Q.2 a. Using the principle of mathematical induction, prove that $(10^{(2n-1)}+1)$ is divisible by 11 for all n ∈ N (8)

Answer:

Let P(n): $(10^{(2n-1)}+1)$ is divisible by 11

For n = 1, the given expression becomes $(10^{(2*1-1)}+1) = 11$, which is divisible by 11.

So, the given statement is true for n=1, i.e. P(1) is true.

Let P(k) be true. Then

P(k): $(10^{(2k-1)}+1)$ is divisible by 11

=> $(10^{(2k-1)}+1) = 11m$, for some natural number m.

Now, $\{(10^{(2(k+1)-1)}+1)\} = (10^{(2k+1)}+1) = \{10^2 . 10^{(2k-1)} +1\}$

$= 100 \times \{10^{(2k-1)}+1\} - 99$

$= (100 \times 11m)-99$

$= 11 \times (100m - 9)$, which is divisible by 11

=> P(k+1): $(10^{(2(k+1)-1)}+1)$ is divisible by 11

=> P(k+1) is true, whenever P(k) is true.

Thus, P(1) is true and P(k+1) is true, whenever P(k) is true.

Hence by the principle of mathematical induction, P(n) is true for all n ∈ N.

b. Discuss the description of Finite Automata. Why we study Automata Theory in computer science? (8)

Answer:

Finite Automata

A finite automaton is an abstract model of a digital computer. A finite automaton has a mechanism to read input, which is a string over a given alphabet. This input is actually written on an “input file”, which can be read by the automaton but cannot change it.

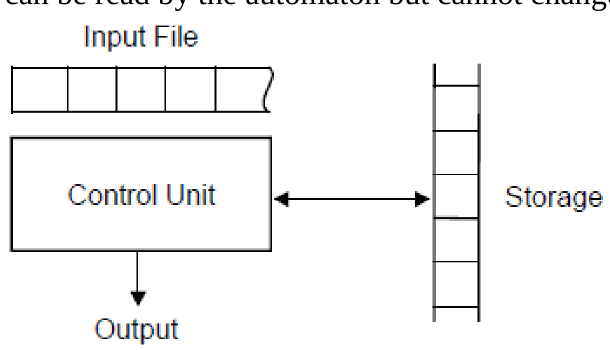


Fig. Automaton

Input file is divided into cells, each of which can hold one symbol. The automaton has a temporary “storage” device, which has unlimited number of cells, the contents of which can be altered by the automaton. Automaton has a control unit, which is said to be in one of a finite number of “internal states”.

The automaton can change state in a defined way.

Types of Finite Automaton

(a) Deterministic Finite Automata

(b) Non-deterministic Finite Automata

A deterministic automata is one in which each move (transition from one state to another) is unequally determined by the current configuration. If the internal state, input and contents of the storage are known, it is possible to predict the future behaviour of the automaton. This is said to be deterministic finite automata otherwise it is nondeterministic finite automata.

Definition of Deterministic Finite Automaton

A Deterministic Finite Automata (DFA) is a collection of 5-tuples as:

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

where

Q	=	Finite state of "internal states"
Σ	=	Finite set of symbols called "Input alphabet"
$\delta : Q \times \Sigma \to Q$	=	Transition Function
$q_0 \in Q$	=	Initial state
$F \subseteq Q$	=	Set of Final states

The input mechanism can move only from left to right and reads exactly one symbol on each step.

The transition from one internal state to another is governed by the transition function δ.

If $\delta(q0, a) = q1$ then if the DFA is in state $q0$ and the current input symbol is a, the DFA will go into state $q1$.

Definition of Nondeterministic Finite Automaton

A Nondeterministic Finite Automata (NFA) is defined by a collection of 5-tuples:

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

where $Q, \Sigma, \delta, q_0, F$ are defined as follows:

Q = Finite set of internal states

Σ = Finite set of symbols called "Input alphabet"

$\delta = Q \times (\Sigma \cup \{\lambda\}) \to 2^Q$

$q_0 \in Q$ is the Initial states

$F \subseteq Q$ is a set of Final states

NFA differs from DFA in that, the range of δ in NFA is in the powerset 2^Q.

A string is accepted by an NFA if there is some sequence of possible moves that will put the machine in the final state at the end of the string.

Need of Study and Applications of Finite Automata

String Processing

Consider finding all occurrences of a short string (pattern string) within a long string (text string). This can be done by processing the text through a DFA: the DFA for all strings that end with the pattern string. Each time accept state is reached; the current position in the text is output.

Finite-State Machines
A finite-state machine is an FA together with actions on the arcs.

Statecharts
Statecharts model tasks as a set of states and actions. They extend FA diagrams.

Lexical Analysis
In compiling a program, the first step is lexical analysis. This isolates keywords, identifiers etc., while eliminating irrelevant symbols. A token is a category, for example "identifier", "relation operator" or specific keyword.

Q.3 a. Solve the following: (3+2)
(i) Construct a DFA that behaves equivalent to the NDFA given by M such that M = ({q0, q1, q2, q3} , {a, b}, δ , q0 , {q3}) where δ is given by

STATE	Input a	Input b
Initial state →q0	q0, q1	q0
q1	q2	q1
q2	q3	q3
Final State *q3	--	q2

(ii) Find a DFA machine that accepts an even number of either 0's or 1's or both 0's and 1's over input symbol ∑ = {0, 1}.

Answer: (i)

Let $Q = \{q_0, q_1, q_2, q_3\}$. Then the deterministic automaton M_1 equivalent to M is given by

$$M_1 = (2^Q, \{a, b\}, \delta, [q_0], F)$$

where F consists of:

$[q_3]$, $[q_0, q_3]$, $[q_1, q_3]$, $[q_2, q_3]$, $[q_0, q_1, q_3]$, $[q_0, q_2, q_3]$, $[q_1, q_2, q_3]$

and

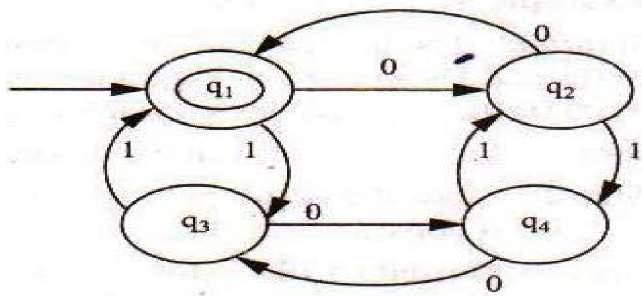
$$[q_0, q_1, q_2, q_3]$$

and where δ is defined by the state table given by Table

TABLE State Table of M_1

State/Σ	a	b
$[q_0]$	$[q_0, q_1]$	$[q_0]$
$[q_0, q_1]$	$[q_0, q_1, q_2]$	$[q_0, q_1]$
$[q_0, q_1, q_2]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_3]$
$[q_0, q_1, q_3]$	$[q_0, q_1, q_2]$	$[q_0, q_1, q_2]$
$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$	$[q_0, q_1, q_2, q_3]$

(ii) The transition system (δ) of the DFA machine is shown in the following figure**:**



Hence the DFA machine is defined as ({q1, q2, q3, q4}, {0, 1}, δ, q1, {q1}).

b. Find a DFA machine for the language L = {0w0 | w ϵ {0, 1}*}. (4)

Answer:

Solution: The language contains first and last letter as 0 and intermediate letters may be of any order of 0′s and 1′s and of any length including ∈ i.e. the set of language looks as {00, 000, 010, 0000, 0010, 0100,0110,…..}. To design a DFA machine we use the following steps

Step 1: Let us consider an initial state say q_0

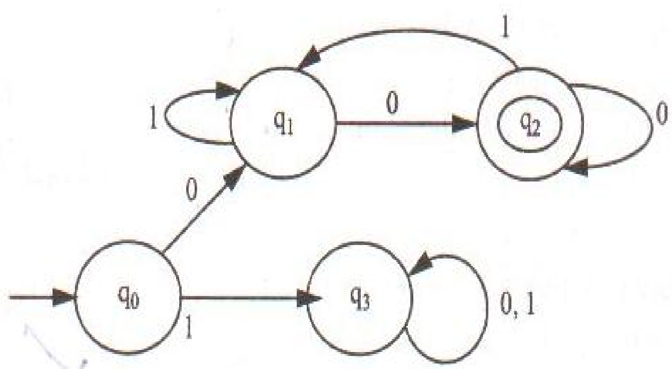
Step 2: Since the first input symbol is 0 then on initial state q_0 (or present state q_0) and input symbol 0 consider the next state as q_1.

