**Theory of Computation (4 credit 3L +1T)**

W**hat is Theory of Computation**?

In theoretical computer science, the **theory of computation** is the branch that deals with **whether and how efficiently problems** can be solved on a **model of computation, using an algorithm**.

The field is divided into several branches, like **Automata theory, Computability theory, and Computational complexity theory.**

In order to perform a rigorous study of computation, computer scientists’ work with a mathematical abstraction of computers called a model of computation.

There are several models in use, but the most commonly examined is the Turing machine.

**Automata theory**

In theoretical computer science, **automata theory** is the study of abstract 'mathematical' machines or systems and the computational problems that can be solved using these machines. **These abstract machines are called automata**.

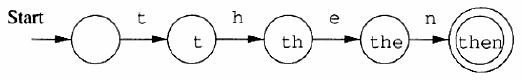
This automaton consists of

**states** (represented in the figure by circles),

and **transitions** (represented by arrows).

As the automaton sees a symbol of input, it makes a **transition**to another state, according to its **transition function**(which takes the current state and the recent symbol as its inputs).

**Fig:- A finite automaton modeling recognition of “then”**

****

TOC Course Outline:

Theory of computation course illustrates various computational models to perform scientific calculations. This course describes how to design mathematical function which admits an algorithm. It describes various kind of automata which are mathematical models to accept or reject sentences belongs to various kinds of formal languages.

|  |
| --- |
| **Course Objective:** The objective of this course is to understand the concept of automata models, to identify the limitation of the proposed model and try to find the models to overcome the limitations. |
| **Course Outcomes**: On completion this course, students will be able to  **CO1:** To learn computing paradigms as membership, design models for various classes of sets.  **CO2:** Able to apply the proposed models to find the performance evaluation of scientific computations.  **CO3:** To be able to design various machines/models for mathematical calculations.  **CO4:** To apply the automata models concepts in development of real-life applications. |

|  |  |
| --- | --- |
| Unit –I | Introduction of Automata theory, languages, string, basic operations on language, union, concatenation, Kleene Star, types of grammar and their equivalent models, Chomsky classification, regular, context free & context sensitive languages. |
| Unit-II | Regular languages, deterministic and non-deterministic finite automata, minimization of automata, equivalence of deterministic and non-deterministic machine, Melay and Moore model, regular expressions, regular grammars, properties of regular languages. |
| Unit-III | Context Free Grammar, derivation trees, simplification of context free grammar, Chomsky normal form (CNF), Greibach normal form (GNF), pushdown automata (PDA), null store and final state PDA and their equivalence, CFG to PDA. |
| Unit-IV | Turing machines, grammars, recursive and recursive enumerable languages, examples of recursive and recursive enumerable languages, language acceptability, decidability, halting problem. |

Reference and text books:

1. Mishra K. L. P., Chandrasekaran N., Theory of Computer Science, Automata, Languages and Computation.

2. Hofcroft J.E., Ullman J.D., Introduction to Automata Theory, Languages and Computation, Narosa Publishing House.

3. Martin J. C., Introduction to Languages and the Theory of Computation, 2e, Tata McGraw-Hill .

4. Lewis H. R. and Papadimitriou C. H., Elements of the theory of computation, Pearson Education Asia

5. Daniel I A Cohen, Introduction to computer Theory, Wiley II Edition

Outcomes:

• Consider one of the computing paradigms as membership, and then design models for various classes of sets.

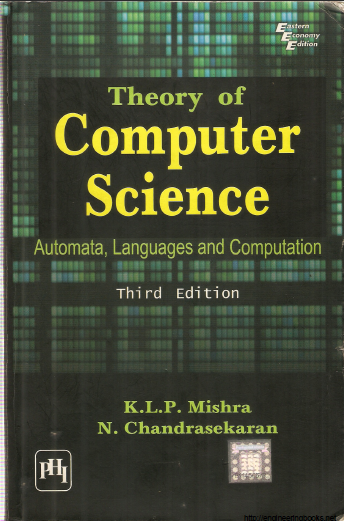
• Able to apply the proposed models to find the performance evaluation of scientific computations.

• Understanding the models to design compiler various phases.

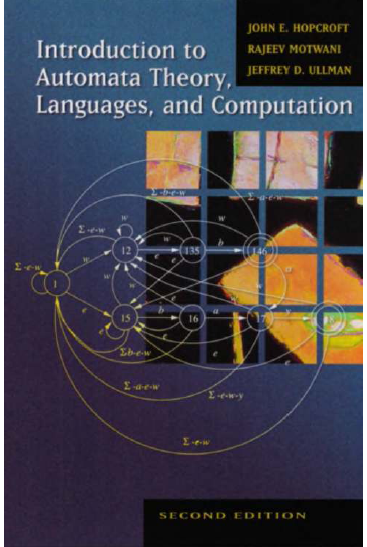
• Design various machines/models for simple calculations.

