Lecture Notes
on
Theory of Automata and Formal Languages

Dharmendra Kumar

June 16, 2021

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 Σ = {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. $| \epsilon | = 0$
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. $| uv | = | u | + | v |$
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.
$\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup .....$

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.
$\Sigma^+ = \bigcup_{i=1}^{\infty} \Sigma^i = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup .....$
$= \Sigma^*$ - $\{\epsilon\}$

Example: Consider Σ = {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 → Finite set of variables or non-terminal symbols
Σ → Finite set of terminal symbols
S ∈ V → Starting symbols of the Grammar
P → Finite set of production rules

And P is defined as following:-
$P \subseteq (V \cup \Sigma)^* X (V \cup \Sigma)^*$
(u,v)∈ P is denoted by
u → 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 $\alpha \to \beta \in P$. 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 $\alpha \to A_1$, $A_1 \to A_2$,$A_2 \to A_3$,.....$A_n \to \beta$, such that
$\alpha \Rightarrow A_1 \Rightarrow A_2 \Rightarrow A_3 \Rightarrow \ldots\ldots\ldots\ldots \Rightarrow A_n \Rightarrow \beta$. 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,

$$\mathrm{L(G)} = \{\ \mathrm{x} \in \Sigma^* \ !\ S \overset{*}{\Rightarrow} x\ \}$$

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,

$$\mathrm{L}(G_1) = \mathrm{L}(G_2).$$

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) $\mathrm{L(G)} = \{\ a^n b a^m\ !\ \mathrm{m,n} \geq 1\}$
   (b) L(G) = The set of all palindromes over {a,b}
   (c) $\mathrm{L(G)} = \{\ wcw^T\ !\ \mathrm{w} \in \{a,b\}^*\ \}$
   (d) $\mathrm{L(G)} = \{\ a^n b^n c^i\ !\ \mathrm{n} \geq 1 \text{ and i} \geq 0\}$
   (e) $\mathrm{L(G)} = \{\ a^j b^n c^n\ !\ \mathrm{n} \geq 1 \text{ and j} \geq 0\}$
   (f) $\mathrm{L(G)} = \{\ a^n b^n c^n\ !\ \mathrm{n} \geq 1\}$
4. Construct the grammars which generates the following languages:-
   (a) $\mathrm{L(G)} = \{\ \mathrm{w} \in \{a,b\}^*\ !$ The number of a's in w is divisible by 3$\}$
   (b) $\mathrm{L(G)} = \{\ \mathrm{w} \in \{a,b\}^*\ !$ w has an equal number of a's and b's$\}$
   (c) $\mathrm{L(G)} = \{\ \mathrm{w} \in \{a,b\}^*\ !\ n_a(w) > n_b(w)\}$
   (d) $\mathrm{L(G)} = \{\ \mathrm{w} \in \{a,b\}^*\ !\ n_a(w) \neq n_b(w)\}$
   (e) $\mathrm{L(G)} = \{\ 0^m 1^n 0^n 1^m\ !\ \mathrm{m,n} \geq 1\ \}$
   (f) $\mathrm{L(G)} = \{\ 0^n 1^{2n}\ !\ \mathrm{n} \geq 1\}$

(g) L(G) = { 0^n1^n ! n ≥ 1}∪{ 1^m0^m ! m ≥ 1}

(h) L(G) = { $0^n1^m0^n$! m,n ≥ 1}∪{ $0^n1^m2^m$! m,n ≥ 1}

5. Find grammars for Σ = {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 Σ = {a}

   (a) L = { w ! | w | mod 3 = 0}

   (b) L = { w ! | w | mod 3 > 0}

   (c) L = { w ! | w | mod 3 ≠ | w | mod 2}

   (d) L = { w ! | w | mod 3 ≥ | w | mod 2}

7. Find grammars for the following languages over Σ = {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 ! | $n_a(w) - n_b(w)$ | = 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:-

$$\phi_1 A \phi_2 \rightarrow \phi_1 \psi \phi_2, \qquad \text{where } \phi_1, \phi_2, \psi \in (V \cup \Sigma)^* \text{ and A} \in \text{V}.$$

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:-

$$A \rightarrow \psi, \qquad \text{where } \psi \in (V \cup \Sigma)^* \text{ and A} \in \text{V}.$$

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

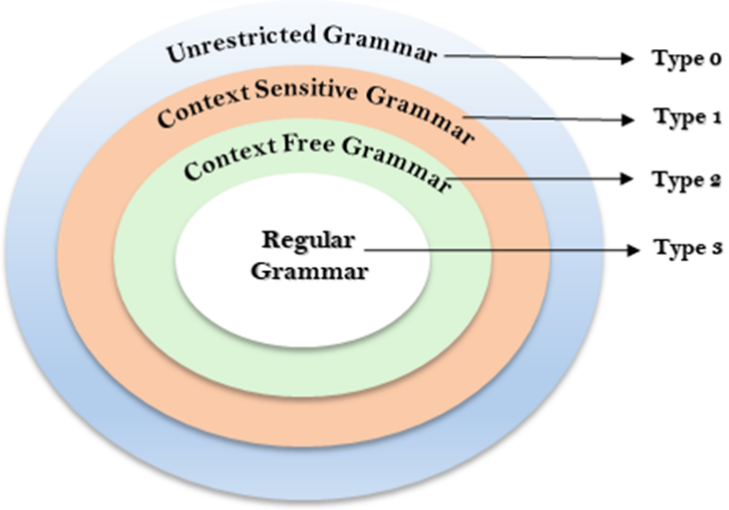


Fig: Chomsky Hierarchy

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) $S \rightarrow 0S1/0A/0/1B/1, \quad A \rightarrow 0A/0, \quad B \rightarrow 1B/1$

(c) $S \rightarrow 0SBA/01A, \quad AB \rightarrow BA, \quad 1B \rightarrow 11, \quad 1A \rightarrow 10, \quad 0A \rightarrow 00$

(d) $S \rightarrow 0S1/0A1, \quad A \rightarrow 1A0/10$

(e) $S \rightarrow 1S/0A/0/1, \quad A \rightarrow 1A/1S/1$

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

(a) L = $\{0^m 1^n \mid m \neq n \text{ and } m, n \geq 1\}$

(b) L = $\{a^l b^m c^n \mid$ one of l,m,n equals 1 and the remaining two are are equal $\}$

(c) L = $\{0^m 1^n \mid 1 \leq m \leq n\}$

(d) L = $\{a^l b^m c^n \mid l + m = n\}$

(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 = $\{a^{2n} \mid n \geq 1\}$

(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 = $\{a^l b^m c^n \mid l, m, n \geq 1\}$

(e) L = $\{(ab)^n \mid n \geq 1\}$

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 = $\{0^m 1^n \mid m \neq n \text{ and } m, n \geq 1\}$

(b) L = $\{a^l b^m c^n \mid l + m = n \text{ and } l, m \geq 1\}$

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 = $\{a^n b^m \mid n \neq m\}$. Also explain the type of this language.

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

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

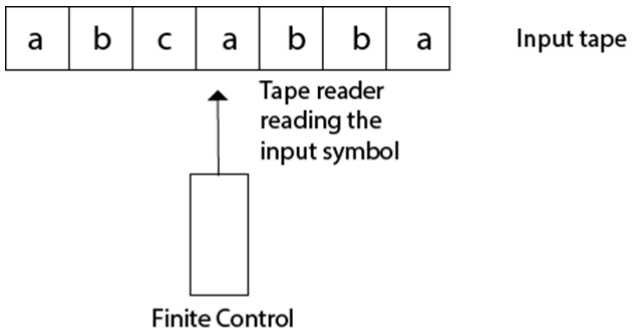


Fig :- Finite automata model

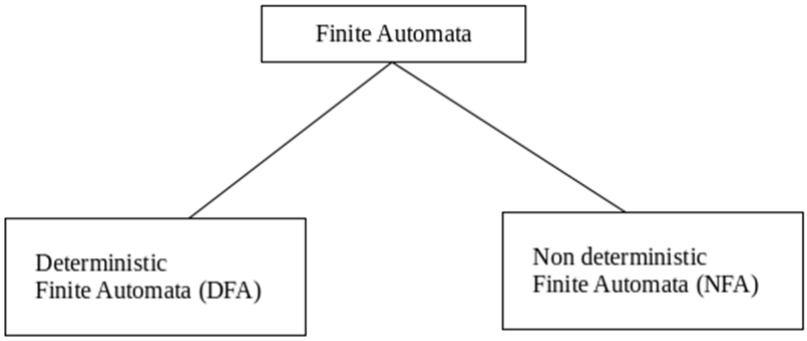
is placed in each cell.

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.



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:-
δ : QxΣ → Q

Extended Transition Function
It is denoted by $\hat{\delta}$. It is defined as following:-
$\hat{\delta}$: QxΣ* → 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:-
L(M) = {x ∈ Σ* ! $\hat{\delta}(q_0, x) \in F$}

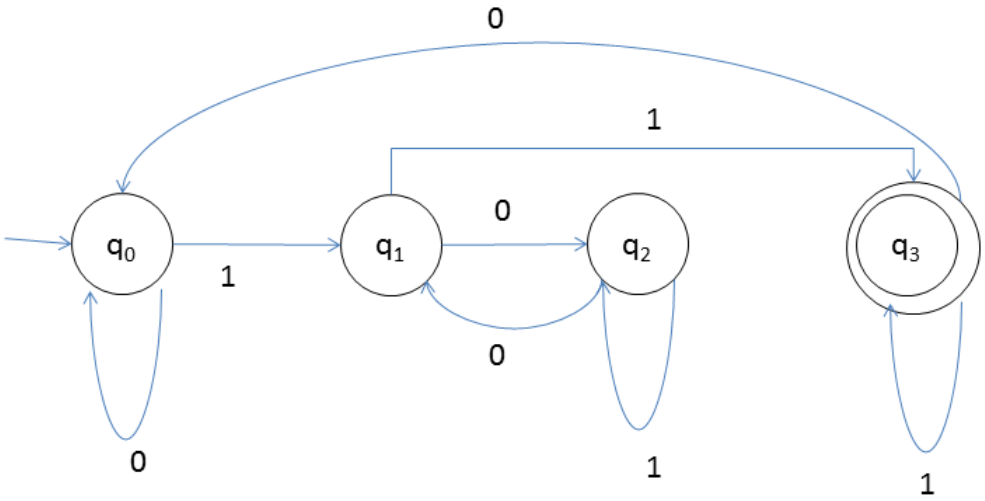
3.2.3 Representation of DFA

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
→ q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_1	q_2
(q_3)	q_0	q_3

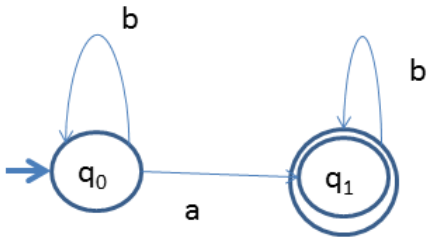
(2) By transition diagram
.



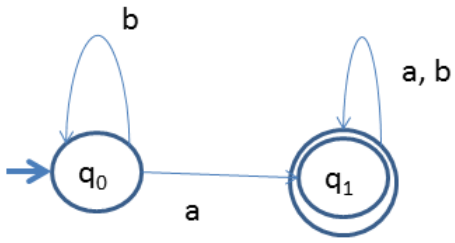
3.2.4 Some Examples

Example: Find the language accepted by following DFA:-

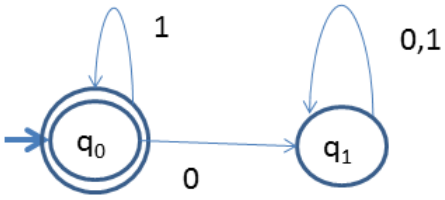
.



Example: Find the language accepted by following DFA:-

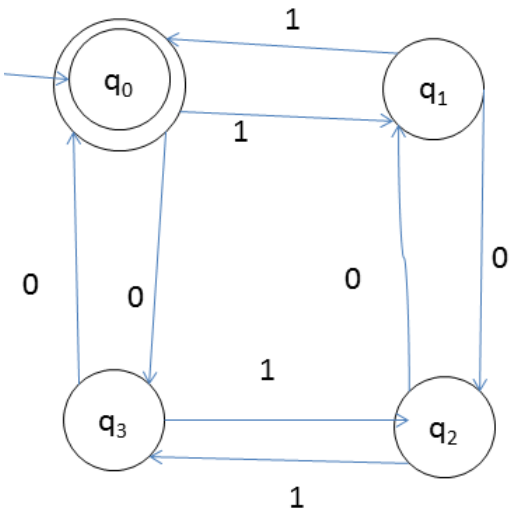


Example: Find the language accepted by following DFA:-

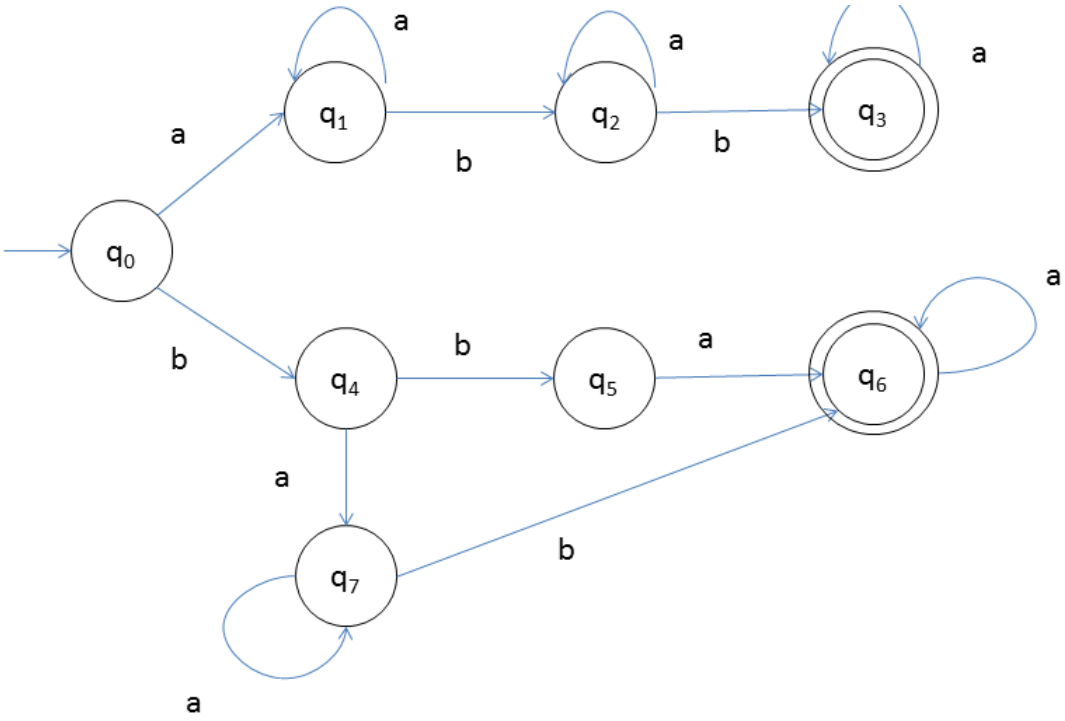


Solution:

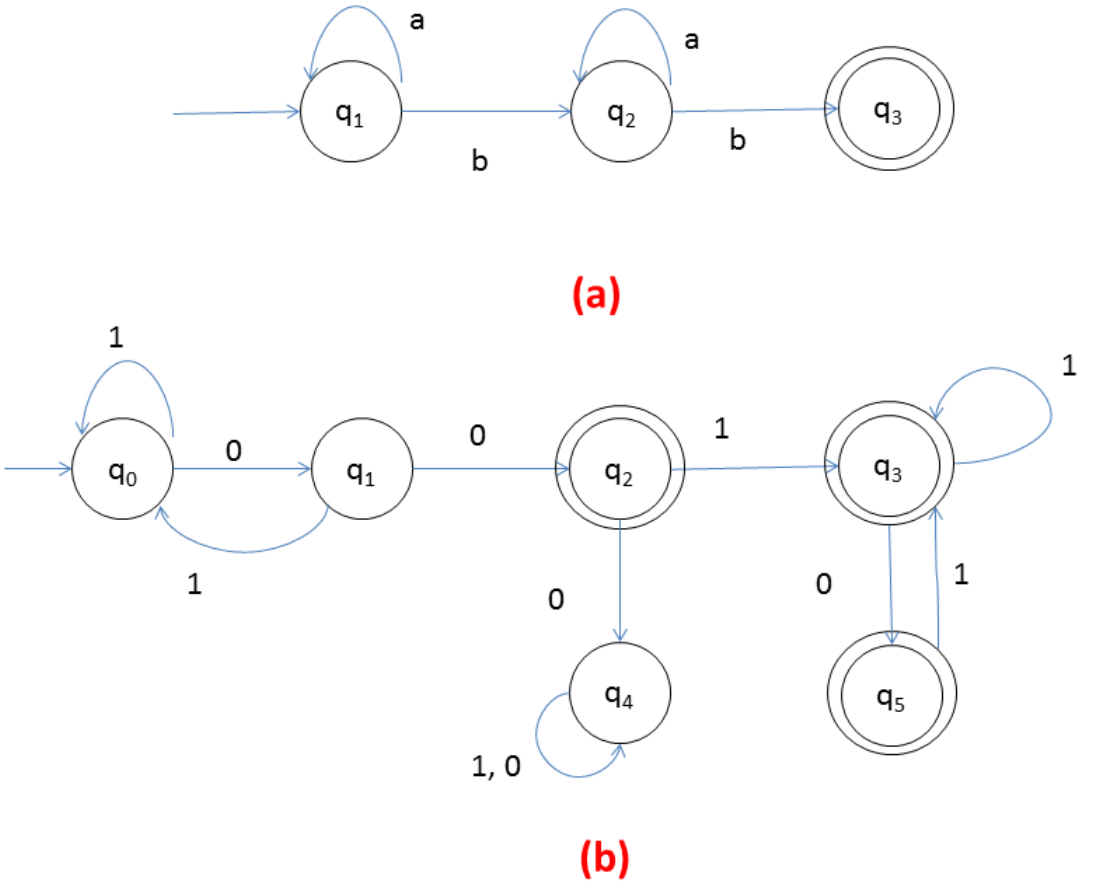
Example: Find the language accepted by following DFA:-



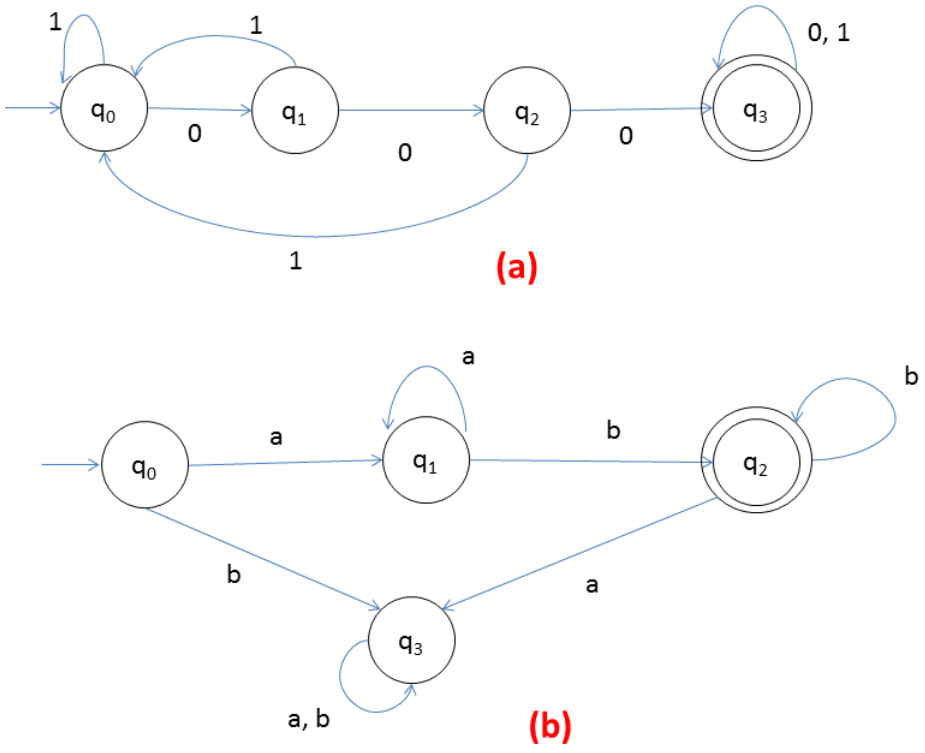
Example: Find the language accepted by following DFA:-



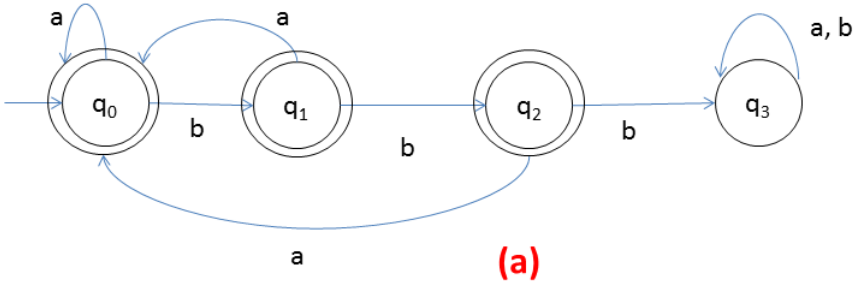
Example: Find the language accepted by following DFAs:-



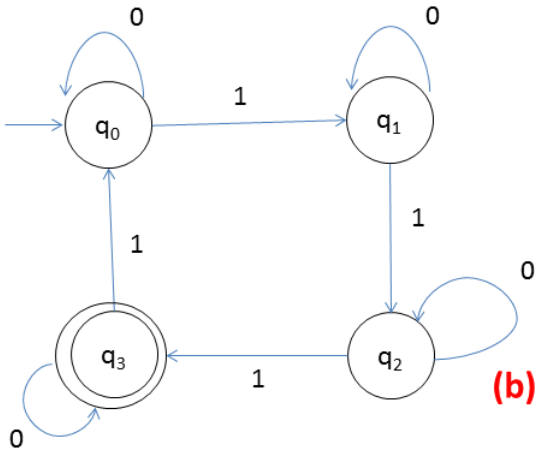
Example: Find the language accepted by following DFAs:-



Example: Find the language accepted by following DFAs:-

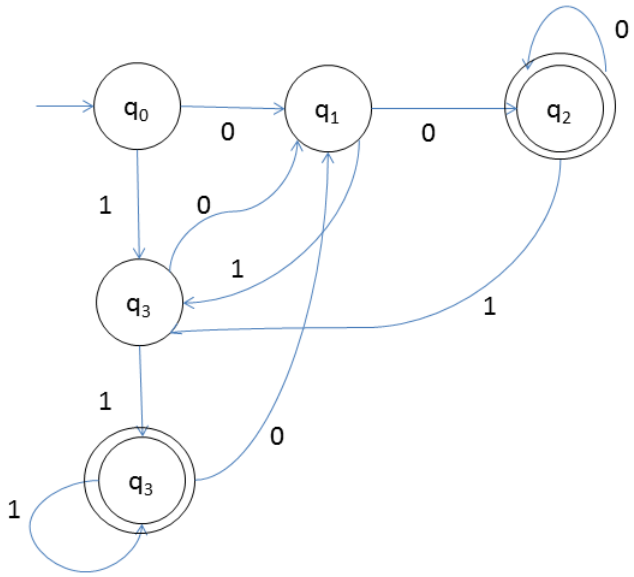


(a)



(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 = $\{a^m b \mid m \geq 0\}$

Solution:

Example: Construct DFA for the following language:-

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

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

is regular.

Solution:

Example: If L = $\{awa \mid w \in \{a, b\}^*\}$, 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 Σ = {a,b}, construct DFA's that accept the sets consisting of

(a) L ={w ! | w | mod 3 = 0}

(b) L ={w ! | w | 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 ! | $n_a(w) - n_b(w)$ | 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:-

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

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

$\delta(q,a) \subseteq Q, \qquad \forall q \in Q \; and \; a \in \Sigma$

Extended Transition Function
It is denoted by $\hat{\delta}$. It is defined as following:-
$\hat{\delta}$: QxΣ* → P(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. $\hat{\delta}(q,wa) = \bigcup_{p \in \hat{\delta}(q,w)} \delta(p,a), \qquad$ 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:-
$\hat{\delta}$: P(Q)xΣ* → P(Q)

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

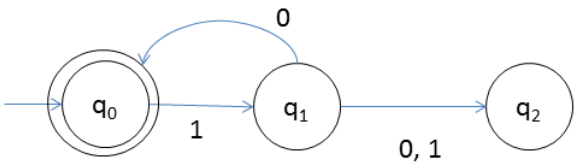
1. $\hat{\hat{\delta}}(P,a) = \bigcup_{p \in P} \hat{\delta}(p,a)$
2. $\hat{\hat{\delta}}(P,w) = \bigcup_{p \in P} \hat{\delta}(p,w), \qquad$ where $P \subseteq Q,\ a \in \Sigma \; and \; w \in \Sigma^*$

3.3.2 Language Accepted by NFA

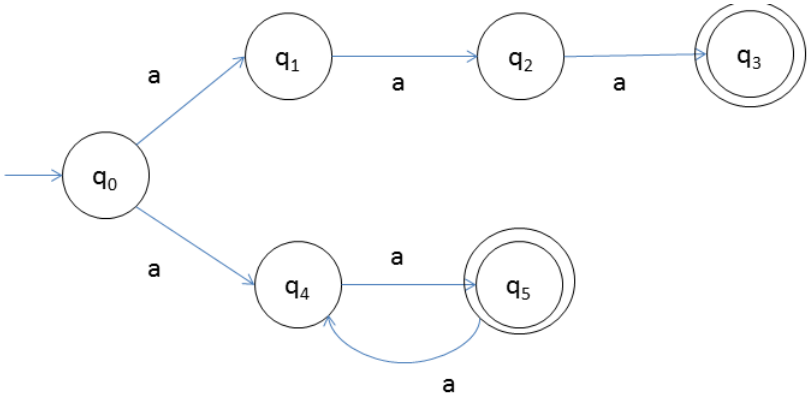
Language accepted by NFA M is denoted by L(M). It is defined as following:-
L(M) = {x ∈ Σ* ! $\hat{\delta}(q_0,x) \cap F \neq \phi$}

3.3.3 Some Examples

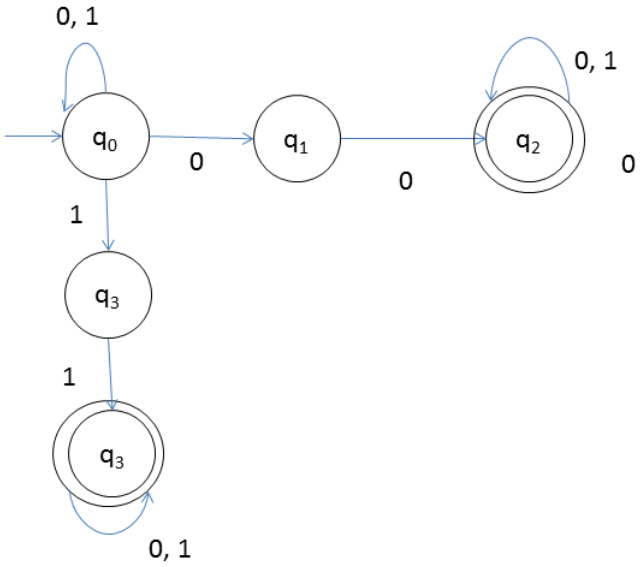
Examples: Find the language accepted by following NFA:-



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 $\{abab^n \ ! \ n{\geq}0\}{\cup}\{aba^n \ ! \ n{\geq}0\}$.