State 3: On state q_1 the input symbol may be any one i.e. either 0 or 1 of any order and of any length. If input symbol is 0 then the next state is q_2 which is a final state and if input symbol is 1 the next state is q_1 because 1 may be of any length.

Step 4: On state q_2 if input symbol is 0 then the next state will not change i.e. it will remain q_2 but if input symbol is 1 then the next state is q_1

Step 5: On initial state q_0 if input symbol is 1 the string will not be accepted by DFA.

The transition graph of the given DFA machine is shown in the following figure. This transition graph consists of 5 tuples as (Q, ∑, δ, q0, F) that is defined as **({q0, q1, q2, q3}, {0, 1}, δ, q0, {q2}).**



c. Make a minimum state finite automaton that is equivalent to a DFA whose transition table is given as: (7)

Definition	State	a	b
Initial State	→q0	q1	q2
	q1	q4	q3
	q2	q4	q3
Final State	*q3	q5	q6
Final State	*q4	q7	q6

q5	q3	q6
q6	q6	q6
q7	q4	q6

Answer:

$$Q_1^0 = \{q_3, q_4\}, \quad Q_2^0 = \{q_0, q_1, q_2, q_5, q_6, q_7\}$$

$$\pi_0 = \{\{q_3, q_4\}, \{q_0, q_1, q_2, q_5, q_6, q_7\}\}$$

q_3 is 1-equivalent to q_4. So, $\{q_3, q_4\} \in \pi_1$.

q_0 is not 1-equivalent to q_1, q_2, q_5 but q_0 is 1-equivalent to q_6. Hence $\{q_0, q_6\} \in \pi_1$. q_1 is 1-equivalent to q_2 but not 1-equivalent to q_5, q_6 or q_7. So, $\{q_1, q_2\} \in \pi_1$.

q_5 is not 1-equivalent to q_6 but to q_7. So, $\{q_5, q_7\} \in \pi_1$

Hence,

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

q_3 is 2-equivalent to q_4. So, $\{q_3, q_4\} \in \pi_2$.

q_0 is not 2-equivalent to q_6. So, $\{q_0\}, \{q_6\} \in \pi_2$.

q_1 is 2-equivalent to q_2. So, $\{q_1, q_2\} \in \pi_2$.

q_5 is 2-equivalent to q_7. So, $\{q_5, q_7\} \in \pi_2$.

Hence,

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

q_3 is 3-equivalent to q_4; q_1 is 3-equivalent to q_2 and q_5 is 3-equivalent to q_7. Hence,

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

As $\pi_3 = \pi_2$, the minimum state automaton is

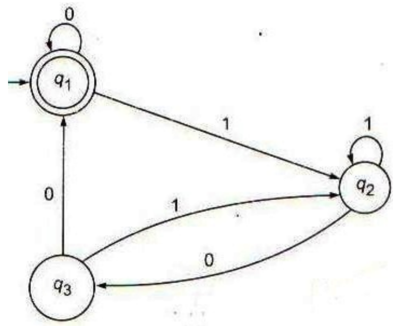
$$M' = (Q', \{a, b\}, \delta', [q_0], \{[q_3, q_4]\})$$

where δ' is defined by Table

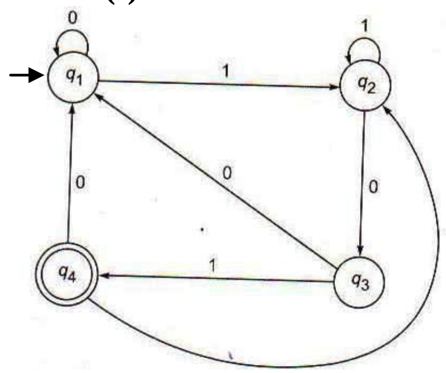
TABLE Transition Table of DFA

State	*a*	*b*
$[q_0]$	$[q_1, q_2]$	$[q_1, q_2]$
$[q_1, q_2]$	$[q_3, q_4]$	$[q_3, q_4]$
$[q_3, q_4]$	$[q_5, q_7]$	$[q_6]$
$[q_5, q_7]$	$[q_3, q_4]$	$[q_6]$
$[q_6]$	$[q_6]$	$[q_6]$

Q.4 a. Find the regular expressions corresponding to the following finite automata; consider q_1 as initial state in both automata (s): (8)



(i)



(ii)

Answer: (i)

There is only one initial state. Also, there are no Λ-moves. The equations are

$$\mathbf{q_1 = q_10 + q_30 + \Lambda}$$

$$\mathbf{q_2 = q_11 + q_21 + q_31}$$

$$\mathbf{q_3 = q_20}$$

So,

$$\mathbf{q_2 = q_11 + q_21 + (q_20)1 = q_11 + q_2(1 + 01)}$$

By applying Theorem , we get

$$\mathbf{q_2 = q_11(1 + 01)^*}$$

Also,

$$\mathbf{q_1 = q_10 + q_30 + \Lambda = q_10 + q_200 + \Lambda}$$

$$\mathbf{= q_10 + (q_11(1 + 01)^*)00 + \Lambda}$$

$$\mathbf{= q_1(0 + 1(1 + 01)^*\ 00) + \Lambda}$$

Once again applying Theorem , we get

$$\mathbf{q_1 = \Lambda(0 + 1(1 + 01)^*\ 00)^* = (0 + 1(1 + 01)^*\ 00)^*}$$

As $\mathbf{q_1}$ is the only final state, the regular expression corresponding to the given diagram is $\mathbf{(0 + 1(1 + 01)^*\ 00)^*}$.

(ii)

There is only one initial state, and there are no Λ-moves. So, we form the equations corresponding to $\mathbf{q_1}$, $\mathbf{q_2}$, $\mathbf{q_3}$, $\mathbf{q_4}$:

$$\mathbf{q_1 = q_10 + q_30 + q_40 + \Lambda}$$

$$\mathbf{q_2 = q_11 + q_21 + q_41}$$

$$\mathbf{q_3 = q_20}$$

$$\mathbf{q_4 = q_31}$$

Now,

$$\mathbf{q_4 = q_31 = (q_20)1 = q_201}$$

Thus, we are able to write $\mathbf{q_3}$, $\mathbf{q_4}$ in terms of $\mathbf{q_2}$. Using the $\mathbf{q_2}$-equation, we get

$$\mathbf{q_2 = q_11 + q_21 + q_2011 = q_11 + q_2(1 + 011)}$$

By applying Theorem , we obtain

$$\mathbf{q}_2 = (\mathbf{q}_1 1)(1 + 011)^* = \mathbf{q}_1(1(1 + 011)^*)$$

From the $\mathbf{q}_1$-equation, we have

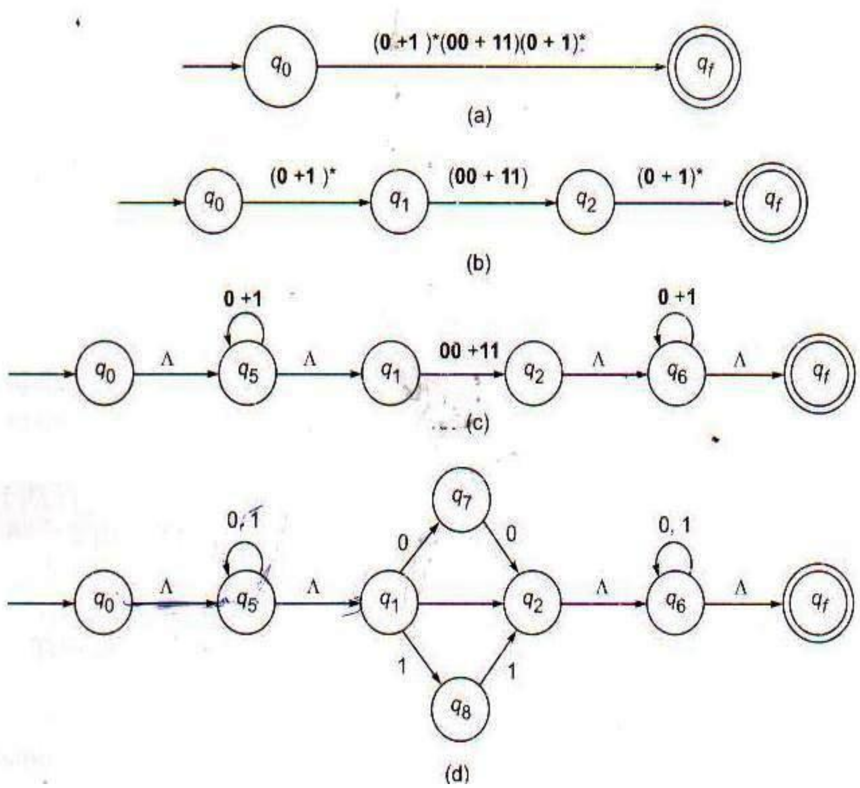
$$\mathbf{q}_1 = \mathbf{q}_1 0 + \mathbf{q}_2 00 + \mathbf{q}_2 010 + \Lambda$$
$$= \mathbf{q}_1 0 + \mathbf{q}_2(00 + 010) + \Lambda$$
$$= \mathbf{q}_1 0 + \mathbf{q}_1 1(1 + 011)^* (00 + 010) + \Lambda$$