Suggested Books

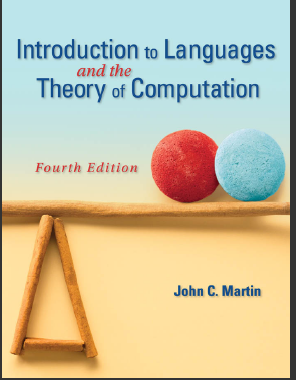
1. Mishra K. L. P., Chandrasekaran N., Theory of Computer Science, Automata, Languages and Computation.



2. Hofcroft J.E., Ullman J.D., Introduction to Automata Theory, Languages and Computation, Narosa Publishing House.



3. Martin J. C., Introduction to Languages and the Theory of Computation, 2e, Tata McGraw-Hill .



**Definition of Automaton**

* An automaton is defined as a system where energy, material , information are transformed , transmitted and used for performing some functions without direct participation of man.
* Examples are automatic machine tools, automatic packing machines, and automatic photo printing machines and many more.

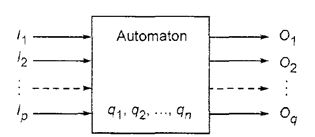
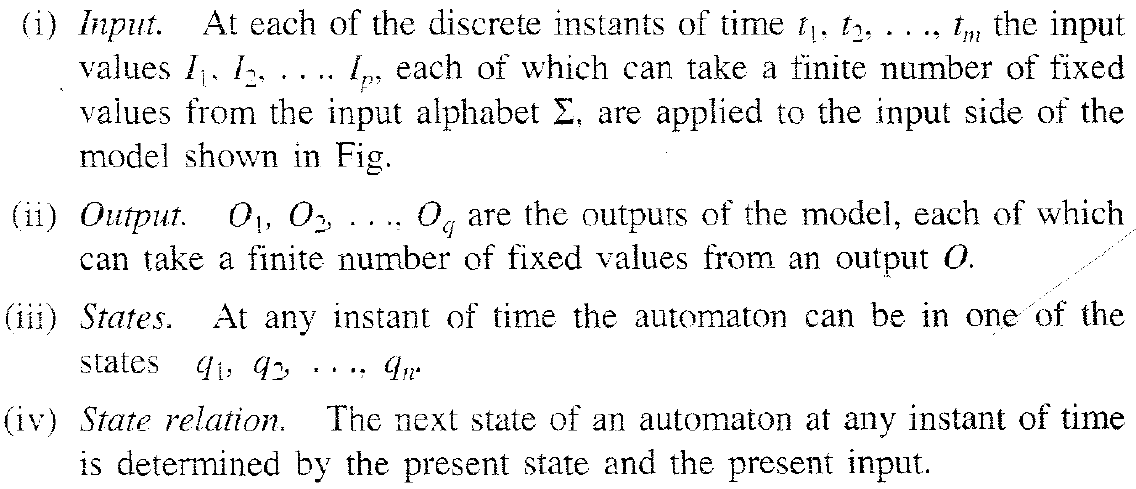


Fig:-Model of discrete automaton

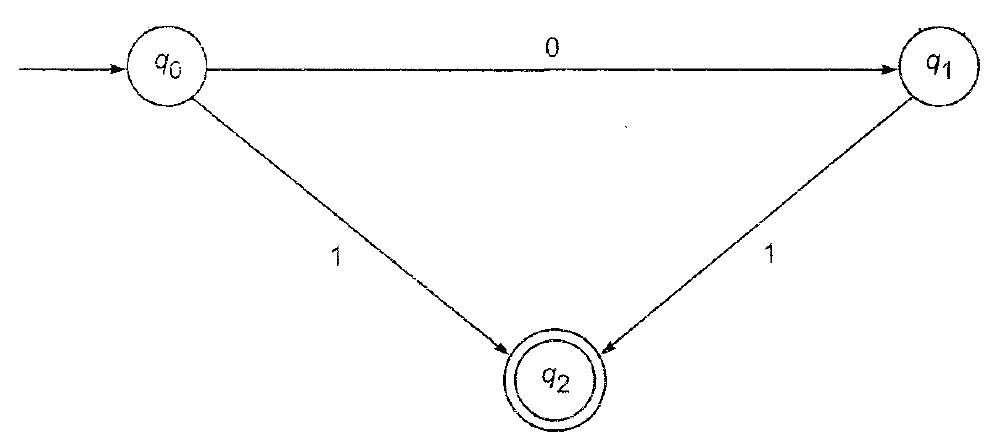
* The characteristics of automaton are as follows.