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',Σ,δ',q_0',F')

Where Q' = P(Q), q_0'={q_0}
F' = {P⊆Q ! P∩F ≠ ϕ}
and $\delta' \equiv \hat{\delta}$
i.e. $\delta'(P,a) = \hat{\delta}(P,a) = \bigcup_{p\in P} \hat{\delta}(p,a) = \bigcup_{p\in P} \delta(p,a)$
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 $|\ w\ | = 0$, 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 $|\ w\ | = 1$, i.e. w = a, where a∈ Σ
$\hat{\delta'}(q_0', a) = \delta'(q_0', a)$

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

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

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

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

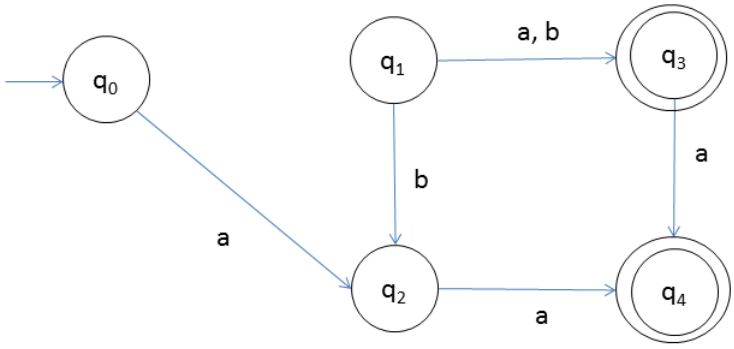
Example: Find DFA equivalent to the following NFA:-
M = ($\{q_0, q_1, q_2\}$,{a,b}, δ, q_0,$\{q_2\}$),
where δ is given by

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 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 $\mid x \mid \leq k$.

3.4.1 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}.

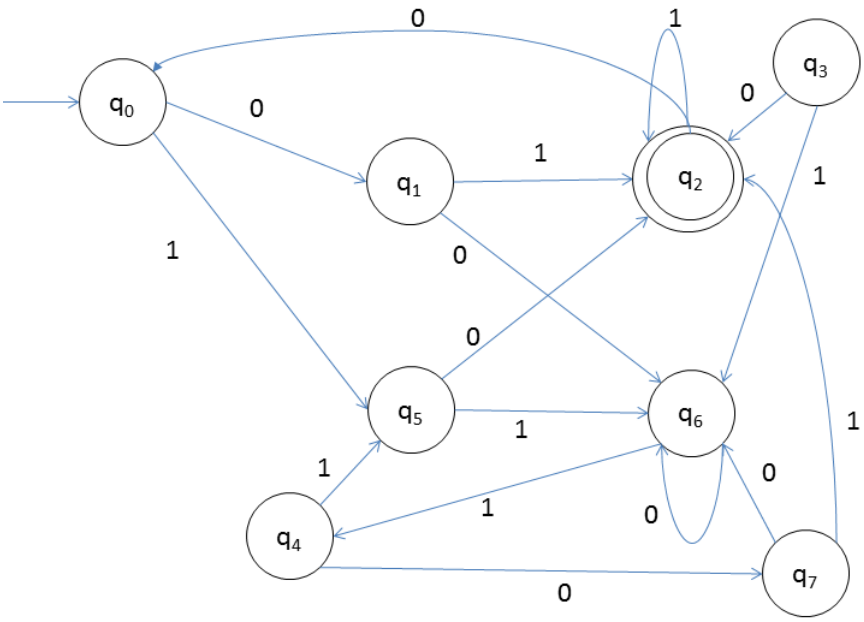
6. Similarly, we apply above procedure for all the sets belong into Π_k.

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

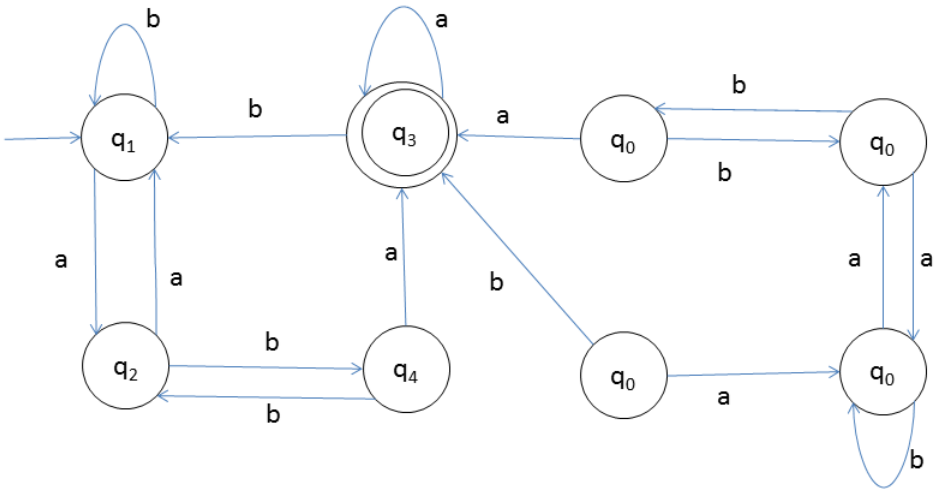
Step 4: For the required minimum state automata, the states are the equivalence classes obtained in step 3 i.e. the elements of Π_n. The state table is obtained by replacing a state q by the corresponding equivalence class [q].

3.4.2 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
→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

3.5 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→ 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 : QxΣ → Q
$\lambda \rightarrow$ Output function
It is defined as following:-
$\lambda : Q \rightarrow \Delta$

Mealy Machine

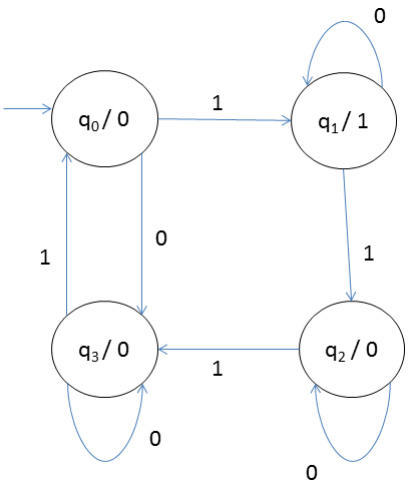
Mealy machine is a six-tuple $(Q, \Sigma, \Delta, \delta, \lambda, q_0)$, where
Q→ Finite set of states
$\Sigma \rightarrow$ Finite set of input symbols
$\Delta \rightarrow$ Finite set of output symbols
$q_0 \in Q \rightarrow$ Initial state
$\delta \rightarrow$ Transition function
It is defined as following:-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

3.5.1 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	
→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



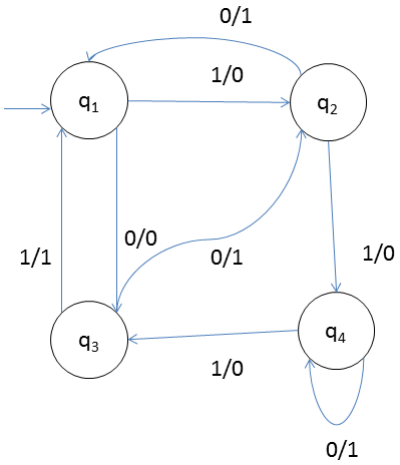
Example: Find the output string corresponding to the input string 0111 in the above Moore machine.
$q_0 \rightarrow q_3 \rightarrow q_0 \rightarrow q_1 \rightarrow q_2$
The output string is 00010.

3.5.2 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
→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.

3.5.3 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

3.5.4 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

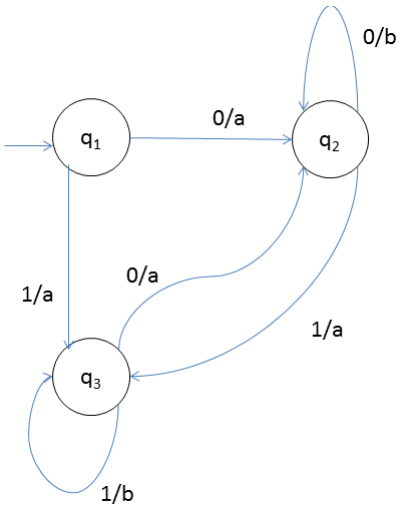
Solution: In the first step we split the some states. We split those states that has different outputs. In this example, we split states q_2 and q_4.

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.

3.6 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:-

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

$\epsilon - closure(q)$
$\epsilon - closure$ 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:-
$\hat{\delta}$: QxΣ* → 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)xΣ* → P(Q)

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

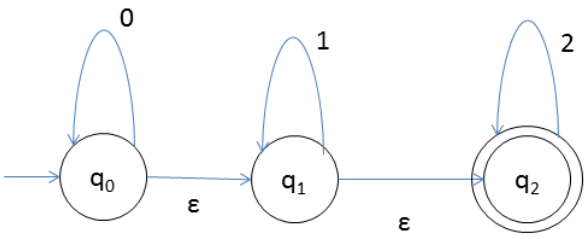
1. $\hat{\hat{\delta}}(P, a) = \underset{p\in P}{\cup} \hat{\delta}(p, a)$
2. $\hat{\hat{\delta}}(P, w) = \underset{p\in P}{\cup} \hat{\delta}(p, w)$, where $P \subseteq Q,\ a \in \Sigma$ *and* $w \in \Sigma^*$

3.6.1 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* ! $\hat{\delta}(q_0, x) \cap F \neq \phi$}

3.6.2 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.

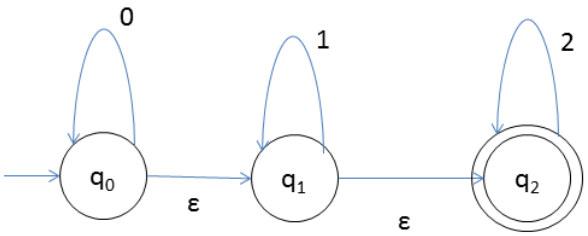
3.6.3 Conversion of NFA with $\epsilon-move$ into NFA without $\epsilon-move$

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

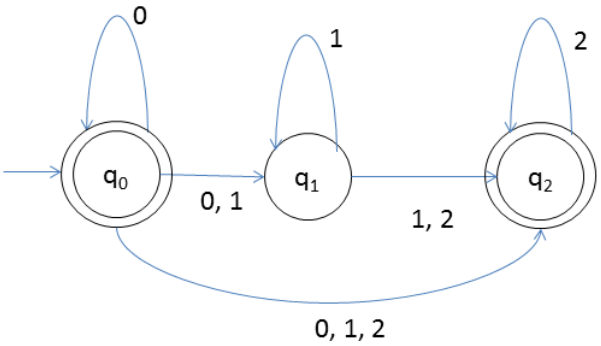
$$F' = \begin{cases} F \cup \{q_0\} \ , & if \ \ \epsilon - closure(q_0) \cap F \neq \phi \\ F \ \ , & otherwise \end{cases} \tag{3.1}$$

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

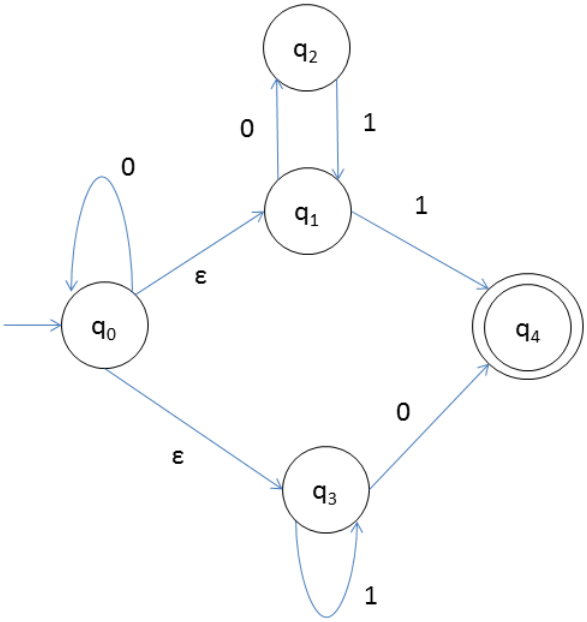
Examples: Consider the following NFA with $\epsilon - move$:-



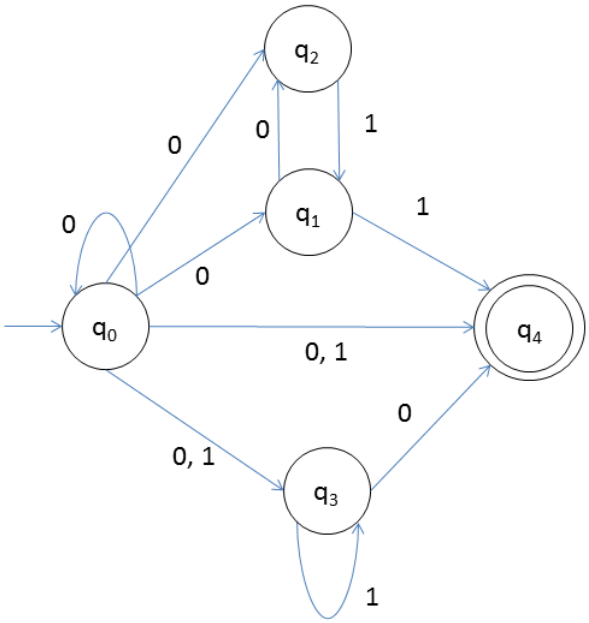
Find NFA without $\epsilon - move$ equivalent to this.
Solution:



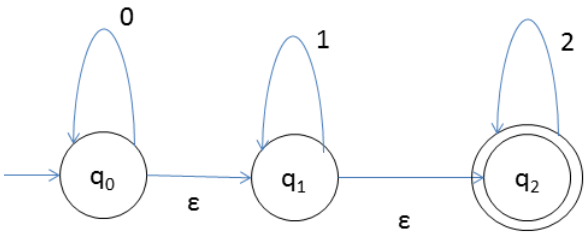
Examples: Consider the following NFA with $\epsilon - move$:-



