- An input 0 in state q_2, will make number of 0's even.

$$\delta(q_2, 0) \Rightarrow q_0$$

- An input 1 in state q_2, will make number of 1's odd.

$$\delta(q_2, 1) \Rightarrow q_3$$

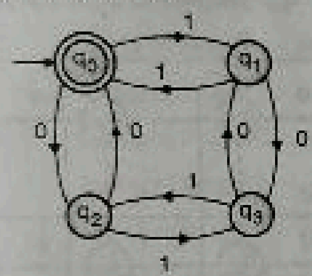
- An input 0 in state q_3, will make number of 0's even.

$$\delta(q_3, 0) \Rightarrow q_1$$

- An input 1 in state q_3, will make number of 1's even.

$$\delta(q_3, 0) \Rightarrow q_2$$

- q_0 is the starting state. An empty string contains even number of 0's and even number of 1's.
- q_0 is a final state. q_0 stands for even number of 0's and even number of 1's.



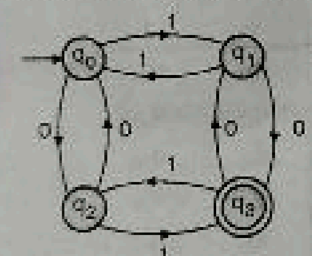
	0	1
$\rightarrow q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

(a) Transition diagram (b) Transition table

Fig. Ex. 2.2.2 : Final DFA for Example 2.2.2(a)

(b) Number of 1's is odd and number of 0's is odd.

In solution of Example 2.2.2(a), the state q_1 stands for odd number of 0's should be declared as final state.



	0	1
$\rightarrow q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3^*	q_1	q_2

(c) Transition diagram (d) Transition table

Fig. Ex. 2.2.2 : Final DFA for Example 2.2.2(b)

Example 2.2.3 : Design a finite state machine to accept following language over the alphabet {0, 1} L(R) = {w | w starts with 0 and has odd length or starts with 1 and has even length}.

MU - May 19, 10 Marks

Solution :

The required DFA is as given in Fig. Ex. 2.2.3.

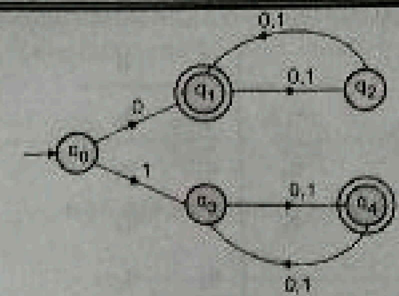


Fig. Ex. 2.2.3

State $q_1 \rightarrow$ strings starting with 0 and has odd length

State $q_4 \rightarrow$ strings starting with 1 and has even length.

Example 2.2.4 : Design a DFA which accepts the odd number 1's and any number of 0's over $\sum = \{0, 1\}$.

Solution :

The DFA must keep track of number of 1's in the string already seen by it.

- The number of 1's seen could be even, state q_0.
- The number of 1's seen could be odd, state q_1.
- The required DFA is given in Fig. Ex. 2.2.4.

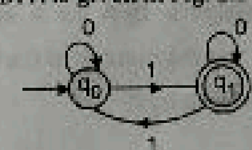


Fig. Ex. 2.2.4

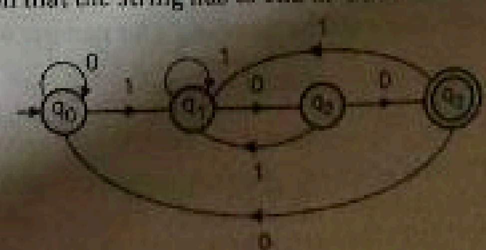
Example 2.2.5 : Design minimized DFA for accepting strings ending with 100 over alphabet (0,1).

MU - May 15, 10 Marks

Solution :

All strings ending in 100

The substring '100' should be at the end of the string. Transitions from q_3 should be modified to handle the condition that the string has to end in '100'.



(a) State transition diagram

Fig. Ex. 2.2.5contd...

	1	0
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
q_3^*	q_1	q_0

(b) State transition table

Fig. Ex. 2.2.5

q_3 to q_1 on input 1 :

An input of 1 in q_3 will make the previous four characters as '1001'. Out of the four characters as '1001' only the last character '1' is relevant to '100'.

q_3 to q_0 on input 0 :

An input of 0 in q_3 will make the previous four characters '1000'. Out of the four characters '1000', nothing is relevant to '100'.

