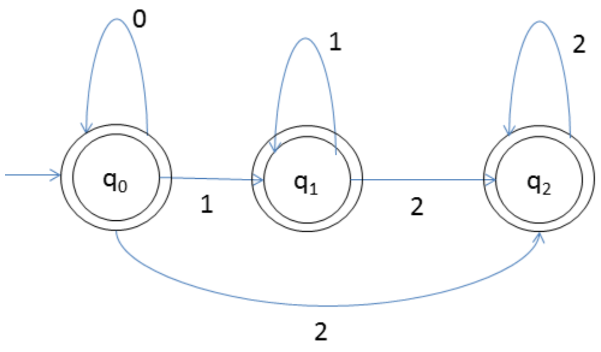
Find NFA without $\epsilon - move$ equivalent to this.
Solution:



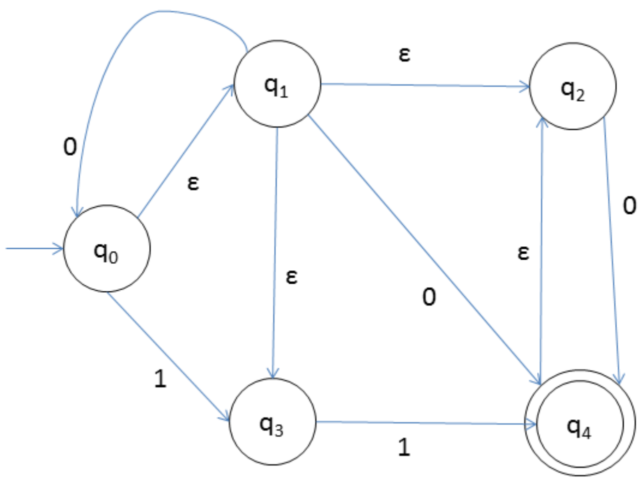
Examples: Consider the following NFA with $\epsilon - move$:-



Construct DFA equivalent to this.
Solution:



Examples: Consider the following NFA with $\epsilon - move$:-



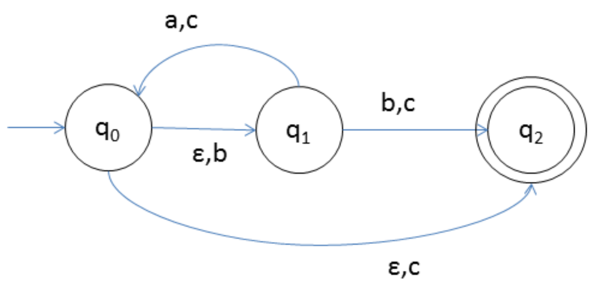
Construct DFA equivalent to this.

3.8 AKTU Examination Questions

1. Design a FA to accept the string that always ends with 101.

2. What do you mean by $\epsilon - Closure$ in FA?

3. Construct a minimum state DFA for the given FA:-



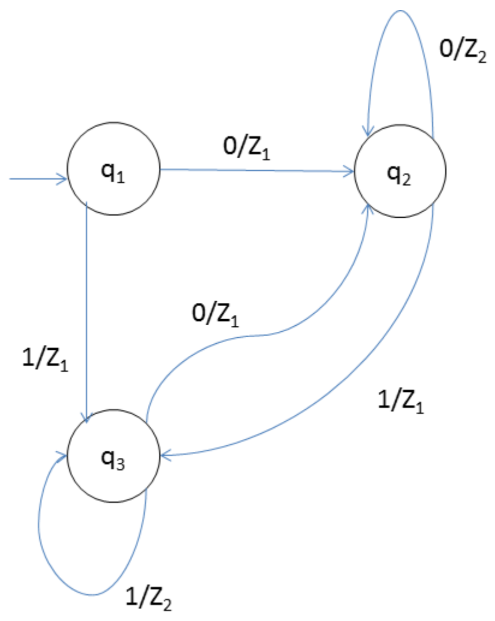
4. Design FA for ternary number divisible by 5.

5. Explain Myhill-Nerode Theorem using suitable example.

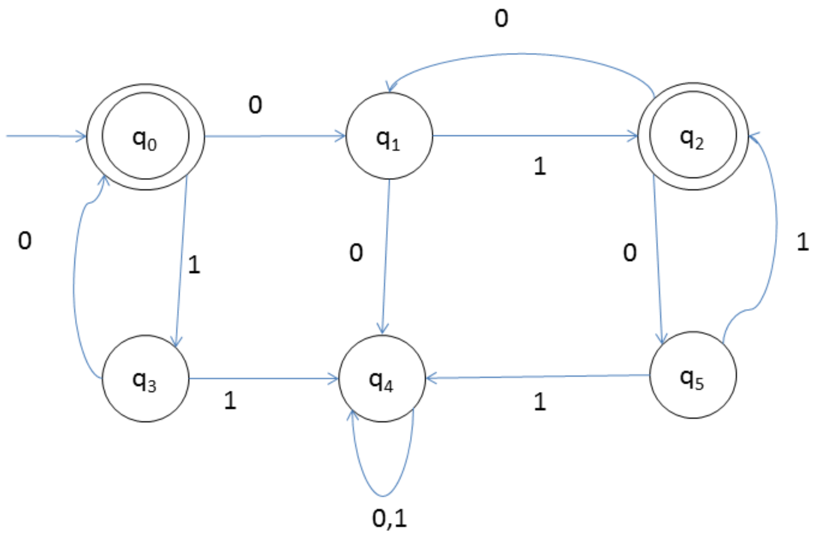
6. Design a NFA that accepts all the strings for input alphabet {a,b} containing the substring abba.