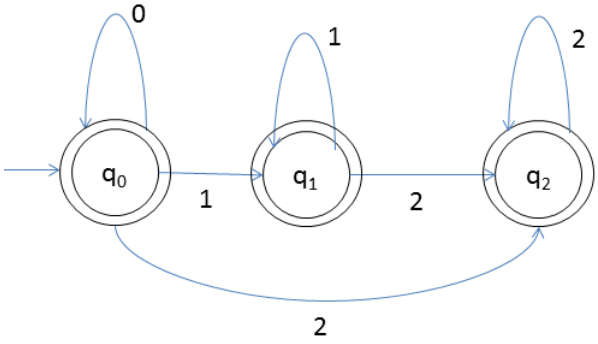
Find NFA without $\epsilon - move$ equivalent to this.
Solution:



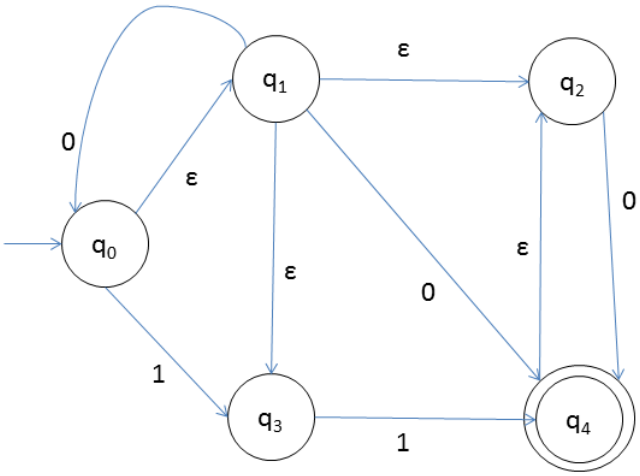
Examples: Consider the following NFA with $\epsilon - move$:-



Construct DFA equivalent to this.
Solution:



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



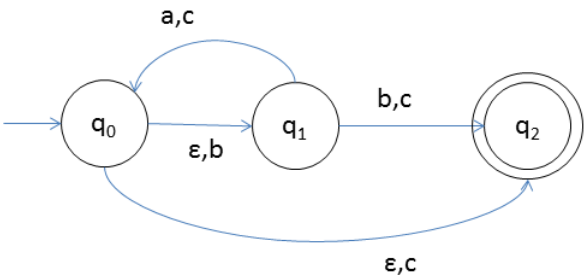
Construct DFA equivalent to this.

3.7 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:-



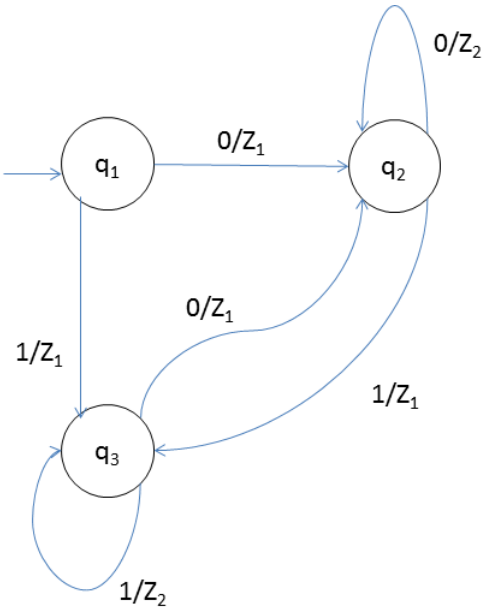
4. Design FA for ternary number divisible by 5.

5. Explain Myhill-Nerode Theorem using suitable example.

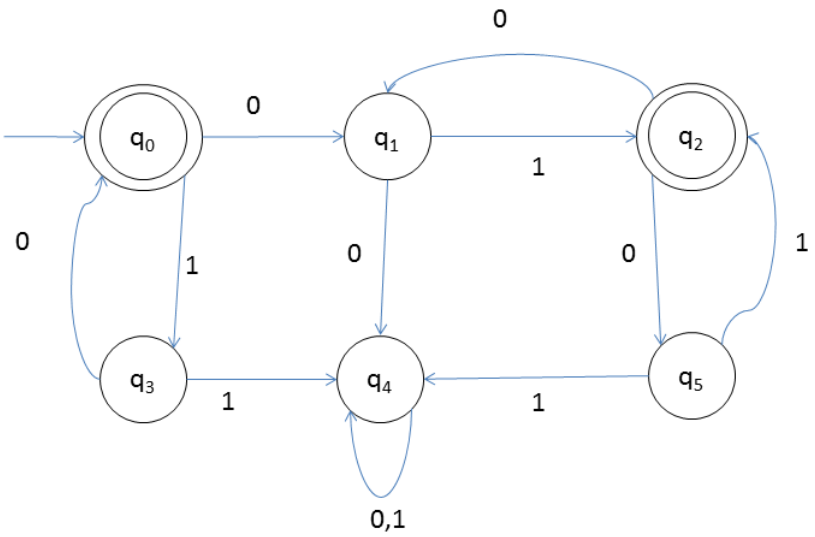
6. Design a NFA that accepts all the strings for input alphabet {a,b} containing the substring abba.

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.

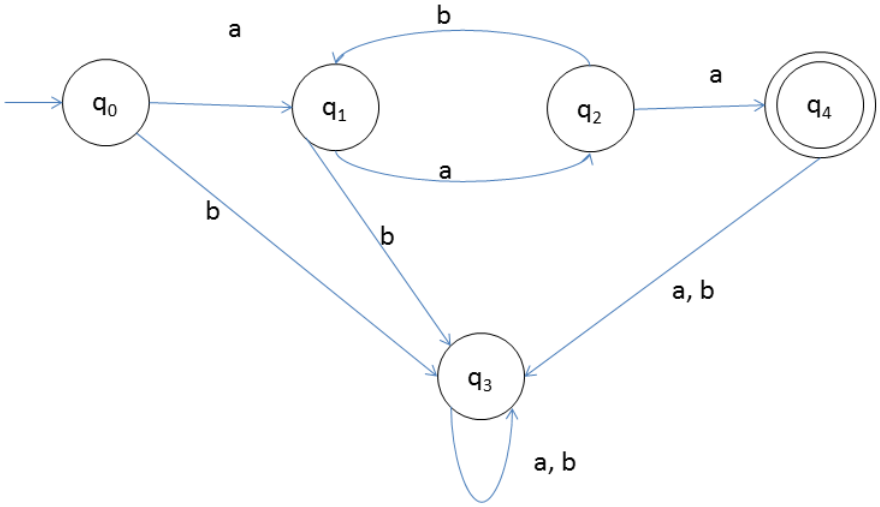
17. How do we determine equivalence of two DFA? Explain with an example.

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.

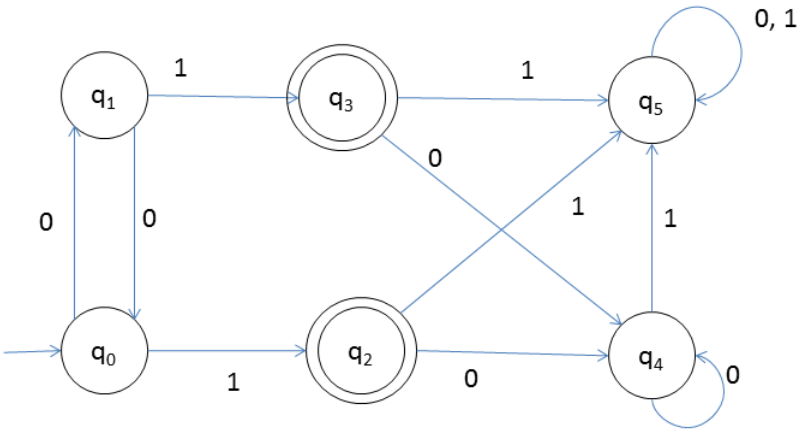
19. Design an NFA for the language containing strings ending with ab or ba. Also convert the obtained NFA into equivalent DFA.

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

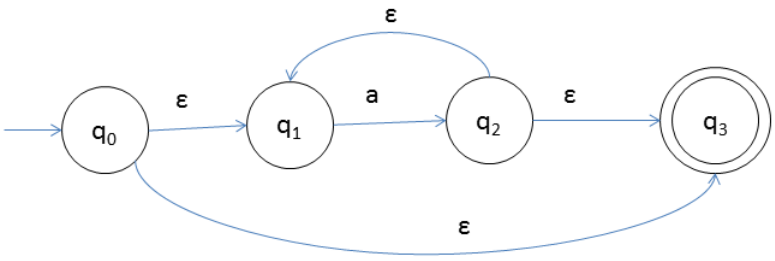
21. Consider the DFA given below and identify the L accepted by the machine.



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.

Chapter 4

Regular Expression and Regular Languages

4.1 Regular expression

4.1.1 Definition

Regular expression is defined recursively as follows:-
(1) ϕ, $\bar{\epsilon}$ and $\bar{a}$ are regular expression, where $a \in \Sigma$. These are called primitives regular expressions.
(2) If $\bar{r_1}$ and $\bar{r_2}$ are two regular expressions then $\bar{r_1} + \bar{r_2}$ and $\bar{r_1}\bar{r_2}$ are also regular expressions.
(3) If $\bar{r}$ are regular expression then $\bar{r^*}$ and $(\bar{r})$ are also regular expressions.
(4) Any expression derived from the step 1 to 3 are also regular expression.

Example: $(\bar{a} + \bar{b} + \bar{c})^*.(\bar{c}+\phi)$ is a regular expression.

4.1.2 Regular language or Regular set

Each regular expression represents a set of elements. This set is said to be regular set.

4.1.3 Some examples

1. Represent the following sets by regular expression:-

   (a) $\{1^{2n} \mid n > 0\}$
   (b) $\{w \in \{a,b\}^* \mid$ w has only one a$\}$
   (c) The set of all strings over {0,1} which has most two 0's.
   (d) $\{a^2, a^5, a^8, ...\}$
   (e) $\{1^n \mid$ n is divisible by 2 or 3 or n=5$\}$
   (f) The set of all strings over {a,b} beginning and ending with a.

(g) The set of all strings over {a,b} in which the number of occurrences of a is divisible by 3.

(h) The set of all strings over {a,b} with three consecutive b's.

2. Find the regular expression for the following:-

(a) L = $\{a^n b^m \mid n \geq 1, m \geq 1, mn \geq 3\}$

(b) L = $\{a^n \mid n \geq 0, n \neq 3\}$

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

(d) L = $\{a^n b^m \mid$ n+m is even$\}$

3. Write regular expression for the following language over alphabet {0,1}.

(a) All strings ending in 01.

(b) All strings not ending in 01.

(c) All strings containing an even number of 0's.

(d) All strings with at most two occurrences of the substring 00.

(e) All strings not containing the substring 10.

4. Write regular expression for the following language.

(a) L = {w$\in \{0,1\}^*$! w has at least one pair of consecutive 0's }

(b) L = {w$\in \{0,1\}^*$! w has no pair of consecutive 0's }

(c) L = {w ! w$\in \{a,b\}^*$ and $\mid w \mid \ mod\ 3 = 0$}

(d) L = {w ! w$\in \{a,b\}^*$ and $n_a(w)\ mod\ 3 = 0$}

5. Find the set corresponding to the following regular expressions:-

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

(b) $(0+1)^*00(0+1)^*$

(c) $((0+1)(0+1)^*)^*00(0+1)^*$

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

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

4.1.4 Equivalent regular expressions

Two regular expressions $\bar{p}$ and $\bar{q}$ are said to be equivalent if they represent the same set of strings.

4.1.5 Identities for regular expressions

Let p, q, r are all the regular expressions. Then

(1) ϕ + r = r
(2) ϕ . r = ϕ = r.ϕ
(3) ϵ.r = r.ϵ = r
(4) $\epsilon^* = \epsilon$ and $\phi^* = \epsilon$ (5) r+r = r
(6) $r^*r^* = r^*$
(7) $rr^* = r^*r$
(8) $(r^*)^* = r^*$
(9) $\epsilon + rr^* = r^* = \epsilon + r^*r$
(10) $(p.q)^*p = p.(p.q)^*$
(11) $(p+q)^* = (p^*q^*)^* = (p^* + q^*)^*$
(12) $(p+q).r = p.r + q.r$ and $r.(p+q). = r.p + r.q$

4.2 ARDEN's Theorem

Statement: Let P and Q be two regular expressions over Σ. If P does not contain ϵ , then the following equation in R,

R = Q + RP .....(1)

has a unique solution R = QP^*.

Proof:

Q+RP = Q + QP*P
= Q($\epsilon + P^*P$)
= QP*
= R

Therefore equation (1) is satisfied when R = QP*.
Therefore R = QP* is a solution of equation (1).
To prove uniqueness of (1), replacing R by Q+RP.

R = Q +RP = Q + (Q+RP)P
= Q + QP + RP^2
= Q + QP + $(Q + RP)P^2$
= Q + QP + QP^2 + RP^3
.....
.....
= Q + QP +QP^2 +.....+ QP^i + RP^{i+1}
= Q(ϵ + P +P^2 +.....+ P^i) + RP^{i+1}

Therefore, R = Q(ϵ + P +P^2 +.....+ P^i) + RP^{i+1} .....(2)
Let w$\in$ R and $| w |$ = i.
From equation (2), w will belong into right hand side of equation (2).
Therefore, w$\in$ Q(ϵ + P +P^2 +.....+ P^i) + RP^{i+1}
Clearly, w$\notin$ RP^{i+1} since $| w |$ = i.
Therefore, w $\in$ Q(ϵ + P +P^2 +.....+ P^i)
Therefore, w $\in$ QP^*
Therefore, R$\subseteq$ QP^* .....(3)
Let w $\in$ QP^*.
Then w$\in$$QP^k$ for some k$\geq$ 0.