Example 2.2.6 : Design a DFA for a set of strings over alphabet {0, 1} such that the number of 0's is divisible by five, and number of 1's divisible by 3.

Solution :

At any instance of time, we will have following cases for number of 0's.

Case 1 – 5n

Case 2 – 5n + 1

Let us represe
as q_{ij}, where i can
number of 0's seen

Similarly, in
depending on nu

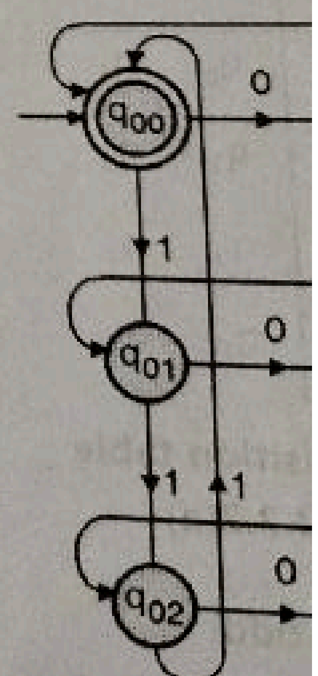


Fig. E

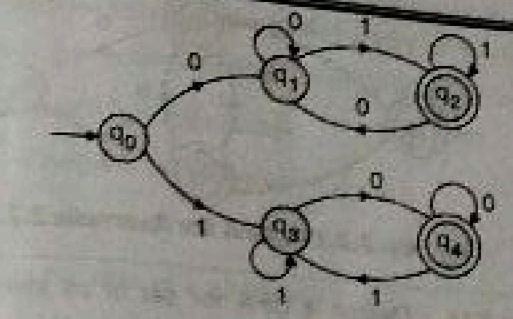
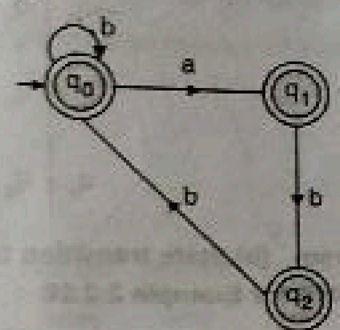


Fig. Ex. 2.2.12 : Final DFA for Example 2.2.12

Example 2.2.13 : Design a DFA for set of strings over {a, b} in which there are at least two occurrences of b between any two occurrences of a.

Solution :



	a	b
$\rightarrow q_0^*$	q_1	q_0
q_1^*	ϕ	q_2
q_2^*	ϕ	q_0

(a) State transition diagram (b) State transition table

Fig. Ex. 2.2.13 : Final DFA (without explicit failure state) for Example 2.2.13

- An input 'a' in q_0 takes the machine from q_0 to q_1. Before the next 'a' can come, there should be at least two b's taking the machine from q_1 to q_2 and from q_2 to q_0.
- An input 'a' in either q_1 or q_2 causes a failure.
- All the three states are 'accepting states'.

Example 2.2.14 : Design a DFA for set of all strings over {a, b} ending in either ab or ba.

Solution :

Meaning of different states :