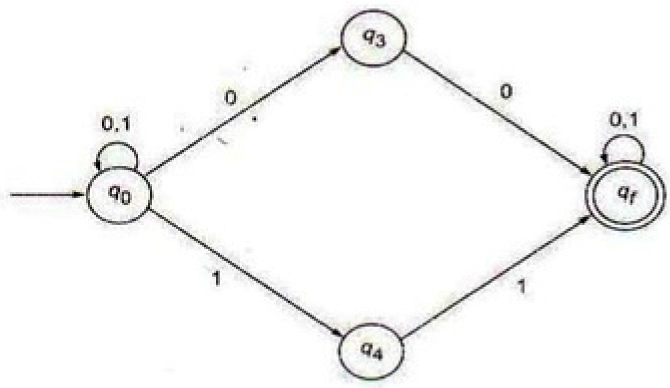
Again, by applying Theorem , we obtain

$$\mathbf{q}_1 = \Lambda(0 + 1(1 + 011)^* (00 + 010))^*$$
$$\mathbf{q}_4 = \mathbf{q}_2 01 = \mathbf{q}_1 1(1 + 011)^* 01$$
$$= (0 + 1(1 + 011)^*(00 + 010))^*(1(1 + 011)^* 01)$$

b. Obtain a Deterministic Finite Automaton with minimized or reduced states for the following regular expressions (8)
(i) (0 + 1)* (00 + 11) (0 + 1)*
(ii) 10 + (0 + 11) 0* 1

Answer: (i)





(e)

Fig. Construction of finite automaton equivalent to $(0 + 1)^*(00 + 11)(0 + 1)^*$.

Step 2 (Construction of DFA) We construct the transition table for the NDFA defined by Table

TABLE Transition Table

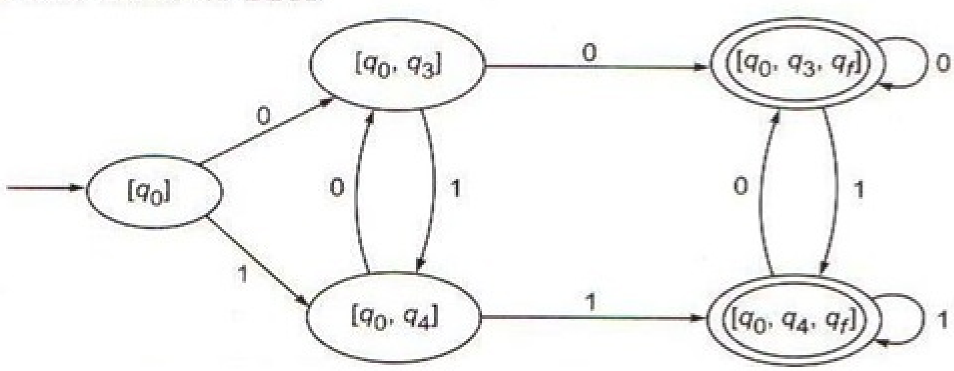
State/Σ	0	1
$\rightarrow q_0$	q_0, q_3	q_0, q_4
q_3	q_f	
q_4		q_f
q_f	q_f	q_f

The successor table is constructed as given in Table

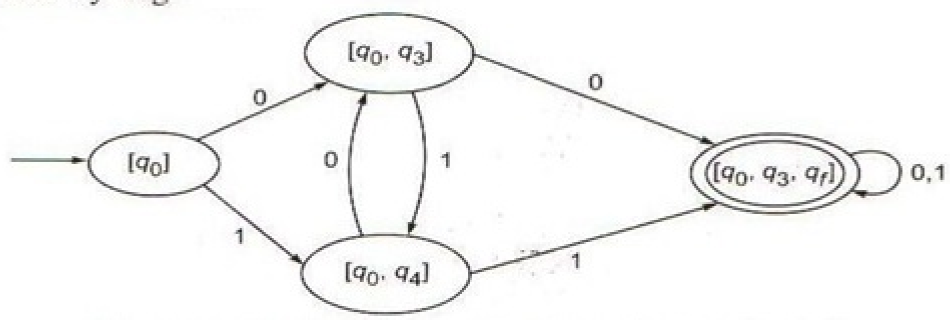
TABLE Transition Table for the DFA

Q	Q_0	Q_1
$\rightarrow [q_0]$	$[q_0, q_3]$	$[q_0, q_4]$
$[q_0, q_3]$	$[q_0, q_3, q_f]$	$[q_0, q_4]$
$[q_0, q_4]$	$[q_0, q_3]$	$[q_0, q_4, q_f]$
$[q_0, q_3, q_f]$	$[q_0, q_3, q_f]$	$[q_0, q_4, q_f]$
$[q_0, q_4, q_f]$	$[q_0, q_3, q_f]$	$[q_0, q_4, q_f]$

The state diagram for the successor table is the required DFA as described by Fig. As q_f is the only final state of NDFA, $[q_0, q_3, q_f]$ and $[q_0, q_4, q_f]$ are the final states of DFA.

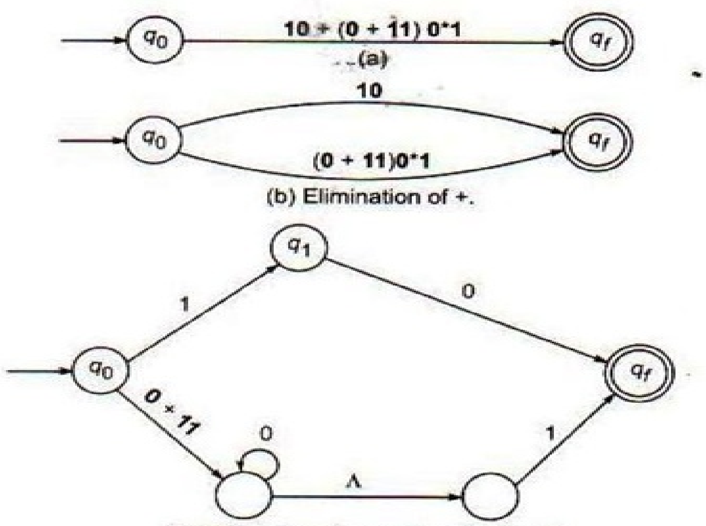


Finally, we try to reduce the number of states. (This is possible when two rows are identical in the successor table.) As the rows corresponding to $[q_0, q_3, q_f]$ and $[q_0, q_4, q_f]$ are identical, we identify them. The state diagram for the equivalent automaton, where the number of states is reduced, is described by Fig.



Sol (ii):

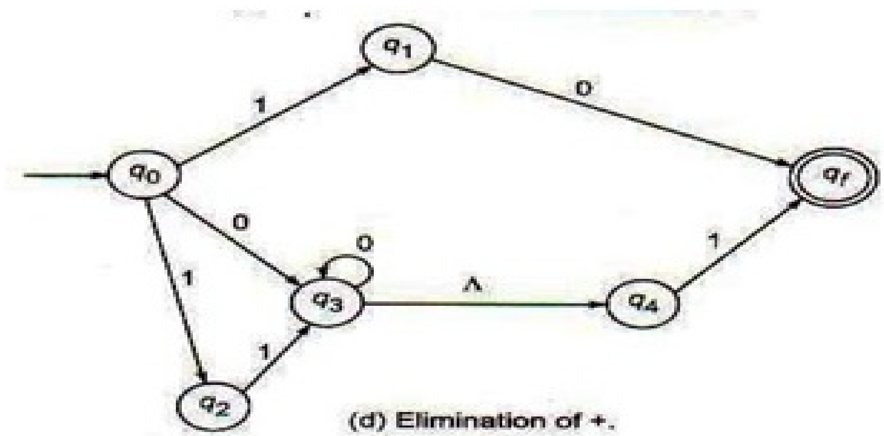
Step 1 (Construction of NDFA) The NDFA is constructed by eliminating the operation +, concatenation and *, and the Λ-moves in successive steps. The step-by-step construction is given in Figs.