Therefore, w $\in$ Q(ϵ + P +P^2 +.....+ P^k)
Therefore, w belong into the right hand side of equation (2).
Therefore, w$\in$ R.
Therefore, $QP^* \subseteq$ R .....(4)
From (3) and (4),
R = QP^*

Example:
(a) Give a regular expression for representing the set L of strings in which every 0 is immediately followed by at least two 1's.
(b) Prove that the regular expression
r = $\epsilon + 1^*(011)^*(1^*(011)^*)^*$
also describes the same set of strings.

Example: Prove that
$(1 + 00^*1) + (1 + 00^*1)(0 + 10^*1)^*(0 + 10^*1) = 0^*1(0 + 10^*1)^*$

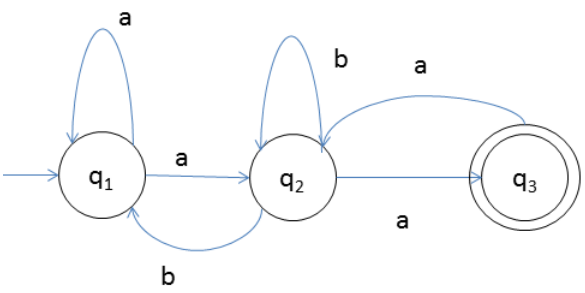
4.3 Determination of regular expression from finite automata

The following assumptions are made regarding the transition system:-
(i) The transition graph does not contain $\epsilon - move$.
(ii) It has only one initial state, say q_1.
(iii) Let all the states are q_1, q_2, q_3,.....,q_n,
(iv) Let α_{ij} denotes the regular expression representing the set of labels of edges from q_i to q_j. When there is no such edge, $\alpha_{ij} = \phi$.

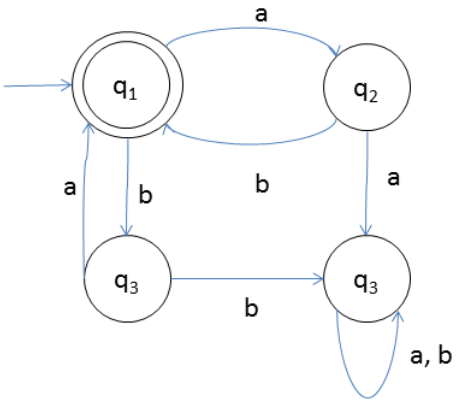
In this process to find regular expression, initially we make n equations as the following:-
$q_1 = q_1\alpha_{11} + q_2\alpha_{21} + q_3\alpha_{31} +$ .....+ $q_n\alpha_{n1} + \epsilon$
$q_2 = q_1\alpha_{12} + q_2\alpha_{22} + q_3\alpha_{32} +$ .....+ $q_n\alpha_{n2}$
.....
$q_n = q_1\alpha_{1n} + q_2\alpha_{2n} + q_3\alpha_{3n} +$ .....+ $q_n\alpha_{nn}$
We solve these equations by using ARDEN's theorem . The regular expression will be the union of regular expressions corresponding to each final states.

4.3.1 Some Examples

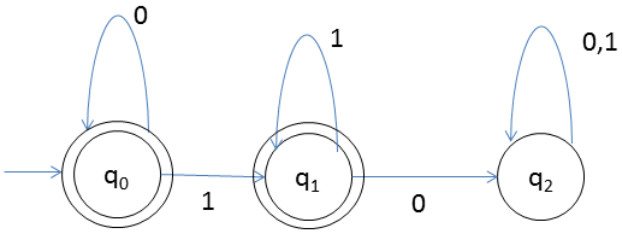
Example: Find the regular expression corresponding to the following finite automata:-



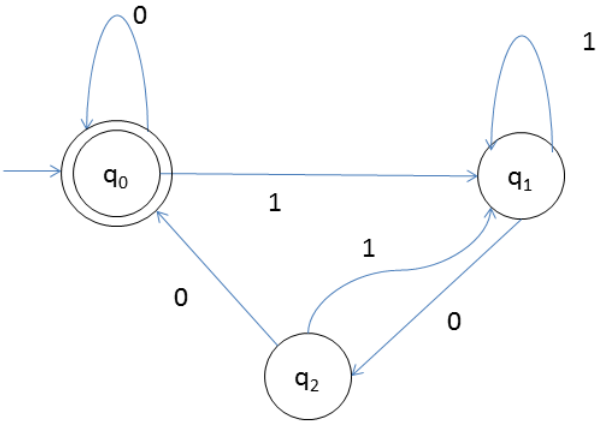
Example: Find the regular expression corresponding to the following finite automata:-



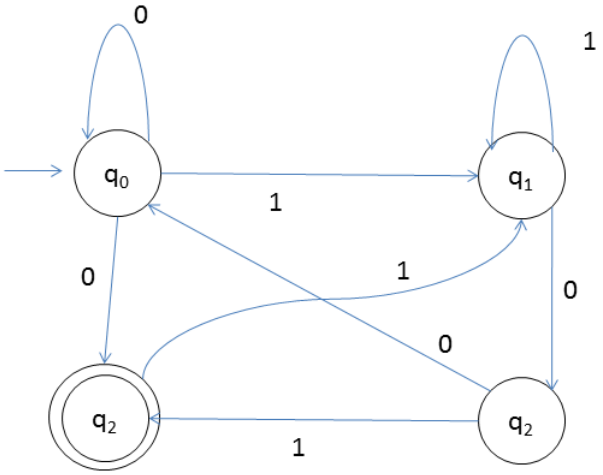
Example: Find the regular expression corresponding to the following finite automata:-



Example: Find the regular expression corresponding to the following finite automata:-

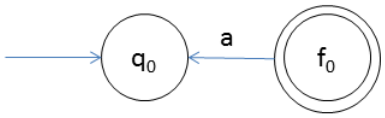


Example: Find the regular expression corresponding to the following finite automata:-



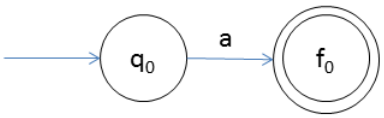
4.4 Kleene's Theorem

Statement: Let r be a regular expression. Then there exists some non-deterministic finite automata that accepts L(r). Consequently, L(r) is a regular language.
Proof:
We will prove this theorem by using induction method.
Apply the induction method on the no. of operators used in the regular expression.i **Basis step:** In this step, we will prove for the regular expressions which does not contain any operators.
In this case, regular expressions are
ϕ, ϵ and a, where a$\in \Sigma$.
The finite automata corresponding to the above regular expressions are:-



L(M) = ϕ

L(M) = { ε }



L(M) = { a }

Clearly, there exists a finite automata for every regular expressions which not contain any operator. Therefore, theorem is true for this case.

Induction step:

Suppose the theorem is true for the regular expressions which contain n operators. We will prove the theorem for the regular expressions that contain n+1 operators. The $(n+1)^{th}$ operator in the regular expressions may be one of the following:-

(1) + (Union operator) (2) . (Concatenation operator) (3) * (Kleene closure operator)

Case 1 Union operator($r_1 + r_2$):

Suppose the $(n+1)^{th}$ operator is $+$.

Consider the regular expressions r_1 and r_2 of n operators.

Since the theorem is true for r_1 and r_2, therefore there exists finite automata M_1 and M_2 corresponding to r_1 and r_2. Let these finite automatas are

$M_1 = (Q_1, \Sigma_1, \delta_1, q_1, \{f_1\})$

$M_2 = (Q_2, \Sigma_2, \delta_2, q_2, \{f_2\})$

Now, we construct the finite automata corresponding to $r_1 + r_2$. This is constructed as following:-

$M = (Q, \Sigma, \delta, q_0, \{f_0\})$

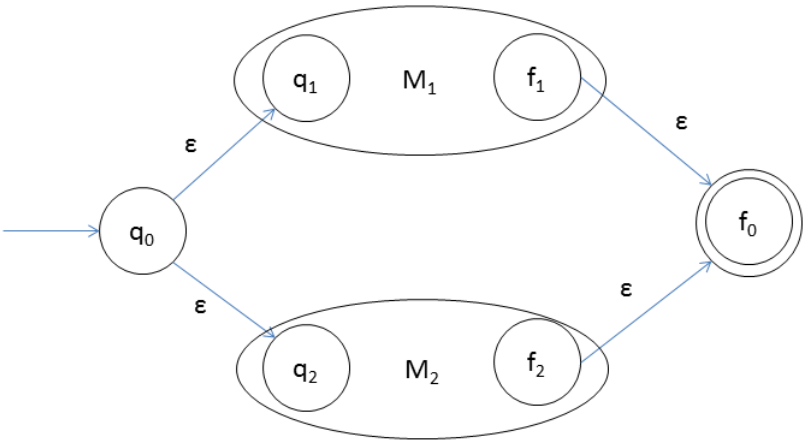
Where, Q $= Q_1 \cup Q_2 \cup \{q_0, f_0\}$

$\Sigma = \Sigma_1 \cup \Sigma_2$

$$\delta(q,a) = \begin{cases} \delta_1(q,a), & if q \in Q_1 \text{ and } a \in \Sigma_1 \\ \delta_2(q,a), & if q \in Q_2 \text{ and } a \in \Sigma_2 \end{cases} \tag{4.1}$$

$\delta(q_0, \epsilon) = \{q_1, q_2\}$

$\delta(f_1, \epsilon) = \{f_0\} = \delta(f_2, \epsilon)$



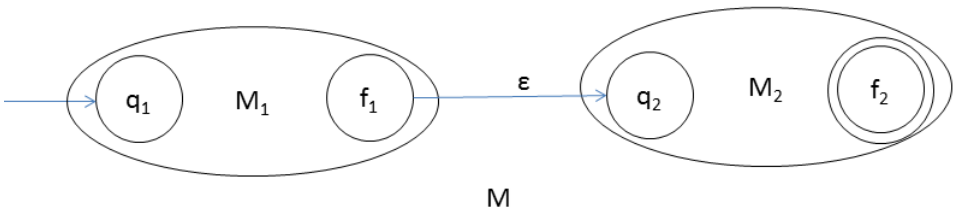
M

Now,we have to show that
L(M) = $L(M_1) \cup L(M_2)$ .....(1)
Let x∈ L(M) ⇔ $\epsilon x \epsilon \in L(M)$
⇔ x∈ L(M_1) or x∈ L(M_2)
⇔ x∈ $L(M_1) \cup \in L(M_2)$
Therefore, L(M) = $L(M_1) \cup L(M_2)$
Therefore M is a finite automata which accepts the set $L(r_1) \cup L(r_2)$.
Therefore M is a finite automata for the regular expression $r_1 + r_2$.
Therefore theorem is true for this case.

Case 2 Concatenation operator($r_1.r_2$):
Suppose the $(n+1)^{th}$ operator is . .
Consider the regular expressions r_1 and r_2 of n operators.
Since the theorem is true for r_1 and r_2, therefore there exists finite automata M_1 and M_2 corresponding to r_1 and r_2. Let these finite automatas are
$M_1 = (Q_1, \Sigma_1, \delta_1, q_1, \{f_1\})$
$M_2 = (Q_2, \Sigma_2, \delta_2, q_2, \{f_2\})$
Now, we construct the finite automata corresponding to $r_1.r_2$. This is constructed as following:-
$M = (Q, \Sigma, \delta, q_1, \{f_2\})$
Where, Q = $Q_1 \cup Q_2$
$\Sigma = \Sigma_1 \cup \Sigma_2$

$$\delta(q,a) = \begin{cases} \delta_1(q,a), & if q \in Q_1 \text{ and } a \in \Sigma_1 \\ \delta_2(q,a), & if q \in Q_2 \text{ and } a \in \Sigma_2 \end{cases} \tag{4.2}$$

$\delta(f_1, \epsilon) = \{q_2\}$

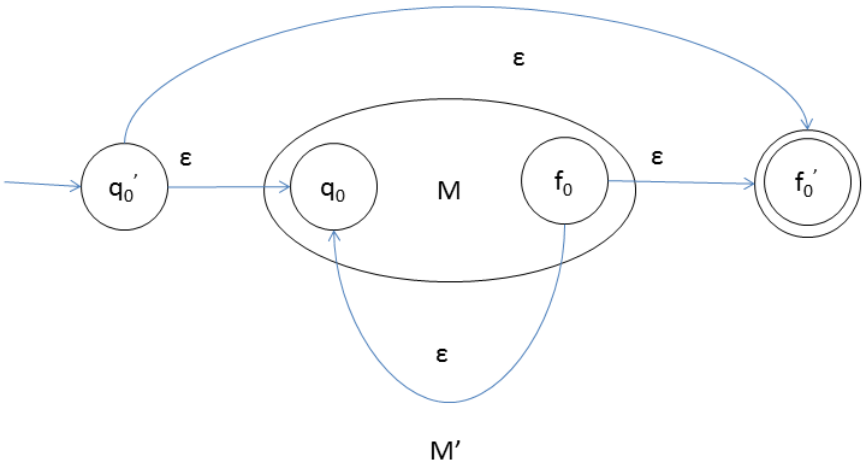


Now,we have to show that
L(M) = $L(M_1).L(M_2)$ .....(2)
Let x∈ L(M) ⇔ $x_1 \epsilon x_2 \in L(M)$ (Let x=$x_1.x_2$)
⇔ $x_1 \in$ L(M_1) and $x_2 \in$ L(M_2)
⇔ $x_1.x_2 \in L(M_1).L(M_2)$
⇔ x ∈ $L(M_1).L(M_2)$
Therefore, L(M) = $L(M_1).L(M_2)$
Therefore M is a finite automata which accepts the set $L(r_1).L(r_2)$.
Therefore M is a finite automata for the regular expression $r_1.r_2$.
Therefore theorem is true for this case.