$q_0 \rightarrow$ starting state

$q_1 \rightarrow$ a of sequence ab

$q_2 \rightarrow$ ab of sequence ab

$q_3 \rightarrow$ b of sequence ba

$q_4 \rightarrow$ ba of sequence ba

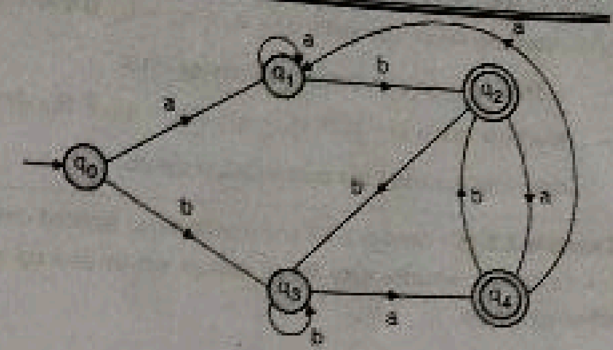


Fig. Ex. 2.2.14 : DFA for Example 2.2.14

Transitions

- Input a in q_0 takes the machine to q_1 as the first character 'a' of 'ab' is the preceding character.
- Input b in q_0 takes the machine to q_3 as the first character 'b' of 'ba' is the preceding character.
- Input 'a' in q_1 makes the preceding two characters as 'aa'. Out of 'aa', only the last character 'a' is relevant to 'ab' and hence the machine requires in q_1.
- Input 'b' in q_1 makes the preceding two characters as 'ab'. Machine enters the state q_2 which stands for previous two characters as ab.
- Input 'a' in q_3 makes the preceding two characters as 'ba'. Machine enters the state q_4 which stands for previous two characters as ba.
- Input b in q_2 makes the preceding two characters as 'bb'. Out of 'bb', only the last character 'b' is relevant to 'ba' and hence the machine enters the state q_3.
- Similar explanation can be given for q_3 and q_4.

Example 2.2.15 : Design an DFA for set of all strings over {a, b} containing both ab and ba as substrings.

Solution :

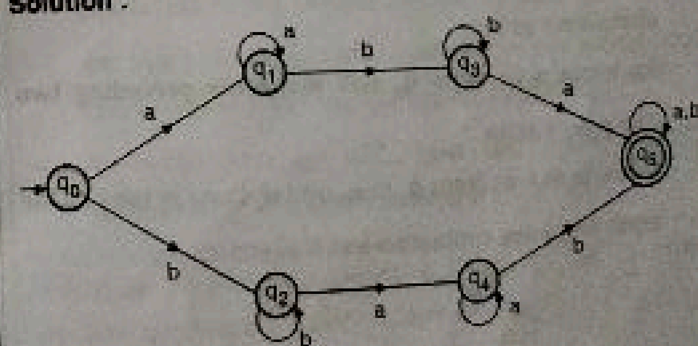


Fig. Ex. 2.2.15 : DFA for Example 2.2.15

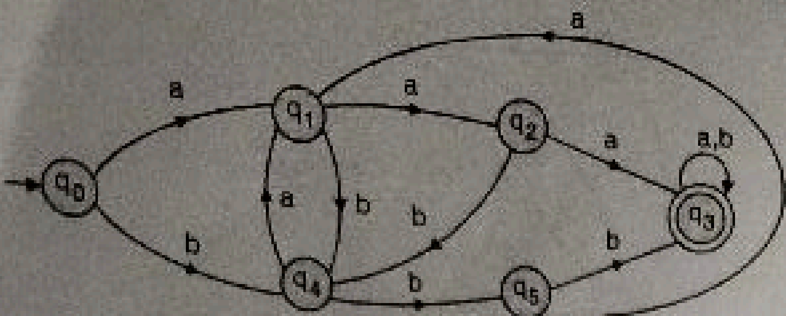


Fig. Ex. 2.2.22 : DFA for Example 2.2.22

Meaning of various states :

$q_0 \rightarrow$ Starting state.

$q_1 \rightarrow$ previous character is a of 'aaa'.

$q_2 \rightarrow$ previous two character are 'aa' of 'aaa'.

$q_4 \rightarrow$ previous character is b of 'bbb'.

$q_5 \rightarrow$ previous two character are 'bb' of 'bbb'.

$q_3 \rightarrow$ substring 'aaa' or 'bbb' is seen.

- An input 'b' in q_0, q_1 or q_2 moves the machine to q_4 as b is the first character of the sequence bbb.
- An input 'a' in q_0, q_4 or q_5 moves the machine to q_1 as 'a' is the first character of the sequence aaa.

Example 2.2.23 : Construct a DFA for accepting a set of strings over alphabet {0,1} not ending in 010.

Solution :

Above DFA can be constructed in two steps :

1. DFA for strings ending in 010.
2. By taking complement of DFA derived in step 1; make every final state as non-final state and non-finals state as final state.

Step 1 : DFA for accepting strings ending in 010.