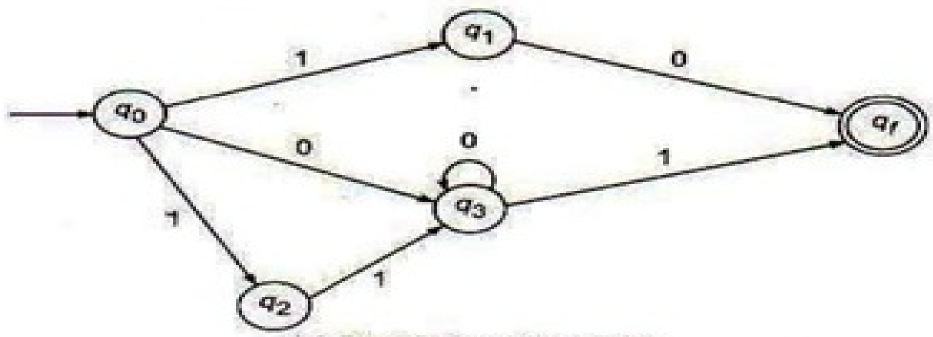
(a)

(b) Elimination of +.

(c) Elimination of concatenation and *.



(d) Elimination of +.



(e) Elimination of Λ-moves.

Step 2 (Construction of DFA) For the NDFA given in Fig. , the corresponding transition table is defined by Table

TABLE Transition Table

State/Σ	0	1
→ q_0	q_3	q_1, q_2
q_1	q_f	
q_2		q_3
q_3	q_3	q_f
q_f		

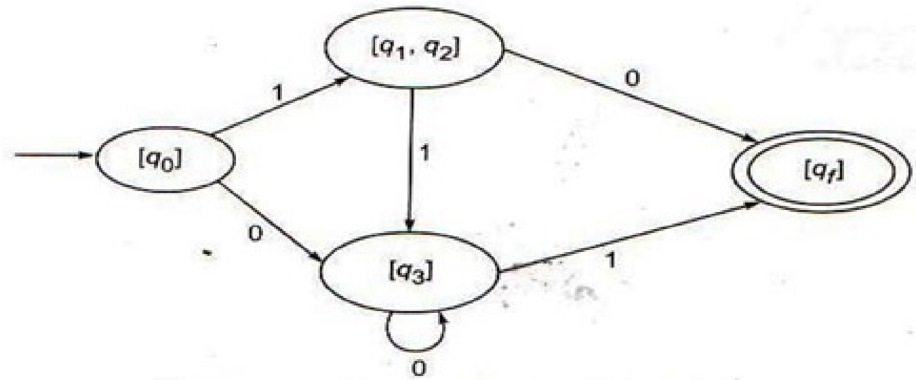
The successor table is constructed

In Table the columns corresponding to $[q_f]$ and $\emptyset$ are identical. So we can identify $[q_f]$ and $\emptyset$.

TABLE Transition Table of DFA

Q	Q_0	Q_1
→ $[q_0]$	$[q_3]$	$[q_1, q_2]$
$[q_3]$	$[q_3]$	$[q_f]$
$[q_1, q_2]$	$[q_f]$	$[q_3]$
$[q_f]$	$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$

The DFA with the reduced number of states corresponding to Table is defined by



Q.5 a. Solve the following: (3+3)

(i) Show that the language denoted by L = { 0^i 1^i | i >= 1} is not a regular language.

(ii) Check whether the language represented as L = { a^i b^j c^k | k > i + j} is regular or not? Also justify your answer using pumping Lemma.

Answer:(i)

Step 1 Suppose L is regular. Let n be the number of states in the finite automaton accepting L.

Step 2 Let $w = 0^n1^n$. Then $|w| = 2n > n$. By pumping lemma, we write $w = xyz$ with $|xy| \le n$ and $|y| \ne 0$.

Step 3 We want to find i so that $xy^iz \notin L$ for getting a contradiction. The string y can be in any of the following forms:

Case 1 y has 0's, i.e. $y = 0^k$ for some $k \ge 1$.

Case 2 y has only 1's, i.e. $y = 1^l$ for some $l \ge 1$.

Case 3 y has both 0's and 1's, i.e. $y = 0^k1^j$ for some $k, j \ge 1$.

In Case 1, we can take $i = 0$. As $xyz = 0^n1^n$, $xz = 0^{n-k}1^n$. As $k \ge 1$, $n - k \ne n$. So, $xz \notin L$.

In Case 2, take $i = 0$. As before, xz is 0^n1^{n-l} and $n \ne n - l$. So, $xz \notin L$.

In Case 3, take $i = 2$. As $xyz = 0^{n-k}0^k1^j1^{n-j}$, $xy^2z = 0^{n-k}\,0^k1^j0^k1^j1^{n-j}$. As xy^2z is not of the form 0^i1^i, $xy^2z \notin L$.

Thus in all the cases we get a contradiction. Therefore, L is not regular.

(ii)

We prove this by contradiction. Assume $L = T(M)$ for some DFA with n states. Choose $w = a^nb^nc^{3n}$ in L. Using the pumping lemma, we write $w = xyz$ with $|xy| \le n$ and $|y| > 0$. As $w = a^nb^nc^{3n}$, $xy = a^i$ for some $i \le n$. This means that $y = a^j$ for some j, $1 \le j \le n$. Then $xy^{k+1}z = a^{n+jk}b^nc^{3n}$. Choosing k large enough so that $n + jk > 2n$, we can make $n + jk + n > 3n$. So, $xy^{k+1}z \notin L$. Hence L is not regular.

b. Test the equivalence of two regular languages represented by the regular expressions

P = (a + b)* and Q = a*(b a*)* respectively. Is P = Q

Explain your answer with proper justification? (4)

Answer:

Let **P** and **Q** denote **(a + b)*** and **a*(ba*)***, respectively. Using the construction **P** is given by the transition system depicted in Fig.

Fig. Transition system for **(a + b)***.

The transition system for **Q** is depicted in Fig.

It should be noted that Figs. are obtained after eliminating Λ-moves. As these two transition diagrams are the same, we conclude that **P = Q**.

We now summarize all the results and constructions given in this section.

(i) Every r.e. is recognized by a transition system

(ii) A transition system M can be converted into a finite automaton accepting the same set as M

(iii) Any set accepted by finite automaton is represented by an r.e.

(iv) A set accepted by a transition system is represented by an r.e. (from (ii) and (iii)).

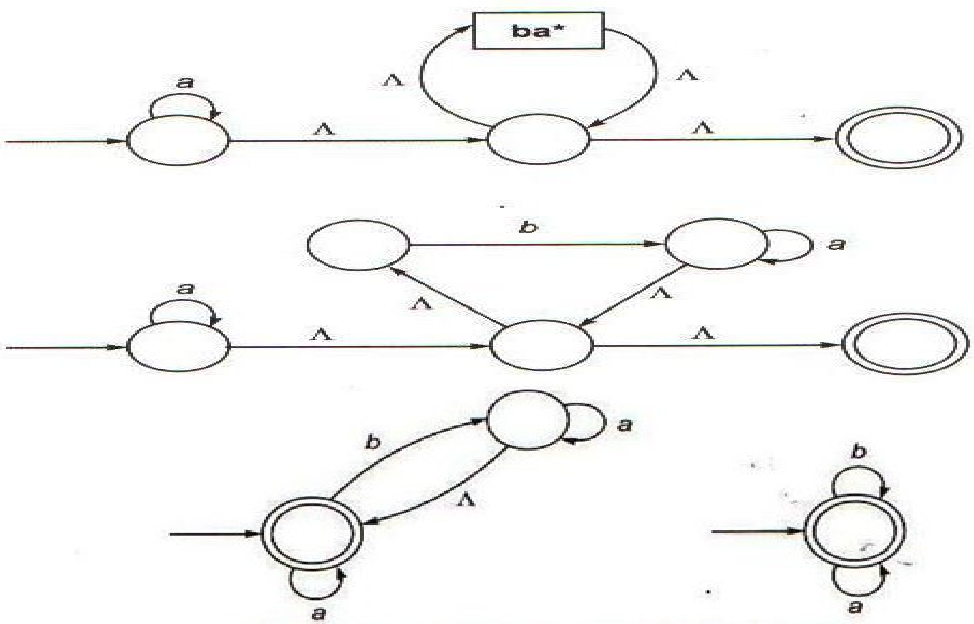


Fig. Transition system for **a*(ba*)***.