Case 3 Kleene closure operator(r^*):
Suppose M is a finite automata corresponding to regular epression r.
$M = (Q, \Sigma, \delta, q_0, \{f_0\})$
Now, we construct finite automata corresponding to r^*. This is constructed as following:-
$M' = (Q', \Sigma, \delta', q_0', \{f_0'\})$
Where, Q = $Q \cup \{q_0', f_0'\}$

$$\delta'(q,a) = \begin{cases} \delta(q,a), & if q \in Q \text{ and } a \in \Sigma \end{cases} \tag{4.3}$$

$\delta'(q_0', \epsilon) = \{q_0, f_0'\} = \delta'(f_0, \epsilon)$



Now,we have to show that
L(M') = $(L(M))^*$ .....(3)

Let x$\in$ L(M') $\Leftrightarrow$ $\epsilon x \epsilon \in L(M')$ (Let x=$x_1.x_2$)
$\Leftrightarrow$ $\epsilon x_1 \epsilon x_2 \epsilon .....\epsilon x_n \epsilon \in L(M')$ (Let x=$x_1.x_2.....x_n$)
$\Leftrightarrow$ $x_i \in L(M), \quad \forall i = 1, 2, ..., n$
$\Leftrightarrow$ $x_1.x_2.....x_n \in (L(M))^n$
$\Leftrightarrow$ $x \in (L(M))^n$
$\Leftrightarrow$ $x \in (L(M))^*$ (since $(L(M))^n) \subseteq (L(M))^*$
Therefore, L(M) = $(L(M))^*$
Therefore M is a finite automata which accepts the set $L(r_1).L(r_2)$.
Therefore M is a finite automata for the regular expression $r_1.r_2$.
Therefore theorem is true for this case.

Since in all the three cases, theorem is true, therefore theorem is true for induction step also.
Therefore, for any regular expression, there exists a finite automata.
Now the theorem is proved.

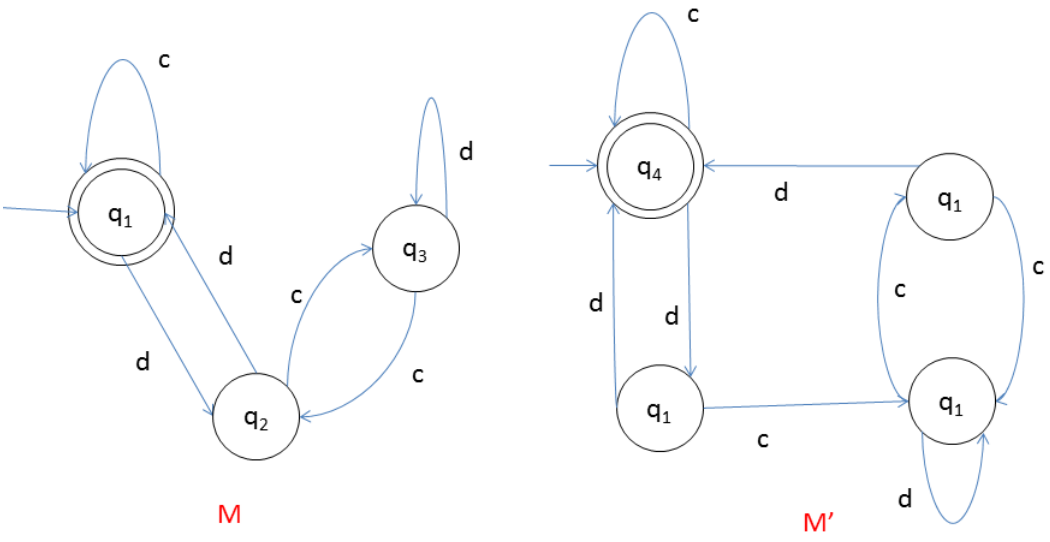
4.5 Construction of finite automata from regular expression

Example: Construct finite automata for the regular expressions:-

1. r = $(a+b)^*(aa+bb)(a+b)^*$
2. r = $10+(0+11)0^*1$
3. r = $(a+b)^*b(a+bb)^*$
4. r = $aa^*+aba^*b^*$

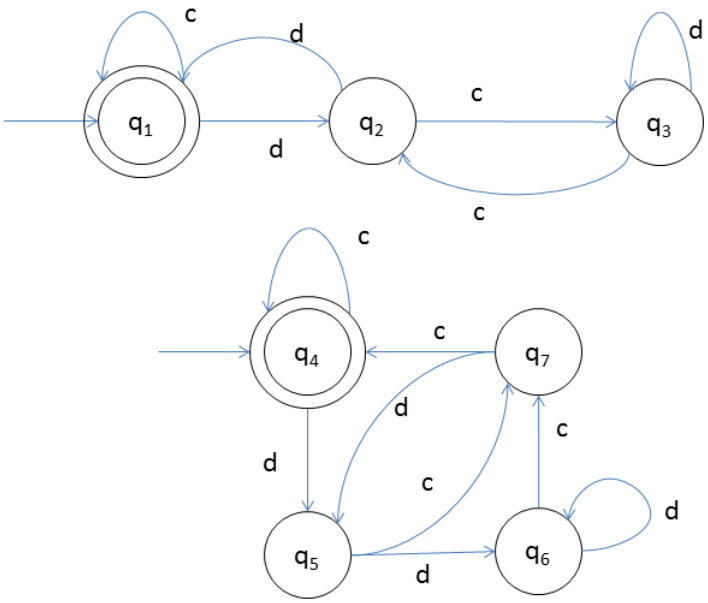
4.6 Equivalence of two finite automata

Example: Consider two DFA M and M':



Determine whether M and M' are equivalent.

Example: Show that following automata M_1 and M_2 are not equivalent.



4.7 Right and Left linear grammars

A grammar is said to be right linear grammar if all production rules are of the following form:-

A $\rightarrow$ xB or A $\rightarrow$ x, where A,B $\in$ V and x$\in \Sigma^*$

A grammar is said to be left linear grammar if all production rules are of the following form:-

A $\rightarrow$ Bx or A $\rightarrow$ x, where A,B $\in$ V and x$\in \Sigma^*$

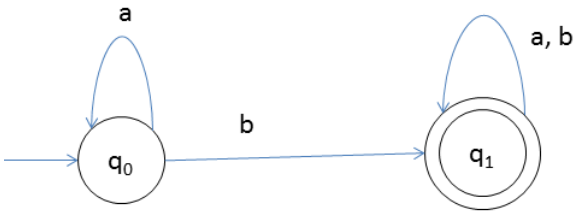
A regular grammar is one that is either right linear or left linear.

4.7.1 Construction of regular grammar from the given DFA

Suppose the given DFA is
$M = (\{q_0, q_1, ..., q_n\}, \Sigma, \delta, q_0, F)$
Now we construct the grammar G for M as
G = $(\{Q_0, Q_1, ..., Q_n\}, \Sigma, Q_0, P)$

Where P is defined as
(i) $Q_i \rightarrow aQ_j \in P$ if $\delta(q_i, a) = q_j \notin F$
(ii) $Q_i \rightarrow aQ_j$ *and* $Q_i \rightarrow a \in P$ if $\delta(q_i, a) = q_j \in F$

Example: Find the regular grammar for the following DFA



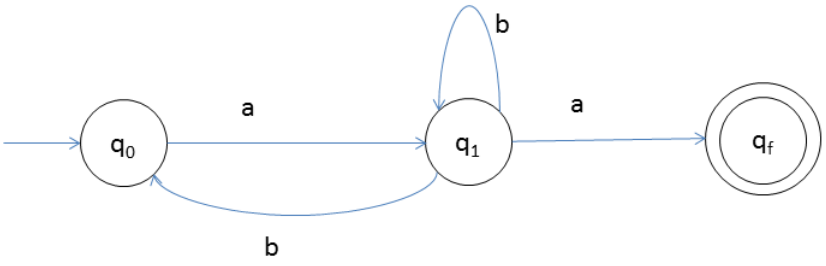
Solution: Since the number of states are 2, therefore number of variables in the grammar will be 2. Let these variables are Q_0 and Q_1 corresponding to states q_0 and q_1. The starting symbol will be Q_0.
The production rules of the grammar are the following:-
$Q_0 \rightarrow aQ_0$
$Q_0 \rightarrow b/bQ_1$
$Q_1 \rightarrow a/b/aQ_1/bQ_1$

4.7.2 Construction of a FA from given regular grammar

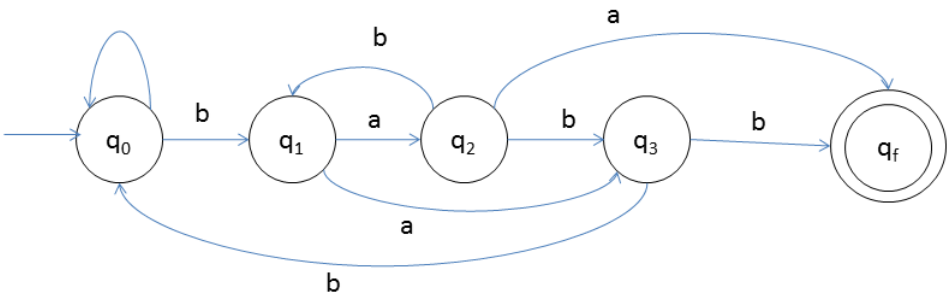
G = $(\{A_0, A_1, ..., A_n\}, \Sigma, A_0, P)$
We construct finite automata M as
$M = (\{q_0, q_1, ..., q_n, q_f\}, \Sigma, \delta, q_0, \{q_f\})$

and δ is defined as
(i) If $A_i \rightarrow aA_j$ then $\delta(q_i, a) = q_j$
(i) If $A_i \rightarrow a$ then $\delta(q_i, a) = q_f$

Example: Let G = $(\{A_0, A_1\}, \{a, b\}, A_0, P)$
Where P is
$A_0 \rightarrow aA_1$, $A_1 \rightarrow bA_1$, $A_1 \rightarrow a$, $A_1 \rightarrow bA_0$,
Construct finite automata accepting L(G).
Solution:



Example: Let G = $(\{A_0, A_1, A_2, A_3\}, \{a, b\}, A_0, P)$
Where P is
$A_0 \rightarrow aA_0/bA_1$, $A_1 \rightarrow aA_2/aA_3$, $A_2 \rightarrow a/bA_1/bA_3$, $A_3 \rightarrow b/bA_0$,
Construct finite automata accepting L(G).
Solution:



4.8 Pumping Lemma

Let L be an infinite regular language. Then there exists some positive integer n such that any x∈ L with x≥n, can be decomposed as

x=uvw

with $| uv | \leq n$ and $| v | \geq 1$,
such that
$uv^i w \in$ L, ∀ i = 0,1,2,3,.....

4.8.1 Application of Pumping Lemma

It is used to show that a language is not regular.
Note: Pumping lemma is based on the principle of pigeonhole principle.
Pigeonhole Principle
If we put n objects into m boxes and n>m, then at least one box must have more than one item in it.

4.8.2 Some Examples

1. Show that L = $\{a^{i^2} \mid i \geq 1\}$ is not regular.
2. Show that L = $\{a^n b^n \mid i \geq 1\}$ is not regular.
3. Show that L = $\{ww^R \mid w \in \{a,b\}^*\}$ is not regular.
4. Show that L = $\{a^p \mid$ p is a prime number $\}$ is not regular.
5. Show that L = $\{a^{n!} \mid n \geq 0\}$ is not regular.
6. Show that L = $\{(ab)^n a^k \mid n > k,\ k \geq 0\}$ is not regular.

4.9 Closure properties of regular languages

Theorem: Show that class of regular languages is closed under union operation.

or

If L_1 and L_2 are two regular set then $L_1 \cup L_2$ is also regular set.

Theorem: Show that class of regular languages is closed under concatenation operation.

or

If L_1 and L_2 are two regular set then $L_1.L_2$ is also regular set.

Theorem: Show that class of regular languages is closed under Kleene closure operation.

or

If L is regular set then L^* is also regular set.

Theorem: Show that class of regular languages is closed under complement operation.

or

If L is regular set then $\bar{L}$ is also regular set.

Proof:

Suppose L is a regular set. Since L is regular set then there exists a finite automata. Let this finite automata is

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

Now we construct the finite automata M' as following:-

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

Where F' = Q-F

Now we have to show that L(M') = $\bar{L}(M) = \Sigma^* - L(M)$

Let x$\in L(M') \Leftrightarrow \delta(q_0, x) \in F'$

$\Leftrightarrow \delta(q_0, x) \in Q - F$

$\Leftrightarrow \delta(q_0, x) \notin F$

$\Leftrightarrow x \notin L(M)$

$\Leftrightarrow x \in \bar{L}(M)$

Therefore L(M') = $\bar{L}(M)$

Clearly M' is a finite automata accepting $\bar{L}$. Therefore $\bar{L}$ is a regular set.

Theorem: Show that class of regular languages is closed under intersection operation.

or

If L_1 and L_2 are two regular set then $L_1 \cap L_2$ is also regular set.

Proof:

$L_1 \cap L_2 = (\bar{L_1} \cup \bar{L_1}) = \Sigma^* - ((\Sigma^* - L_1) \cup (\Sigma^* - L_2))$ .....(1)

Since L_1 and L_2 are two regular set therefore $(\Sigma^* - L_1)$ and $(\Sigma^* - L_1)$ are also regular sets according to the previous theorem.

Since $(\Sigma^* - L_1)$ and $(\Sigma^* - L_1)$ is regular therefore $(\Sigma^* - L_1) \cup (\Sigma^* - L_1)$ is also regular

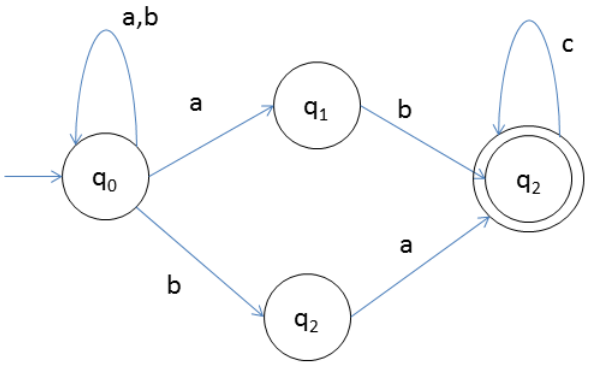
set.
According to the previous theorem, $\Sigma^* - ((\Sigma^* - L_1) \cup (\Sigma^* - L_2))$ is also regular.
Now from equation (1), $L_1 \cap L_2$ is also regular set. Now it is proved.