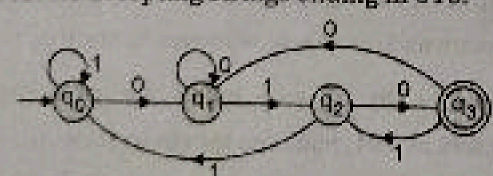


Fig. Ex. 2.2.23(a) : DFA for strings ending in 010

Step 2 : Complementing the DFA by reversing a non-final state to final state and a final state to non-final state.

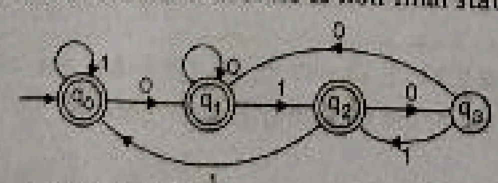


Fig. Ex. 2.2.23(b) : DFA for strings not ending in 010

Example 2.2.24 : Design a DFA that reads strings made of letters in the word 'CHARIOT' and recognizes the strings that contain the word 'CAT' as a substring.

Solution :

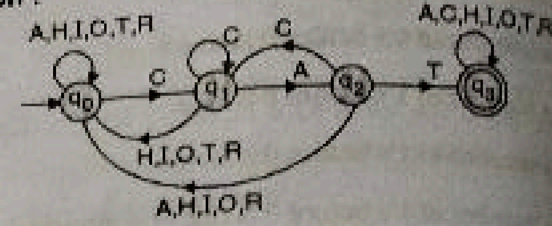


Fig. Ex. 2.2.24(a) :State transition diagram

	C	H	A	R	I	O	T
$\rightarrow q_0$	q_1	q_0	q_0	q_0	q_0	q_0	q_0
q_1	q_1	q_0	q_2	q_0	q_0	q_0	q_0
q_2	q_1	q_0	q_0	q_0	q_0	q_0	q_3
q_3^*	q_3	q_3	q_3	q_3	q_3	q_3	q_3

(b) State transition table

Fig. Ex. 2.2.24 : DFA for Example 2.2.24

Meaning of various states :

q_0 : Starting state.

q_1 : First character C of 'CAT' is the preceding character.

q_2 : First two characters CA of 'CAT' are the preceding two characters.

q_3 : entire 'CAT' has been seen.

Example 2.2.25 : Design a FA that reads strings made of letter in the word 'UNIVERSITY' and recognize these strings that contains the word UNITY as substring.

Solution :

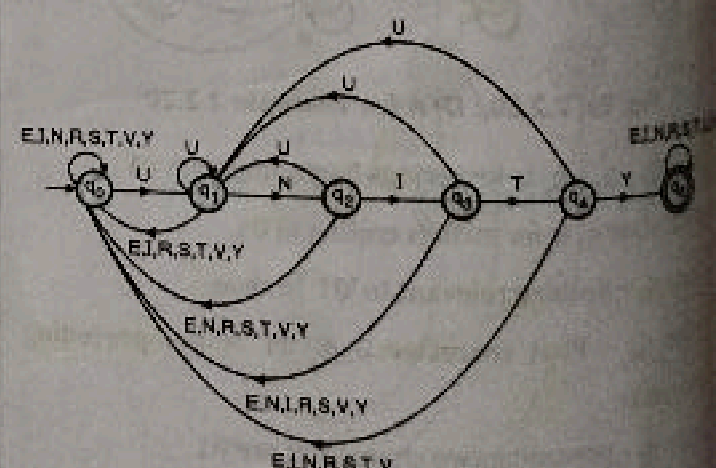


Fig. Ex. 2.2.25

Meaning of various states

q_0 : Starting state

q_1 : First character U of 'UNITY' is the preceding character.

q_2 : First two characters UN of 'UNITY' are the preceding two characters.

q_3 : First three characters UNI of 'UNITY' are the preceding three characters.

q_4 : First four characters UNIT of 'UNITY' are the preceding four characters.

q_5 : Entire 'UNITY' has been seen.

Example 2.2.26 : Design a DFA to accept string of 0's and 1's ending with the string 100.