7. Define Deterministic Finite Automata(DFA)and design a DFA that accepts binary numbers whose equivalent decimal number is divisible by 5.

8. Describe Mealy and Moore machine. Convert the following Mealy machine into Moore machine:-



9. Construct minimum state automata equivalent to the following DFA:-



10. Explain the applications and limitations of finite automata.

11. What is extended transition function δ^*? Explain with example.

12. Give the difference between Mealy and Moore machine.

13. Design DFA to accept all string over {0, 1} not ending with 10.

14. Give finite automata for:
    i) L = $\{a^n b^{2m} c^{3l}$! n, m, l $\geq 0\}$.
    ii) L = $\{a^n b^{2m}$! $0 < n < 3, m \geq 0\}$.

15. Differentiate between FA and PDA.

16. Write the applications of finite automata.

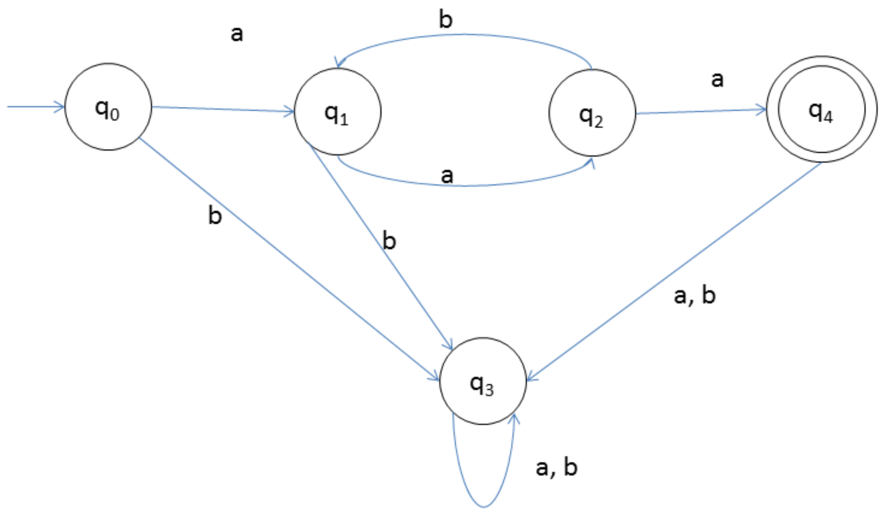
17. How do we determine equivalence of two DFA? Explain with an example.

18. Design FA for the following languages containing binary strings:
    (i) Having both 00 and 11 as substring.
    (ii) Number of 0's is odd and number of 1's is multiple of 3.

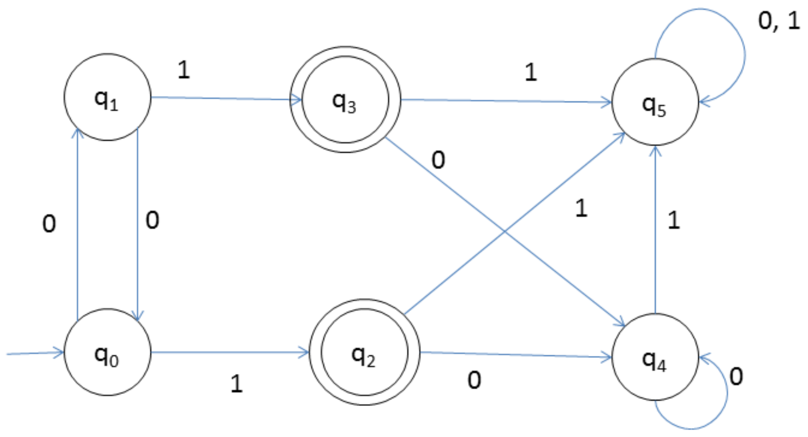
19. Design an NFA for the language containing strings ending with ab or ba. Also convert the obtained NFA into equivalent DFA.

20. Design the DFA that accepts an even number of a's and even number of b's.

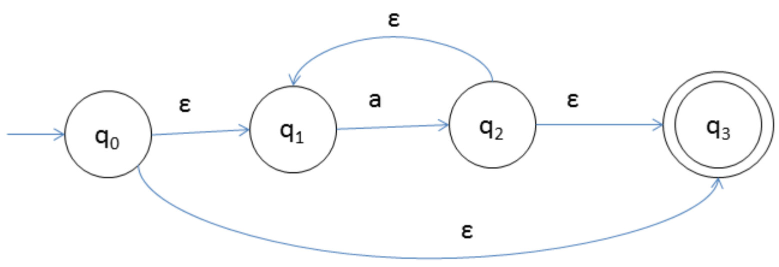
21. Consider the DFA given below and identify the L accepted by the machine.



22. Minimize the automata given below



23. Compute the epsilon- closure for the given NFA. Convert it into DFA.



24. Design a DFA for languages L containing strings of 0 and 1's where number of 0's is not divisible by 3.

25. Define NFA. What are various points of difference between NFA and DFA?

26. What are various points of difference between Moore & Mealy Machine? Explain the procedure to convert a Moore machine into Mealy machine.