4.10 AKTU Examination Questions

1. For the given language $L_1 = \epsilon$, $L_2 = \{a\}$, $L_3 = \phi$. Compute $L_1 L_2^* \cup L_3^*$.

2. Write regular expression for set of all strings such that number of a's divisible by 3 over Σ = {a,b}.

3. Find the regular expression corresponding to the finite automata given bellow:

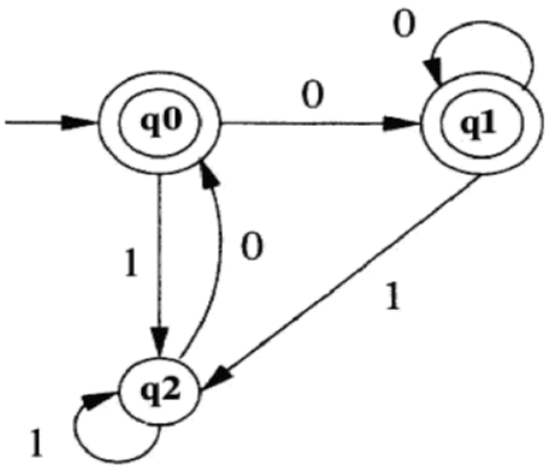


4. Prove that the following Language L = $\{a^n b^n\}$ is not regular.

5. Explain the Closure properties of regular expression.

6. Design a regular expression that accepts all the strings for input alphabet {a,b} containing exactly 2 a's.

7. State recursive definition of regular expression and construct regular expression corresponding to the following state transition diagram:-



8. State pumping lemma for regular sets. Show that L = $\{a^p$! p is a prime number$\}$ is not regular set.

9. Discuss closure properties of regular languages under the operations: concatenation, union, intersection and complement.

10. Give the regular expression for set of all strings over 0,1 containing exactly three 0's.

11. Write a regular expression to denote a language L which accepts all the strings that begin or end with either 00 or 11.

12. State closure properties of regular languages. Also Prove that regular languages are closed under intersection and difference.

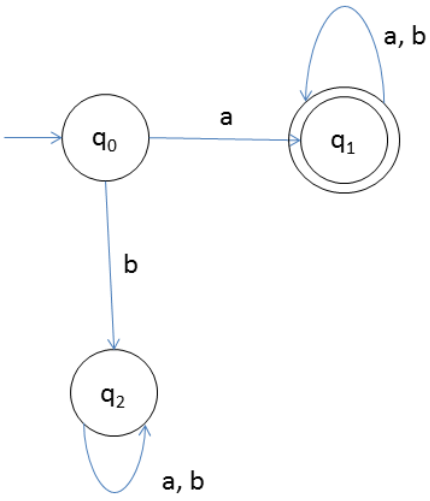
13. State Arden's theorem and construct regular expression for the following FA using Arden's theorem:

State	Input	
	0	1
A	{A, B}	ϕ
B	C	{A, B}
C	B	ϕ
A is the initial state and C is Final State		

14. Using pumping lemma, prove that the language L = $\{a^{i^2}$! i$\geq$ 1$\}$ is not regular.

15. State the pumping lemma theorem for regular languages.

16. Convert the FA given below to left linear grammar.



17. Prove that the compliment, homomorphism and inverse homomorphism, closure of a regular language is regular.

18. State and prove kleene's theorem with an example.

19. Write regular expression for a language containing strings of 0's and 1's which does not end in '01'.

20. State and prove Arden's Theorem.

21. Prove or disprove the following regarding regular expressions:
    i. (R + S)* = R* + S*
    ii. (RS + R)* RS = (RR*S)*

Chapter 5

Context Free Grammar (CFG)

5.1 Definition

A grammar is said to be context free grammar if all the production rules of the grammar are of the following form:-

$A \to \alpha$ where $\alpha \in (V \cup \Sigma)^*$ and $A \in V$

5.2 Derivation Tree

A tree is said to be derivation tree if it satisfies the following properties:-

1. All the nodes of the tree are labeled by variable, terminal or ϵ symbol.
2. The root node of the tree has labeled S (Starting symbol of the grammar).
3. All the internal nodes have labeled variable symbol.
4. All the leaf nodes have labeled terminal symbol or ϵ symbol.
5. If A $\to X_1X_2.....X_n$ be a production rule used in the derivation of the string, then in the tree, A will be at the parent node and $X_1, X_2, ..., X_n$ will be at the children of this node A.

5.3 Left Most Derivation

A derivation A$\overset{*}{\Rightarrow}$w is said to be left most derivation if we apply the production rule in the derivation at the left most variable in every step.

5.4 Right Most Derivation

A derivation A$\overset{*}{\Rightarrow}$w is said to be right most derivation if we apply the production rule in the derivation at the right most variable in every step.

5.5 Some Examples

Example: Consider the following grammar
S→ 0B/1A
A→ 0/0S/1AA
B→ 1/1S/0BB
For the string 00110101, find the left most derivation, right most derivation and derivation tree.

Example: Consider the following grammar
S→ AA
A→ a/bA/Ab/AAA
Find parse tree for the string bbaaaab.

Example: Consider the following grammar
S→ aAS/a
A→ SbA/SS/ba
Find derivation tree for the string aabbaa.

5.6 Ambiguity in Grammar and Language

Ambiguous String
A string w ∈ L(G) is said to be ambiguous string if there exists more than one derivation for the string.

Ambiguous Grammar
A grammar G is said to be ambiguous if there exists some string w ∈ L(G) for which more than one derivation tree are possible.

Example: Consider the following grammar:-
S→S+S/S*S/a/b
Is this grammar ambiguous?
Solution:

Example: Consider the grammar G,
S→SbS/a.
Show that grammar G is ambiguous.
Solution:

Example: Consider the following grammar:-
S→ a/abSb/aAb
A→ bS/aAAb
Is this grammar ambiguous?
Solution:

Example: Consider the following grammar:-

S→ aB/ab
A→ aAB/a
B→ ABb/b
Is this grammar ambiguous?
Solution:

5.7 Inherent Ambiguity

- If L is a context free language for which there exists an unambiguous grammar, then L is said to be unambiguous.
- If every grammar that generates L is ambiguous, then the language is said to be inherently ambiguous.

Example: Following language is inherent ambiguous
L = $\{a^n b^n c^m d^m !n \geq 1, m \geq 1\} cup \{a^n b^m c^m d^n !ngeq1, m \geq 1\}$

5.8 Simplification of Context Free Grammar

Simplification of grammar means to remove the useless symbols (variables and terminals) and production rules from the grammar.

5.8.1 Active Variable

A variable A is said to be active if it derives a terminal string i.e
A→ x , where x∈ Σ^*

5.8.2 Reachable symbols

A symbol is said to be reachable if it appears in a string derives from starting symbol of the grammar i.e.
If S ⇒ X, then every symbols in X are said to be reachable.

5.8.3 Useful Variable

A variable is said to be useful if it is active and reachable both.

5.8.4 Construction a Grammar in which all the variables are active (Elimination of Non-Active Variables from a Grammar)

Suppose the given context free grammar is
G = (V,Σ,S,P)
Now we construct a context free grammar G' in which all the variables are active.

G' = (V',Σ, S, P')
Step-1: Determination of V'
Let A_i denote the set of active variables. A_i is determined recursively as following:-
A_1 = {A ∈ V ! if A→ $x \in P$, where $x \in \Sigma^*$ }
$A_2 = A_1 \cup$ {A ∈ V ! if A→ $\alpha \in P$, where $\alpha \in (A_1|cup\Sigma)^*$ }
.....
.....
$A_{i+1} = A_i \cup$ {A ∈ V ! if A→ $\alpha \in P$, where $\alpha \in (A_i|cup\Sigma)^*$ }
Repeat this process until $A_{i+1} = A_i$
Now, we terminate this process.
Now, V' = A_i
Step-2: Determination of P'
P' is obtained from P by removing those production rules in which non-active variables belong.

Example: Consider the grammar
G = ({S,A,B,C,E}, {a,b,c}, P, S)
where P = { S→AB, A→a, B→b, B→C, E→c/ϵ
Eliminate the non-active variables from this grammar.

5.8.5 Construction a Grammar in which all the symbols are reachable (Elimination of Non-Reachable Symbols from a Grammar)

Suppose the given context free grammar is
G = (V,Σ,S,P)
Now we construct a context free grammar G' in which all the symbols are reachable.
G' = (V',Σ', S, P')
Step-1: Determination of V' and ,Σ'
Let R_i denote the set of reachable symbols. R_i is determined recursively as following:-
R_1 = {S }
$R_2 = R_1 \cup$ {x ! A→ $\alpha \in P$, where $A \in R_1$ and $\alpha \in (V \cup \Sigma)^*$ and α contains x}
.....
.....
$R_{i+1} = R_i \cup$ {x ! A→ $\alpha \in P$, where $A \in R_i$ and $\alpha \in (V \cup \Sigma)^*$ and α contains x}
Repeat this process until $R_{i+1} = R_i$
Now, we terminate this process.
Now, V' = $R_i \cap V$
Now, Σ' = $R_i \cap \Sigma$
Step-2: Determination of P'
P' is obtained from P by removing those production rules in which non-reachable symbols belong.

Example: Consider the grammar
G = ({S,A,B,C,E}, {a,b,c}, P, S)
where P = { S→AB, A→a, B→b, B→C, E→c/ϵ
Eliminate the non-reachable symbols from this grammar.

5.8.6 Construction of Reduced Grammar

Reduced Grammar: A grammar is said to be reduced grammar if all the symbols and production rules are useful.

Procedure:
Suppose the given grammar is G.
This methods consists of two steps. They are:-
Step-1: Find the grammar G' equivalent to G in which all the variables are active.
Step-2: Find the grammar G" equivalent to G' which includes only reachable symbols.
This grammar G" is a reduced grammar equivalent to G.

Example: Find the reduced grammar equivalent to the following grammar
S $\rightarrow$ AB/CA
B $\rightarrow$ BC/AB
A $\rightarrow$ a
C $\rightarrow$ aB/b

Example: Reduce the following grammar
S $\rightarrow$ aAa
A $\rightarrow$ Sb/bCC/DaA
C $\rightarrow$ abb/DD
E $\rightarrow$ aC
D $\rightarrow$ aDA

5.9 Elimination of null productions

Null Production: A production rule is said to be null production if it is of the following form:-

$$A \rightarrow \epsilon$$

Nullable Variable: A variable A is said to be nullable if it derives empty string i.e.

$$A \Rightarrow \epsilon,\ \forall A \in V$$

Procedure:
Consider a grammar G = (V, Σ, S, P). Let G' is a grammar having no null productions such that L(G') = L(G)-{ϵ}.

G' is constructed as following:-

G' = (V, Σ, S, P')

Step-1: Determination of the set of nullable variables
Let W_i is the set of nullable variables. W_i is calculated as following:-
$W_1 = \{A \mid A \rightarrow \epsilon \in P\}$
$W_2 = W_1 \cup \{A \mid \exists$ a production $A \rightarrow \alpha \in P$ and $\alpha \in W_1^*\}$
.....
.....

$W_{i+1} = W_i \cup \{A \mid \exists$ a production $A \to \alpha \in P$ and $\alpha \in W_i^*\}$
Repeat this process until $W_{i+1} = W_i$
Now, we terminate this process.
Step-2: Determination of P'
(i) We add the production rules of P into P' whose RHS does not include any nullable variable.
(ii) Consider the remaining production rules of P.
If A$\to X_1X_2.....X_n \in P$ then we add A$\to \alpha_1\alpha_2.....\alpha_n$ into P', where $\alpha_i = X_i$ or ϵ but not all α_i equal to ϵ if X_i is nullable variable otherwise put $\alpha_i = X_i$.

Example: Consider the following grammar
S$\to$ aS/AB
A$\to \epsilon$
B$\to \epsilon$
D$\to$ b
Eliminate the null productions.

5.10 Elimination of Unit Productions

Unit Production: A production rule is said to be unit production if it is of the following form:-
A$\to$B, where A, B $\in$ V.

Example: Consider the following grammar
S$\to$ AB
A$\to$a
B$\to$C/b
C$\to$ D
D$\to$ E
E$\to$ a
Eliminate the unit productions.

5.11 Chomosky Normal Form (CNF)

5.11.1 Definition

A grammar G is said to be in Chomosky nomal form if every prodution rules are of the following form:-

A$\to$BC or A$\to$ a,

where A,B,C $\in$V and a $\in \Sigma$.

5.11.2 Reduction into Chomosky Normal Form

Step-1: Elimination of null productions and unit productions
In this step, we eliminate the null production and unit productions from the grammar. Let the resultant grammar is G=(V,Σ,S,P).
Step-2: Elimination of terminals from RHS if rule is not in CNF

Let $G_1 = (V_1, \Sigma, S, P_1)$ be the grammar obtained in this step.
All the productions of P which are in CNF, must also belong into P_1 and all the variables of V must also belong into V_1.
Consider the remaining production rules.
Add the new variables into V_1 equal to the number of terminals on the right hand side of these production rules. And if the terminals are $a_1, a_2, ..., a_n$, then add variables $X_1, X_2, ..., X_n$ into V_1 and add $X_1 \rightarrow a_1$, $X_2 \rightarrow a_2$,.....,$X_n \rightarrow a_n$ rules into P_1.
Step-3: Restricting the number of variables on the RHS
Let $G_2 = (V_2, \Sigma, S, P_2)$ be the grammar obtained in this step.
Add all the elements of V_1 into V_2. Add the production rules of P_1 into P_2 which are in CNF.
Consider those production rules of P_1 which are not in CNF. These production rules contains at least three elements on the right hand side.
Consider production rules $A \rightarrow X_1X_2.....X_n$, where n≥3. Then we add n-2 variable into V_2. Let these are $Y_1, Y_2, ..., Y_{n-2}$. Add production rules $A \rightarrow X_1Y_1$, $Y_{\rightarrow}X_2Y_2$,.....,$Y_{n-2} \rightarrow X_{n-1}X_n$ to P_2.
Now, the grammar G_2 is the resultant grammar which is in CNF.