MU - Dec. 19, 5 Marks

Solution :

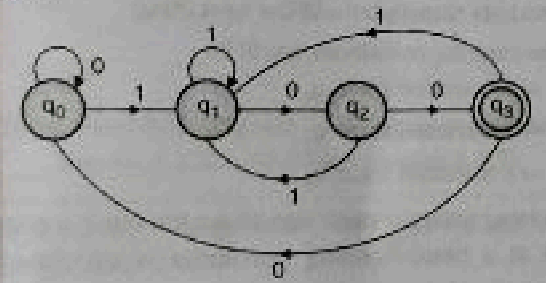


Fig. Ex. 2.2.26

Example 2.2.27 : Design a DFA over an alphabet $\Sigma = \{a, b\}$ to recognize a language in which every 'a' is followed by 'b'

MU - Dec. 16, 5 Marks

Solution. :

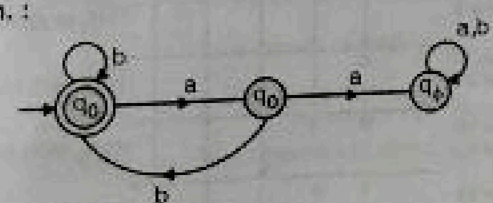


Fig. Ex. 2.2.27

- If 'a' is followed by 'a' then the machine enters the failure state q_ϕ
- A 'b' immediately after 'a' takes the machine to the accepting state q_0

Example 2.2.28 : Design the DFA to accept all the binary strings over $\Sigma = \{0, 1\}$ that are beginning with 1 and having its decimal value multiple of 5. **MU - May 16, 10 Marks**

Solution :

Running remained is maintained through the states q_0, q_1, q_2, q_3, q_4. If the number start with 0, it is rejected.

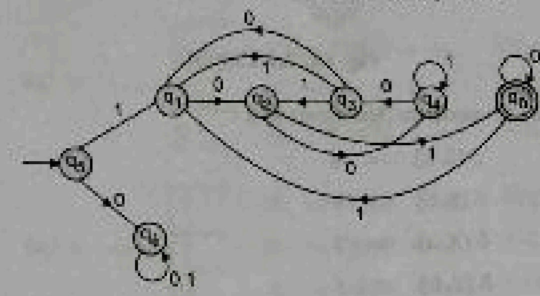


Fig. Ex. 2.2.28 : DFA

Reminder calculation for finding the next state

State	Binary value of the state	$\delta(q_i, 0)$	$\delta(q_i, 1)$
q_0	0	$00 \div 5 = 0 (q_0)$	$01 \div 5 = 1 (q_1)$
q_1	1	$10 \div 5 = 2 (q_2)$	$11 \div 5 = 3 (q_3)$
q_2	10	$100 \div 5 = 4 (q_4)$	$101 \div 5 = 0 (q_0)$
q_3	11	$110 \div 5 = 1 (q_1)$	$111 \div 5 = 2 (q_2)$
q_4	100	$1000 \div 5 = 3 (q_3)$	$1001 \div 5 = 4 (q_4)$

The operator ÷ is for reminder.

Example 2.2.29 : Convert the following grammar into finite automata. **MU - Dec. 15, 5 Marks**

$S \rightarrow aX \mid bY \mid a \mid b$

$X \rightarrow aS \mid bY \mid b$

$Y \rightarrow aX \mid bS$

Solution :

The above grammar can be converted to FA as follows :

For every non terminating symbol we consider it as a different state

$$M = \{Q, \Sigma, \delta, S, F\}$$
$$Q = \{S, X, Y\}$$
$$\Sigma = \{a, b\}$$
$$S = \text{initial state}$$
$$F = \{X, Y\}$$

Meaning of various states

q_0 : Starting state

q_1 : First character U of 'UNITY' is the preceding character.

q_2 : First two characters UN of 'UNITY' are the preceding two characters.

q_3 : First three characters UNI of 'UNITY' are the preceding three characters.

q_4 : First four characters UNIT of 'UNITY' are the preceding four characters.

q_5 : Entire 'UNITY' has been seen.

Example 2.2.26 : Design a DFA to accept string of 0's and 1's ending with the string 100.