(v) To get the r.e. representing a set accepted by a transition system, we can apply the algebraic method using the Arden's theorem

(vi) If **P** is an r.e., then to construct a finite automaton accepting the set **P**, we can apply the construction

(vii) A subset L of Σ^* is a regular set (or represented by an r.e.) iff it is accepted by an FA (from (i), (ii) and (iii)).

(viii) A subset L of Σ^* is a regular set iff it is recognized by a transition system (from (i) and (iv)).

(ix) The capabilities of finite automaton and transition systems are the same as far as acceptability of subsets of strings is concerned.

c. Solve the following: (3+3)

(i) Consider the two regular Languages represented by the regular expressions P and Q respectively, where P is defined as P = (b + a a* b) and Q is any regular expression then show that

$$P + P\,Q^*Q = a^*b\,Q^*$$

(ii) Consider a context free grammar G that consists of the productions

S → 0B | 1A   A → 0 | 0S | 1AA   B → 1|1S|0BB

For the string 00110101, Find the

(i) the Leftmost Derivation (ii) the Rightmost Derivation (iii) Parse Tree

Answer: (i)

L.H.S. $= P\Lambda + P\,Q^*Q$

$= P(\Lambda + Q^*Q)$

$= PQ^*$

$= (b + aa^*b)Q^*$ (by definition of P)

$= (\Lambda b + aa^*b)Q^*$

$= a^*bQ^*$

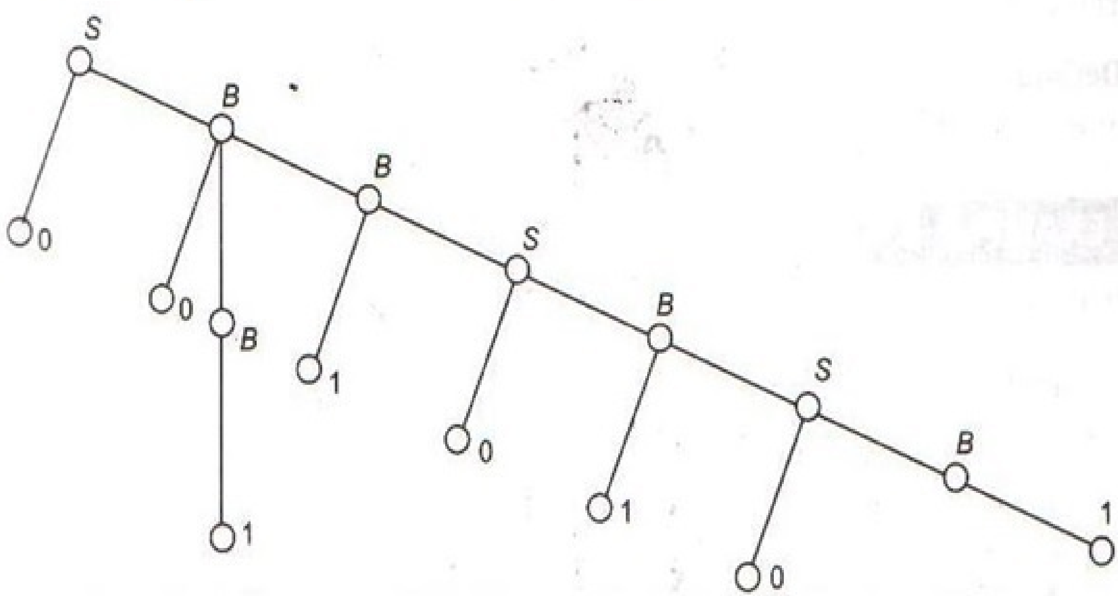
= R.H.S.

(ii)

(i) $S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 001B \Rightarrow 0011S$
$\Rightarrow 0^21^20B \Rightarrow 0^21^201S \Rightarrow 0^21^2010B \Rightarrow 0^21^20101$

(ii) $S \Rightarrow 0B \Rightarrow 00BB \Rightarrow 00B1S \Rightarrow 00B10B$
$\Rightarrow 0^2B101S \Rightarrow 0^2B1010B \Rightarrow 0^2B10101 \Rightarrow 0^2110101.$

(iii) The **parse** tree is given in Fig.



Q.6 a. Solve the following: (3+2)

(i) Make a deterministic PDA by Final State that accepts the language L = { w ϵ {a, b}*|the number of a's in w equal to the no. of b's in w}.

(ii) Convert the context free grammar S → aSb | A , A → bSa | S | ϵ to a equivalent PDA by empty stack.

Answer: (i)

We define a pda M as follows:

$$M = (\{q_0, q_1\}, \{a, b\}, \{a, b, Z_0\}, \delta, q_0, Z_0, \{q_1\})$$

where δ is defined by

$$\delta(q_0, a, Z_0) = \{(q_1, Z_0)\}$$
$$\delta(q_0, b, Z_0) = \{(q_0, bZ_0)\}$$
$$\delta(q_0, a, b) = \{(q_0, \Lambda)\}$$
$$\delta(q_0, b, b) = \{(q_0, bb)\}$$
$$\delta(q_1, a, Z_0) = \{(q_1, aZ_0)\}$$
$$\delta(q_1, b, Z_0) = \{(q_0, aZ_0)\}$$
$$\delta(q_1, a, a) = \{(q_1, aa)\}$$
$$\delta(q_1, b, a) = \{(q_1, \Lambda)\}$$

The construction can be explained as follows:

If the pda M is in the final state q_1, it means it has seen more a's than b's. On seeing the first a, M changes state (from q_0 to q_1) Afterwards it stores the a's in PDS without changing state It stores the initial b in PDS and also the subsequent b's The pda cancels a in the input string, with the first (topmost) b in PDS If all b's are matched with stored a's, and M sees the bottom of PDS, M moves from q_1 to q_0 The b's in the input string are cancelled on seeing a in the PDS

M is deterministic since δ is not defined for input Λ. The reader is advised to check that q_1 is reached on seeing an input string w in L.

(ii)

We construct a pda A as

$$A = (\{q\}, \{a, b\}, \{S, A, a, b\}, \delta, q, S, \emptyset)$$

where δ is defined by the following rules

$$\delta(q, \Lambda, S) = \{(q, aSb), (q, A)\}$$
$$\delta(q, \Lambda, A) = \{(q, bSA), (q, S), (q, \Lambda)\}$$
$$\delta(q, a, a) = \{(q, \Lambda)\}$$
$$\delta(q, b, b) = \{(q, \Lambda)\}$$

and A is the required pda.

b. Construct a Pushdown Automata (PDA) that accepts the language L defined as: L = {a^n b^m a^n | m, n ≥ 1} by empty stack. Also make the corresponding CFG productions accepting the same set or language. (8)

Answer:

The pda A accepting $\{a^n b^m a^n \mid m, n \geq 1\}$ is defined as follows:

$$A = (\{q_0, q_1\}, \{a, b\}, \{a, Z_0\}, \delta, q_0, Z_0, \emptyset)$$

where δ is defined by

$$R_1: \delta(q_0, a, Zo) = \{(q_0, aZ_0)\}$$
$$R_2: \delta(q_0, a, a) = \{(q_0, aa)\}$$
$$R_3: \delta(q_0, b, a) = \{(q_1, a)\}$$
$$R_4: \delta(q_1, b, a) = \{(q_1, a)\}$$
$$R_5: \delta(q_1, a, a) = \{(q_1, \Lambda)\}$$
$$R_6: \delta(q_1, \Lambda, Z_0) = \{(q_1, \Lambda)\}$$

We start storing a's until a b occurs (Rules R_1 and R_2). When the current input symbol is b, the state changes, but no change in PDS occurs (Rule R_3). Once all the b's in the input string are exhausted (using Rule R_4), the remaining a's are erased (Rule R_5). Using R_6, Z_0 is erased. So,

$$(q_0, a^n b^m a^n, Z_0) \overset{*}{\vdash} (q_1, \Lambda, Z_0) \vdash (q_1, \Lambda, \Lambda)$$

This means that $a^n b^m a^n \in N(A)$. We can show that

$$N(A) = \{a^n b^m a^n \mid m, n \geq 1\}$$

by using Rules R_1–R_6.