Example: Reduce the following grammar into CNF:-
(1) S→aAD, A→aB/bAB, B→b, D→d.
(2) S→aAbB, A→a/aA, B→b/bB.
(3) S→˜S/[S⊃)S]/p/q.

5.12 Greibach Normal Form (GNF)

5.12.1 Definition

A CFG is said to be in Greibach normal form if all the production rules are of the following form:

A→ $a\gamma$, where $\gamma \in V^*$, a ∈ Σ and A∈V.

5.12.2 Lemma-1

Let A→ $B\gamma$ be an A-production and let B-productions are B→ $\beta_1|\beta_2|.....|\beta_n$. Then A→ $B\gamma$ production is replaced by the following rules:-
A→ $\beta_1\gamma|\beta_2\gamma|.....|\beta_n\gamma$.

5.12.3 Lemma-2

Let the set of A-productions be A→ $A\alpha_1|A\alpha_2|.....|A\alpha_m|\beta_1|\beta_2|.....|\beta_n$ (β_i's do not start with A). Then we replace the rules A→ $A\alpha_1|A\alpha_2|.....A\alpha_m$ by the following procedure:-
Add new variable Z to the grammar. And the set of A-production are
A→ $\beta_1|\beta_2|.....|\beta_n$
A→ $\beta_1Z|\beta_2Z|.....|\beta_nZ$
The set of Z-productions are
Z→ $\alpha_1|\alpha_2|.....|\alpha_m$

Z→ $\alpha_1 Z|\alpha_2 Z|.....|\alpha_m Z$

5.12.4 Reduction to Greibach Normal Form

Step-1: Construct the given grammar into CNF. Next, we rename the variables as $A_1, A_2, ..., A_n$ with S = A_1.
Step-2: Consider the production rules which are of the following form
$A_i \rightarrow A_j\gamma$, where j¡i and $\gamma \in V^*$
Apply the lemma-1 in these rules until j≥i.
Step-3: Consider the production rules which are of the following form
$A_i \rightarrow A_j\gamma$, where j = i and $\gamma \in V^*$
Apply the lemma-2 in these rules until j≥i.
Step-4: Consider the production rules which are not in GNF. Apply lemma-1 in all these rules. After this, the resultant grammar will be in GNF.

Example: Convert the following grammar into GNF.

1. S→AB, A→BS/b, B→SA/a.
2. S→AA/a, A→SS/b.
3. E→E+T/T, T→T*F/F, F→(E)/a,
4. S→ABb/a, A→aaA, B→bAb.
5. S→SS, S→0S1/01, B→SA/a.

5.13 Exercise

1. Eliminate the useless production from the following grammar
   S→a/aA/B/C, A→aB/ϵ, B→aA, C→cCD, D→ddd,.
2. Eliminate all the ϵ-productions from the following grammar:-
   S→AaB/aaB, A→ ϵ, B→bbA/ϵ
3. Remove all unit productions, all useless productions and all ϵ-productions from the grammar
   S→aA/aBB, A→aaA/ϵ, B→bB/bbC, C→B.
4. Transform the following grammar into CNF
   S→abAB, A→bAB/ϵ, B→BAa/A/ϵ .
5. Transform the following grammar into CNF
   S→AB/aB, A→aab/ϵ, B→bbA .

5.14 Closure Properties of Context Free Languages

Theorem: Show that the family of context free languages is closed under union operation.
Proof: Let L_1 and L_2 be two context free languages generated by context free grammar $G_1 = (V_1, \Sigma_1, S_1, P_1)$ and $G_1 = (V_2, \Sigma_2, S_2, P_2)$ respectively.
Now, we construct the grammar G as the following;-
$G = (V, \Sigma, S, P)$
Where, V $= V_1 \cup V_2 \cup \{S\}$
$\Sigma = \Sigma_1 \cup \Sigma$
and P $= P_1 \cup P_2 \cup \{S \to S_1 | S_2\}$
Now, we have to show that
L(G) $= L(G_1) \cup L(G_2)$
Let x $\in$ L(G) $\Leftrightarrow$ S $\stackrel{*}{\Rightarrow}$ x
$\Leftrightarrow$ S $\Rightarrow S_1 \stackrel{*}{\Rightarrow}$ x or
S $\Rightarrow S_2 \stackrel{*}{\Rightarrow}$ x or
$\Leftrightarrow S_1 \stackrel{*}{\Rightarrow}$ x or $S_2 \stackrel{*}{\Rightarrow}$ x
$\Leftrightarrow$ x$\in L(G_1)$ or x$\in L(G_2)$
$\Leftrightarrow$ x$\in L(G_1) \cup L(G_2)$
Therefore, L(G) $= L(G_1) \cup L(G_2)$
Clearly, G is a CFG for $L_1 \cup L_2$, therefore $L_1 \cup L_2$ is also a context free language.

Theorem: Show that the family of context free languages is closed under concatenation operation.
Proof: Let L_1 and L_2 be two context free languages generated by context free grammar $G_1 = (V_1, \Sigma_1, S_1, P_1)$ and $G_1 = (V_2, \Sigma_2, S_2, P_2)$ respectively.
Now, we construct the grammar G as the following;-
$G = (V, \Sigma, S, P)$
Where, V $= V_1 \cup V_2 \cup \{S\}$
$\Sigma = \Sigma_1 \cup \Sigma$
and P $= P_1 \cup P_2 \cup \{S \to S_1.S_2\}$
Now, we have to show that
L(G) $= L(G_1).L(G_2)$
Let x $\in$ L(G) $\Leftrightarrow$ S $\stackrel{*}{\Rightarrow}$ x
$\Leftrightarrow$ S $\Rightarrow S_1.S_2 \stackrel{*}{\Rightarrow}$ x
$\Leftrightarrow S_1 \stackrel{*}{\Rightarrow} x_1$ and $S_2 \stackrel{*}{\Rightarrow} x_2$ (Let x= x_1x_2)
$\Leftrightarrow x_1.x_2 \in L(G_1).L(G_2)$
$\Leftrightarrow$ x$\in L(G_1).L(G_2)$
Therefore, L(G) $= L(G_1).L(G_2)$
Clearly, G is a CFG for $L_1.L_2$, therefore $L_1.L_2$ is also a context free language.

Theorem: Show that the family of context free languages is closed under kleene closure operation.
Proof: Let L is a context free languages and G is a context free grammar generating L.
$G = (V, \Sigma, S, P)$
Now, we construct a grammar G' as the following:-
$G' = (V', \Sigma, S', P')$
Where, V' = V$\cup${S'}

P' = P$\cup\{S' \to SS'|\epsilon\}$
Now, we have to show that L(G') = (L(G))*.
Let x $\in$ L(G') $\Leftrightarrow$ S' $\overset{*}{\Rightarrow}$ x
$\Leftrightarrow$ S' $\Rightarrow$ $S.S.S.....S(n-times) \overset{*}{\Rightarrow}$ x
$\Leftrightarrow$ $S \overset{*}{\Rightarrow} x_1$, $S \overset{*}{\Rightarrow} x_2$ .....$S \overset{*}{\Rightarrow} x_n$ (Let x= $x_1x_2.....x_n$)

$\Leftrightarrow$ $x_1 \in L(G)$, $x_2 \in L(G)$, ..., $x_n \in L(G)$

$\Leftrightarrow$ $x_1.x_2.....x_n \in (L(G))^n$
$\Leftrightarrow$ $x \in (L(G))^*$ $(L(G))^n \subseteq (L(G))^*$

Therefore, L(G') = (L(G))*.
Therefore, G' is a grammar generating the language L*. Hence L* is a context free language.

Theorem: Show that the family of context free languages is not closed under intersection operation.
Proof: Consider the two context free languages:-
$L_1 = \{a^nb^nc^m \mid n \geq 0, m \geq 0\}$
$L_2 = \{a^nb^mc^m \mid n \geq 0, m \geq 0\}$
The intersection of these languages is
$L_1 \cap L_2 = \{a^nb^nc^n \mid n \geq 0\}$
This language is not context free.
Therefore, the family of context free languages is not closed under intersection operation.

Theorem: Show that the family of context free languages is not closed under complement operation.
Proof: Consider the following formula

$$L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})} \quad .....(1)$$

Suppose the context free languages are closed under complement operation.
From (1), If L_1 and L_2 are context free languages, then $\overline{L_1}$ and $\overline{L_2}$ are also context free language.
Since $\overline{L_1}$ and $\overline{L_2}$ are context free, therefore $\overline{L_1} \cup \overline{L_2}$ will also be context free language.
Therefore $\overline{(\overline{L_1} \cup \overline{L_2})}$ ias also context free.
Since the R.H.S. of equation (1) is context free, therefore L.H.S. is also context free. But, by previous theorem $L_1 \cap L_2$ is not context free, therefore context free languages is not closed under complement operation.

5.15 Pumping Lemma for Context Free Languages

Let L be an infinite context free language. Then there exists some positive integer n such that any x$\in$ L with x$\geq$n, can be decomposed as
x=uvwxy
with $| vxy | \leq n$ and $| vx | \geq 1$,

such that
$uv^iwx^iy \in$ L, $\forall$ i = 0,1,2,3,.....

Application: This lemma is used to show a given language is not context free.

Example: Show that the language
L = { $a^nb^nc^n \mid n \geq 0$}
is not context free.
Solution:

5.16 Exercise

1. L = {ww ! w$\in${a,b}*}
2. L = {$a^{n!}$! $n \geq 0$}
3. L = {a^nb^j ! $n = j^2$}
4. L = {a^{n^2} ! $n \geq 0$}
5. L = {a^p ! p is a prime number}
6. L = {$a^nb^jc^k$! $k > n,\ k > j$}

5.17 Decision Properties of Regular and Context Free Languages

Theorem: Given a context free grammar G=(V,Σ,S,P), there exists an algorithm for deciding whether or not L(G) is empty.
Proof: For the simplicity, we assume that $\epsilon \notin L(G)$. We use the algorithm for removing useless symbols and productions. If S is found to be useless, then L(G) is empty otherwise L(G) contains at least one element.

Theorem: Given a context free grammar G=(V,Σ,S,P), there exists an algorithm for deciding whether or not L(G) is infinite.
Proof: We assume that G contains no $\epsilon - productions$, no unit-productions, and no useless symbols.
Convert the grammar into CNF.
We draw a directed graph whose vertices are variables in G. If A$\rightarrow$BC is a production, then there are directed edges from A to B and A to C.
L is finite iff the directed graph has no cycles.

Theorem: Show that there exists an algorithm for deciding whether a regular language, L is empty.
Proof: Construct a deterministic finite automata M accepting L. Determine the set of all the states reachable from q_0. If this set contains a final state, then L is non-empty otherwise L is empty.

Theorem: Show that there exists an algorithm for deciding whether a regular language, L is infinite.
Proof: Construct a deterministic finite automata M accepting L. L is infinite iff M has a cycle.

5.18 AKTU Examination Questions

1. Convert the following CFG to its equivalent GNF:
   S → AA a, A → SS b.

2. Prove that the following Language L = $\{a^n b^n c^n \ ! \ n \geq 1\}$ is not Context Free.

3. Is context free language closed under union? If yes, give an example.

4. Remove useless productions from the following grammar
   S→AB/ab, A→a/aA/B, B→D/E

5. Reduce the given Grammar G = ({S,A,B},{a,b},S,P) to Chomosky Norma Form, where P is the following
   S→bA/aB, A→a/aS/bAA, B→b/bS/aBB

6. Discuss the inherent ambiguity of context free languages with suitable example. Construct the context free grammar that accept the following language
   L = $\{a^i b^j c^k \ ! \ i = j or j = k\}$

7. Define the parse tree. Construct the parse tree for the string abbcde considering the productions
   S→aAcBe, A→b/Ab, B→d
   Is this ambiguous? Justify.

8. Prove or disprove that union and concatenation of two context free languages is also context free.

9. Prove that the language L = $\{a^n b^n c^n \ ! \ n \geq 1\}$ is neither regular nor context free.

10. Determine the language generated by grammar $S \rightarrow Sab|aSb|abS|baS|bSa|Sba|aS|a$

11. What is inherent ambiguity? Explain with the help of suitable example.

12. Remove the Unit productions from the following grammar: $S \rightarrow aSb|A$, $A \rightarrow cAd|cd$

13. Write the procedure to convert a given CFG into equivalent grammar in CNF. Apply the procedure and convert the grammar with following production into CNF: $S \rightarrow bA|aB$, $A \rightarrow bAA|aS|a$, $B \rightarrow aBB|bS|b$

14. Define Greibach normal form for a CFG. Reduce the following CFG into GNF: S → AB, $A \rightarrow BS|a$, $B \rightarrow A|b$

15. Let G be the grammar $S \rightarrow 0B|1A$, $A \rightarrow 0|0S|1AA$, $B \rightarrow 1|1S|0BB$ For the string 00110101, find: (i) The leftmost derivation. (ii) The rightmost derivation. (iii) The derivation tree.

16. Check whether the grammar is ambiguous or not. R→ R+R/ RR/ R*/ a / b / c. Obtain the string w = a+b*c

17. Eliminate unit productions in the grammar. S→A/bb A→B/b B→S/a

18. Find out whether the language L = $\{x^n y^n z^n | n \geq 1\}$ is context free or not.

19. Convert the following CFG into CNF
    S → XY / Xn / p
    X → mX / m
    Y → Xn / o

20. Convert the following CFG into CNF S → ASA / aB, A → B / S, B → b / ϵ

21. Write CFG for language L = $\{a^n b^n \ !n \geq 0\}$. Also convert it into CNF.

22. Define ambiguity. Show that the grammar G with following production is ambiguous.
    S → a / aAb / abSb, A → aAAb / bS

23. Convert the following grammar in GNF: S → AB , A→ BS / a , B→SA / b

24. Define derivation Tree. Show the derivation tree for string 'aabbbb' with the following grammar S→AB/ϵ, A→aB, B→Sb.