MU - Dec. 19, 5 Marks

Solution :

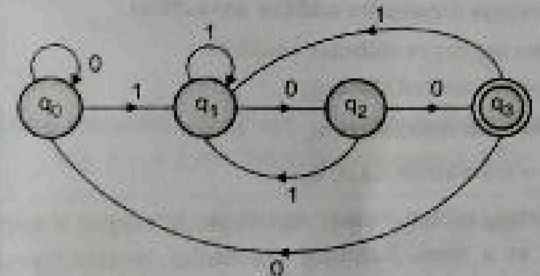


Fig. Ex. 2.2.26

Example 2.2.27 : Design a DFA over an alphabet $\Sigma = \{a, b\}$ to recognize a language in which every 'a' is followed by 'b'

MU - Dec. 16, 5 Marks

Solution. :

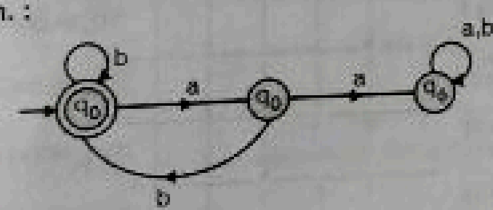


Fig. Ex. 2.2.27

- If 'a' is followed by 'a' then the machine enters the failure state q_ϕ
- A 'b' immediately after 'a' takes the machine to the accepting state q_0

Example 2.2.28 : Design the DFA to accept all the binary strings over $\Sigma = \{0, 1\}$ that are beginning with 1 and having its decimal value multiple of 5. **MU - May 16, 10 Marks**

Solution :

Running remained is maintained through the states q_0, q_1, q_2, q_3, q_4. If the number start with 0, it is rejected.

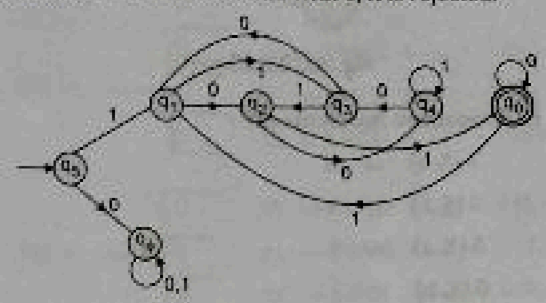


Fig. Ex. 2.2.28 : DFA

Reminder calculation for finding the next state

State	Binary value of the state	$\delta(q_i, 0)$	$\delta(q_i, 1)$
q_0	0	$00 \div 5 = 0 (q_0)$	$01 \div 5 = 1 (q_1)$
q_1	1	$10 \div 5 = 2 (q_2)$	$11 \div 5 = 3 (q_3)$
q_2	10	$100 \div 5 = 4 (q_4)$	$101 \div 5 = 0 (q_0)$
q_3	11	$110 \div 5 = 1 (q_1)$	$111 \div 5 = 2 (q_2)$
q_4	100	$1000 \div 5 = 3 (q_3)$	$1001 \div 5 = 4 (q_4)$

The operator $\div$ is for reminder.

Example 2.2.29 : Convert the following grammar into finite automata. **MU - Dec. 15, 5 Marks**

$S \rightarrow aX \mid bY \mid a \mid b$

$X \rightarrow aS \mid bY \mid b$

$Y \rightarrow aX \mid bS$

Solution :

The above grammar can be converted to FA as follows :

For every non terminating symbol we consider it as a different state

$$M = (Q, \Sigma, \delta, S, F)$$
$$Q = \{S, X, Y\}$$
$$\Sigma = \{a, b\}$$
$$S = \text{initial state}$$
$$F = \{X, Y\}$$

Example 2.2.36 : Design a Finite State Machine for divisibility by 5 tester of a given decimal number.

Solution :

A decimal number will be divisible by 5 if the rightmost digit is either '0' or '5'.

The required DFA is given in Fig. Ex. 2.2.36.

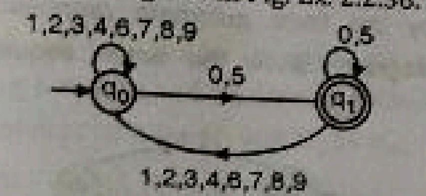


Fig. Ex. 2.2.36

Example 2.2.37 : Design DFA that accepts the following language : (i) Set of all strings with odd number of 1's followed by even number of 0's $\Sigma = \{0, 1\}$. (ii) Set of all strings which begin and end with different letters $\Sigma = \{x, y, z\}$. (iii) Strings ending with 110 or 111.