Define $G = (V_N, \{a, b\}, P, S)$, where V_N consists of

$$[q_0, Z_0, q_0], [q_1, Z_0, q_0], [q_0, a, q_0], [q_1, a, q_0]$$
$$[q_0, Z_0, q_1], [q_1, Z_0, q_1], [q_0, a, q_1], [q_1, a, q_1]$$

The productions in P are constructed as follows:

The S-productions are

$$P_1: S \to [q_0, Z_0, q_0], \qquad P_2: S \to [q_0, Z_0, q_1]$$

$\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$ induces

$$P_3: [q_0, Z_0, q_0] \to a[q_0, a, q_0][q_0, Z_0, q_0]$$
$$P_4: [q_0, Z_0, q_0] \to a[q_0, a, q_1][q_1, Z_0, q_0]$$
$$P_5: [q_0, Z_0, q_1] \to a[q_0, a, q_0][q_0, Z_0, q_1]$$
$$P_6: [q_0, Z_0, q_1] \to a[q_0, a, q_1][q_1, Z_0, q_1]$$

$\delta(q_0, a, a) = \{(q_0, aa)\}$ yields

$$P_7: [q_0, a, q_0] \rightarrow a[q_0, a, q_0][q_0, a, q_0]$$
$$P_8: [q_0, a, q_0] \rightarrow a[q_0, a, q_1][q_1, a, q_0]$$
$$P_9: [q_0, a, q_1] \rightarrow a[q_0, a, q_0][q_0, a, q_1]$$
$$P_{10}: [q_0, a, q_1] \rightarrow a[q_0, a, q_1][q_1, a, q_1]$$

$\delta(q_0, b, a) = (q_1, a)\}$ gives

$$P_{11}: [q_0, a, q_0] \rightarrow b[q_1, a, q_0]$$
$$P_{12}: [q_0, a, q_1] \rightarrow b[q_1, a, q_1]$$

$\delta(q_1, b, a) = \{(q_1, a)\}$ yields

$$P_{13}: [q_1, a, q_0] \rightarrow b[q_1, a, q_0]$$
$$P_{14}: [q_1, a, q_1] \rightarrow b[q_1, a, q_1]$$

$\delta(q_1, a, a) = \{(q_1, \Lambda)\}$ gives

$$P_{15}: [q_1, a, q_1] \rightarrow a$$

$\delta(q_1, \Lambda, Z_0) = \{(q_1, \Lambda)\}$ yields

$$P_{16}: [q_1, Z_0, q_1] \rightarrow \Lambda$$

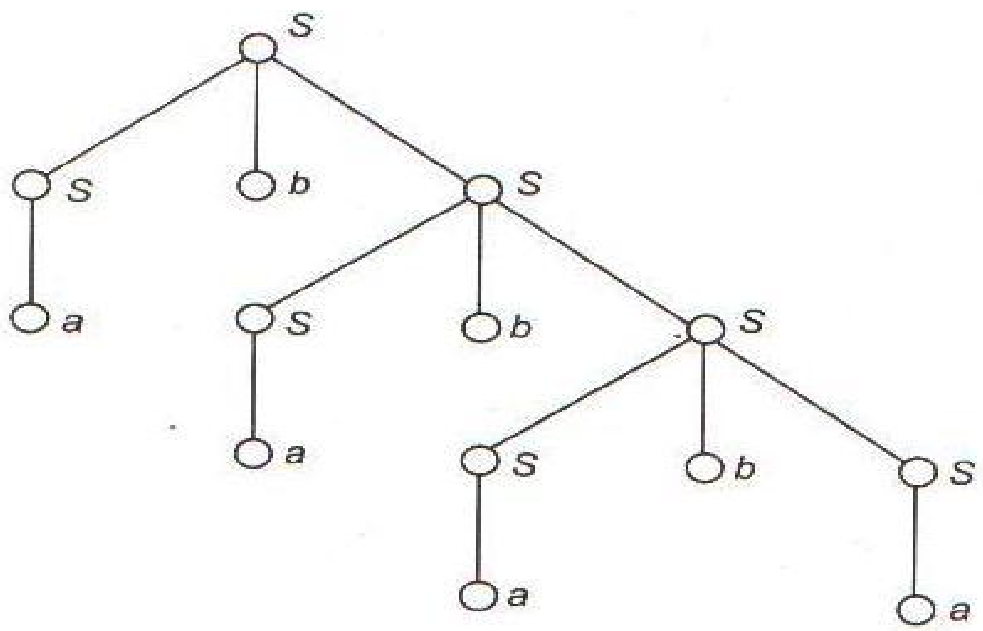
c. **Define Inherent Ambiguity and Inherently Ambiguous Languages. If G is a context free grammar whose production rules are given as S → SbS | a. Check whether the given grammar G is ambiguous or inherently ambiguous. (3)**

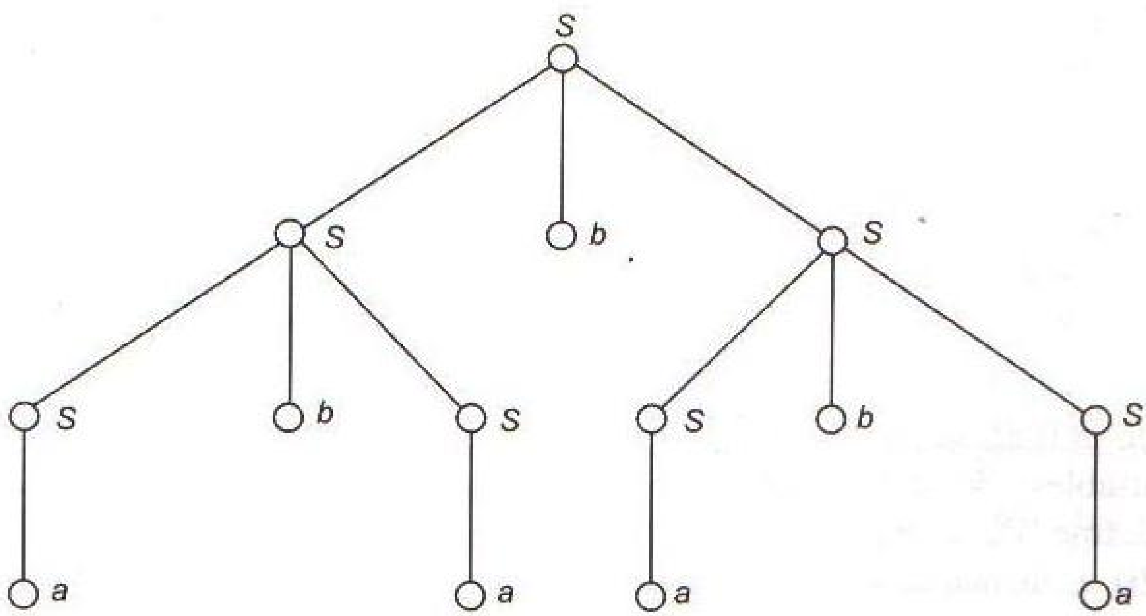
Answer:

Inherent Ambiguity and Inherently Ambiguous Languages

The languages generated by a grammar, that have both ambiguous and unambiguous grammars but there exist languages for which no unambiguous grammar can exist. Such types of languages are called inherently ambiguous languages and the property is known as inherent ambiguity.

To prove that G is ambiguous, we have to find a $w \in L(G)$, which is ambiguous. Consider $w = ababa ba \in L(G)$. Then we get two derivation trees for w Thus, G is ambiguous.


S
S
b
S
a
S
b
S
a
S
b
S
a
a


S
S
b
S
S
b
S
S
b
S
a
a
a
a

Q.7 a. Consider a context free grammar G whose production rules are defined as S→ASA | bA, A→B | S, B→c. Reduce it into Chomsky Normal Form (CNF). (6)

Answer:

Step 1 *Elimination of unit productions:*

The unit productions are $A \to B$, $A \to S$.

$$W_0(S) = \{S\},\ W_1(S) = \{S\} \cup \emptyset = \{S\}$$

$$W_0(A) = \{A\},\ W_1(A) = \{A\} \cup \{S, B\} = \{S, A, B\}$$

$$W_2(A) = \{S, A, B\} \cup \emptyset = \{S, A, B\}$$

$$W_0(B) = \{B\},\ W_1(B) = \{B\} \cup \emptyset = \{B\}$$

The productions for the equivalent grammar without unit productions are

$$S \to ASA \mid bA,\ B \to c$$

$$A \to ASA \mid bA,\ A \to c$$

So, $G_1 = (\{S, A, B\}, \{b, c\}, P, S)$ where P consists of $S \to ASA \mid bA$, $B \to c$, $A \to ASA \mid bA \mid c$.

Step 2 *Elimination of terminals in R.H.S.:*

$S \to ASA$, $B \to c$, $A \to ASA \mid c$ are in proper form. We have to modify $S \to bA$ and $A \to bA$.