Solution :

(i)

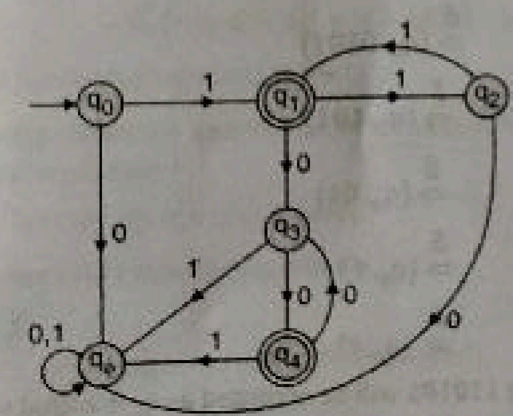


Fig. Ex. 2.2.37(a)

(ii)

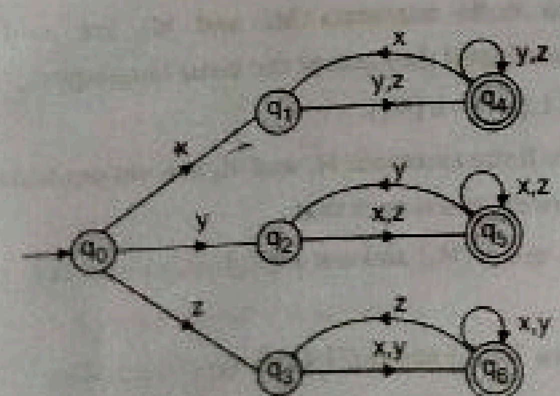


Fig. Ex. 2.2.37(b)

(iii)

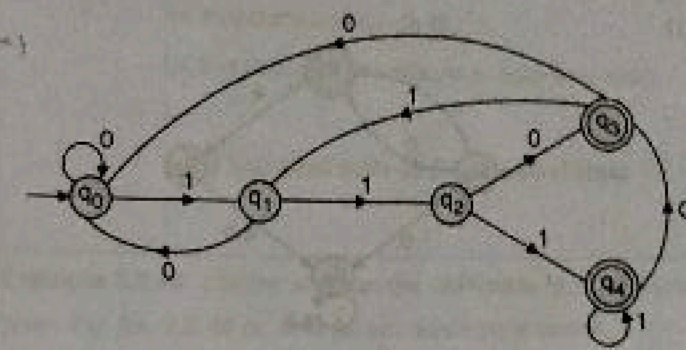


Fig. Ex. 2.2.37(c)

2.2.5 Language of DFA

The language of a DFA $M = \{Q, \Sigma, \delta, q_0, F\}$ is denoted by L (M) and is defined by :

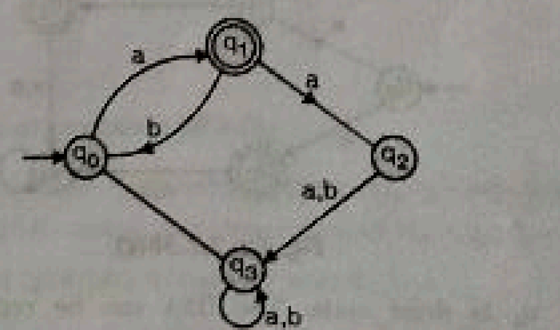
$L(M) = \{\omega \mid \delta^*(q_0, \omega) \text{ is in } F\}$

That is, the language of DFA M is the set of strings accepted by M.

The language of a DFA is also known as regular language. $\delta^*(q_0, \omega)$ stands for a series of transitions starting from q_0.

Example 2.2.38 : Describe the language accepted by the deterministic finite automata shown in Fig. Ex. 2.2.38.

(i)



(ii)

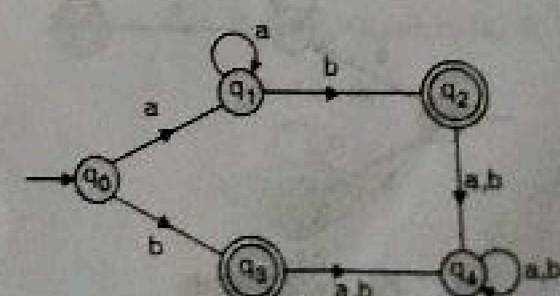


Fig. Ex. 2.2.38