Replace $S \to bA$ by $S \to C_bA$, $C_b \to b$ and $A \to bA$ by $A \to C_bA$, $C_b \to b$.

So, $G_2 = (\{S, A, B, C_b\}, \{b, c\}, P_2, S)$ where P_2 consists of

$$S \to ASA \mid C_bA$$

$$A \to ASA \mid c \mid C_bA$$

$$B \to c,\ C_b \to b$$

Step 3 *Restricting the number of variables on R.H.S.:*

$$S \to ASA \text{ is replaced by } S \to AD,\ D \to SA$$

$$A \to ASA \text{ is replaced by } A \to AE,\ E \to SA$$

So the equivalent grammar in CNF is

$$G_3 = (\{S, A, B, C_b, D, E\}, \{b, c\}, P_3, S)$$

where P_3 consists of

$$S \to C_bA \mid AD$$

$$A \to c \mid C_bA \mid AE$$

$$B \to c,\ C_b \to b,\ D \to SA,\ E \to SA$$

b. Reduce the given CFG defined as S→aAbB, A→aA | a, B→bB | b into Chomsky Normal form. (6)

Answer:

As there are no unit productions or null productions, we need not carry out step 1. We proceed to step 2.

Step 2 Let $G_1 = (V'_N \{a, b\}, P_1, S)$, where P_1 and V'_N are constructed as follows:

(i) $A \to a$, $B \to b$ are added to P_1.

(ii) $S \to aAbB$, $A \to aA$, $B \to bB$ yield $S \to C_aAC_bB$, $A \to C_aA$, $B \to C_bB$, $C_a \to a$, $C_b \to b$.

$$V'_N = \{S, A, B, C_a, C_b\}.$$

Step 3 P_1 consists of $S \to C_aAC_bB$, $A \to C_aA$, $B \to C_bB$, $C_a \to a$, $C_b \to b$, $A \to a$, $B \to b$.

$S \to C_aAC_bB$ is replaced by $S \to C_aC_1$, $C_1 \to AC_2$, $C_2 \to C_bB$

The remaining productions in P_1 are added to P_2. Let

$$G_2 = (\{S, A, B, C_a, C_b, C_1, C_2\}, \{a, b\}, P_2, S),$$

where P_2 consists of $S \to C_aC_1$, $C_1 \to AC_2$, $C_2 \to C_bB$, $A \to C_aA$, $B \to C_bB$, $C_a \to a$, $C_b \to b$, $A \to a$, and $B \to b$.

G_2 is in CNF and equivalent to the given grammar.

c. Check whether the language defined as L = {a^p | p is a prime} is a context free language or not. Justify your answer by using the help of Pumping Lemma. (4)

Answer:

We use the following property of L: If $w \in L$, then $|w|$ is a prime.

Step 1 Suppose $L = L(G)$ is context-free. Let n be the natural number obtained by using the pumping lemma.

Step 2 Let p be a prime number greater than n. Then $z = a^p \in L$. We write $z = uvwxy$.

Step 3 By pumping lemma, $uv^0wx^0y = uwy \in L$. So $|uwy|$ is a prime number, say q. Let $|vx| = r$. Then, $|uv^qwx^qy| = q + qr$. As $q + qr$ is not a prime, $uv^qwx^qy \notin L$. This is a contradiction. Therefore, L is not context-free.

Q.8 a. Design Turing Machines that recognizes the following languages: (5+5)
(i) L = {$0^n 1^n$ | n≥ 1} and
(ii) Set L of all strings over {0, 1} ending with 010

Answer:

We require the following moves:

(a) If the leftmost symbol in the given input string w is 0, replace it by x and move right till we encounter a leftmost 1 in w. Change it to y and move backwards.

(b) Repeat (a) with the leftmost 0. If we move back and forth and no 0 or 1 remains, move to a final state.

(c) For strings not in the form 0^n1^n, the resulting state has to be nonfinal.

Keeping these ideas in our mind, we construct a TM M as follows:

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

where

$$Q = \{q_0, q_1, q_2, q_3, q_f\}$$
$$F = \{q_f\}$$
$$\Sigma = \{0, 1\}$$
$$\Gamma = \{0, 1, x, y, b\}$$

The transition diagram is given in Fig. M accepts $\{0^n1^n \mid n \geq 1\}$. The moves for 0011 and 010 are given below just to familiarize the moves of M to the reader.

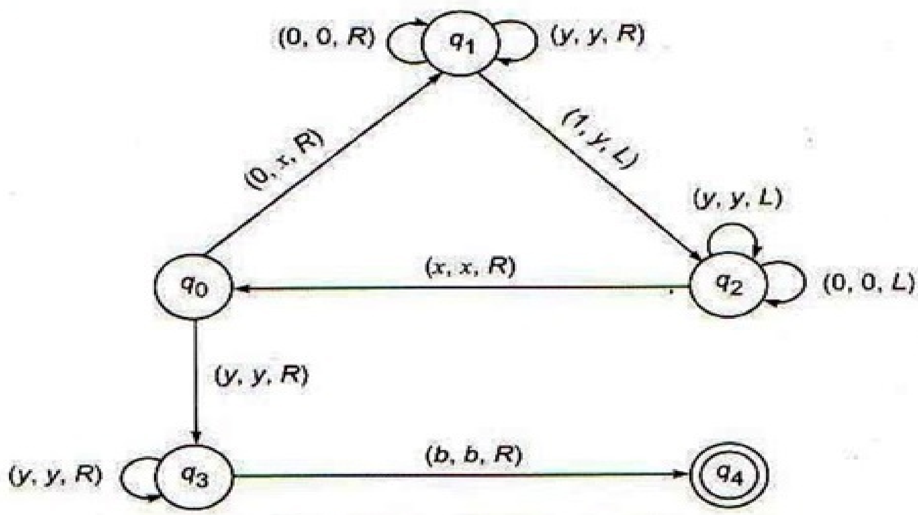


Fig. Transition diagram

$$q_0 0011 \vdash xq_1 011 \vdash x0q_1 11 \vdash xq_2 0y1$$
$$\vdash q_2 x0y1 \vdash xq_0 0y1 \vdash xxq_1 y1 \vdash xxyq_1 1$$
$$\vdash xxq_2 yy \vdash xq_2 xyy \vdash xxq_0 yy \vdash xxyq_3 y$$
$$\vdash xxyyq_3 = xxyyq_3 b \vdash xxyybq_4 b$$

Hence 0011 is accepted by M.

$$q_0 010 \vdash xq_1 10 \vdash q_2 xy0 \vdash xq_0 y0 \vdash xyq_3 0$$

As $\delta(q_3, 0)$ is not defined, M halts. So 010 is not accepted by M.

Sol (ii):

L is certainly a regular set and hence a deterministic automaton is sufficient to recognize L. Figure gives a DFA accepting L.

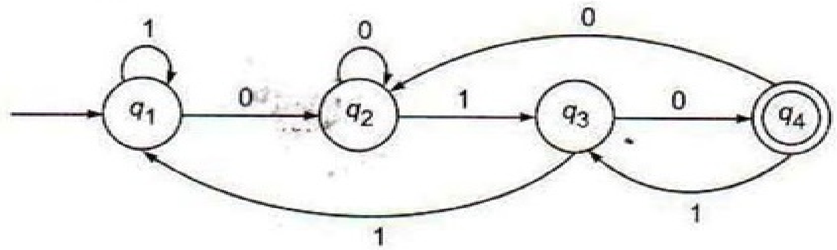
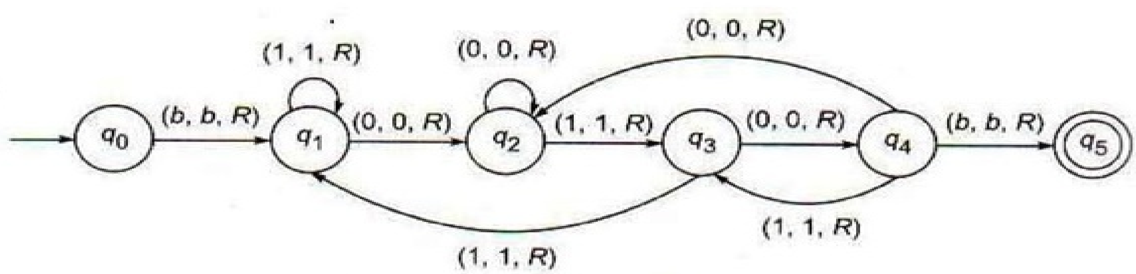


Fig. DFA **for given problem**

Converting this DFA to a TM is simple. In a DFA M, the move is always to the right. So the TM's move will always be to the right. Also M reads the input symbol and changes state. So the TM M_1 does the same; it reads an input symbol, does not change the symbol and changes state. At the end of the computation, the TM sees the first blank b and changes to its final state. The initial ID of M_1 is q_0w. By defining $\delta(q_0, b) = (q_1, b, R)$, M_1 reaches the initial state of M. M_1 can be described by Fig.



b. Define a Turing machine. Construct a Turing Machine that accepts the language given by the expression (0 1* + 1 0*) (6)

Answer:

Definition of Turing Machine

A Turing Machine *M* is a collection of 7-tuples as:

$$(Q, \Sigma, \Gamma, \delta, q_0, \#, F)$$

where Q is a set of states

Σ is a finite set of symbols, "input alphabet".

Γ is a finite set of symbols, "tape alphabet".

δ is the partial transition function

$\# \in T$ is a symbol called 'blank'

$q_0 \in Q$ is the initial state

$F \subseteq Q$ is a set of final states

As the Turing machine will have to be able to find its input, and to know when it has processed all of that input, we require:

(a) The tape is initially "blank" (every symbol is #) except possibly for a finite, contiguous sequence of symbols.
(b) If there are initially nonblank symbols on the tape, the tape head is initially positioned on one of them.
This emphasises the fact that the "input" viz., the non-blank symbols on the tape does not contain #.

We have to construct a TM that remembers the first symbol and checks that it does not appear afterwards in the input string. So we require two states, q_0, q_1. The tape symbols are 0, 1 and b. So the TM, having the 'storage facility in state', is

$$M = (\{q_0, q_1\} \times \{0, 1, b\}, \{0, 1\}, \{0, 1, b\}, \delta, [q_0, b], \{[q_1, b]\})$$

We describe δ by its implementation description.

1. In the initial state, M is in q_0 and has b in its data portion. On seeing the first symbol of the input sting w, M moves right, enters the state q_1 and the first symbol, say a, it has seen.
2. M is now in $[q_1, a]$. (i) If its next symbol is b, M enters $[q_1, b]$, an accepting state. (ii) If the next symbol is a, M halts without reaching the final state (i.e. δ is not defined). (iii) If the next symbol is $\overline{a}$ ($\overline{a} = 0$ if $a = 1$ and $\overline{a} = 1$ if $a = 0$), M moves right without changing state.
3. Step 2 is repeated until M reaches $[q_1, b]$ or halts (δ is not defined for an input symbol in w).

Q.9 a. State the Post correspondence problem (PCP). Find at least three solutions to the PCP defined by the following sets: (8)
A = {1, 10 , 10111} and B = {111, 0, 10}

Answer:
Post correspondence problem (PCP):
An instance of PCP consists of two lines of strings over some alphabet Σ; the two lists must be equal length. We generally refer to the A and B lists, and write A = $w_1, w_2, \ldots, w_k$ and B = $x_1, x_2, \ldots, x_k$, for some integer k. For each i, the pair (w_i, x_i) is said to be a corresponding pair.
We say this instance of PCP has a solution, if there is a sequence of one or more integers $i_1, i_2, \ldots, i_m$ that, when interpreted as indexes for strings in the A and B lists, yield the same string.
That is, $w_{i1} w_{i2} \ldots w_{im} = x_{i1} x_{i2} \ldots x_{im}$.
We say the sequence $i_1, i_2, \ldots, i_m$ is a solution to this instance of PCP, if so, the Post correspondence problem is: "Given an instance of PCP, tell whether this has a solution."

Given : A = (1, 10, 10111)
B = (111, 0, 10)
From the above we conclude that

A1 = 1,	A2 = 10,	A3 = 10111
B1 = 111,	B2 = 0,	B3 = 10

Then $A_3A_1A_1A_2 = B_3B_1B_1B_2$

Hence the PCP with the given list has a solution. Repeating the sequence 3, 1, 1, 2 we can get more solutions.
As a example:
$A_3A_1A_1A_2A_3A_1A_1A_2 = B_3B_1B_1B_2B_3B_1B_1B_2 = 101111110101111110$
Similarly we can get an another solution as having the sequence 3, 1, 1, 2, 3, 1, 1, 2, 3, 1, 1, 2.
So the three solutions to the PCP defined by the given sets:

(i) 3, 1, 1, 2
(ii) 3, 1, 1, 2, 3, 1, 1, 2
(iii) 3, 1, 1, 2, 3, 1, 1, 2, 3, 1, 1, 2

b. Differentiate between Recursive Languages and Recursively Enumerable Languages. Show that if L1 and L2 are Recursively Enumerable Languages than L1 U L2 is also Recursively Enumerable as well as if L1 and L2 are Recursive Languages than L1 U L2 is also recursive. (8)

Answer:

Recursive and Recursively Enumerable language defined according to the behavior of Turing Machine. As we know that any Turing Machine performes following three outcomes at the time of executions.

(i) A Turing Machine may Halt (or terminate) and accept the input string.

(ii) A Turing Machine may Halt (or terminate) and reject the input string

(iiii) A Turing Machine never terminate i.e. Execute infinite times.

Based on these three condition, we can define the Recursive and Recursively Enumerable languages.

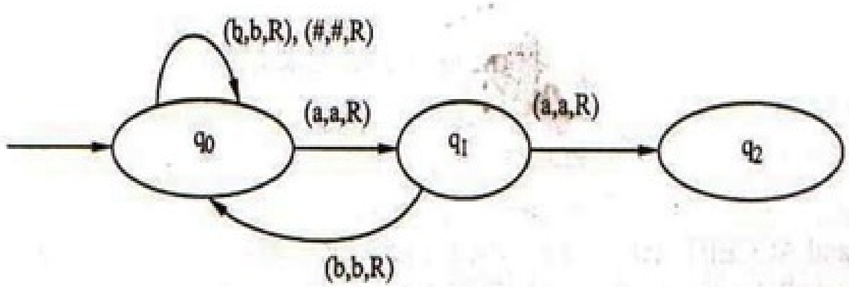
RECURSIVELY ENUMERABLE LANGUAGES

A language over the alphabet Σ is called recursively enumerable if there exists a TM say T that can accept every word in L and either rejects (i.e crashes) or loops forever for every word in the language L′ which is complement of L which can be represented as follows:

Accept (T) = L

Reject (T) + Loop (T) = L′

Example : *Consider the TM given below:*



It divides all inputs into three parts.

Accept (T) = all words with aa

Reject (T) = strings without ending aa

Loop (T) = strings without ending aa

It mean that the language (a+b)* aa (a+b)* is recursively enumerable.

RECURSIVE LANGUAGE

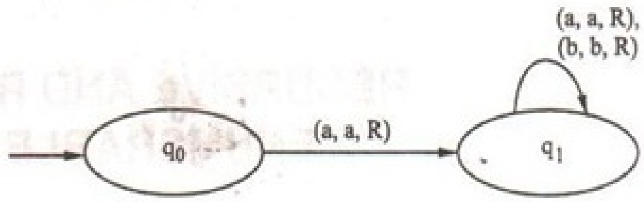
A language over the alphabet Σ is called recursive if there exists a TM say T that accepts every word in L and rejects every word in L′ the complement of L which can be represented as follows:

Accept (T) = L

Reject (T) + Loop (T) = L′

Loop (T) = ϕ

Example : *Consider the following TM*



It accepts the language of all words over $\Sigma = \{a,b\}$ that starts with a and rejects all words that do not start with a. Therefore such a language is recursive.

Note: 1. Every recursive language is recursively enumerable because the TM for recursive languages also satisfy the condition of r.e. Language but the reverse is not always true.

2. We can define recursive and recursively enumerable languages in term of PMs as well as TMs because the languages accepted by them are the same.

Let L_1 and L_2 be two recursive languages and M_1, M_2 be the corresponding TMs that halt. We design a TM M as a two-tape TM as follows:

1. w is an input string to M.
2. M copies w on its second tape.
3. M simulates M_1 on the first tape. If w is accepted by M_1, then M accepts w.
4. M simulates M_2 on the second tape. If w is accepted by M_2, then M accepts w.

M always halts for any input w.

Thus $L_1 \cup L_2 = T(M)$ and hence $L_1 \cup L_2$ is recursive.

If L_1 and L_2 are recursively enumerable, then the same conclusion gives a proof for $L_1 \cup L_2$ to be recursively enumerable. As M_1 and M_2 need not halt, M need not halt.