Q= set of states, q1, q2, …..states

(q1, a) 🡪 q2

* At any instant of time the **automaton is in some state.** On **‘reading’** an input symbol, the automaton **moves to a next state** which is given by the **“state relation”**.



|  |
| --- |
| **Input:- {0,1}** |
| **States:- {q0, q1, q2}** |
| **State relation:- (q0, 0) q1, (q1,1) q2, (q0, 1) q2,** |
| **Output:- as a string 01,1** |
| **(q0, 00) ?? check the steps.** |

* **A finite automaton (FA) or Finite state machine is a device that recognizes a language (set of strings).**

1. It has finite memory and an input tape;
2. Each input symbol that is read causes the machine to update its state based on its current state and the symbol read.
3. The machine accepts the input if it is in an **accept state** at the end of the string; otherwise, the **input is rejected**.

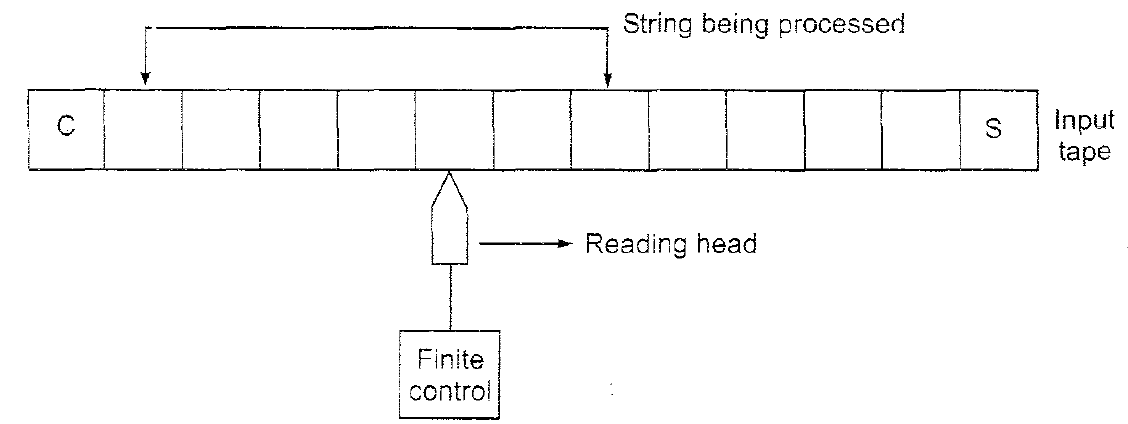


Fig:-Block diagram of Finite Automaton

It has following components

(i) Input tape:-

* Divided into squares, each square containing a single symbol from the input alphabet Ʃ.
* Tap contains left endmarker ¢ and right endmarkers $.
* Absence of endmarkers indicates that the tape is of infinite length.
* The left-to-right sequence of symbols between the two endmarkers is the input string to be processed.

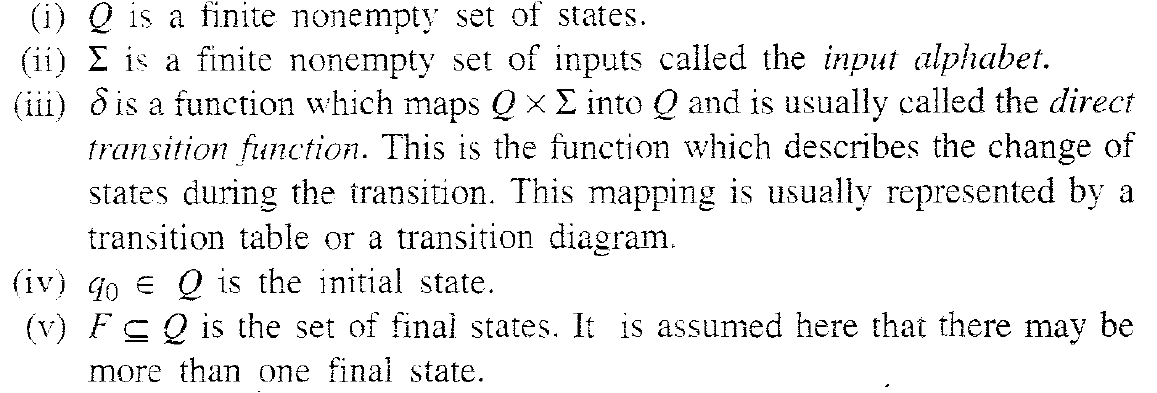
(ii) Reading head (R-head):-

* The head examines only one symbol (or one square) at a time and can move one square either to the RIGHT.
* Here movement is restricted only to the right side.

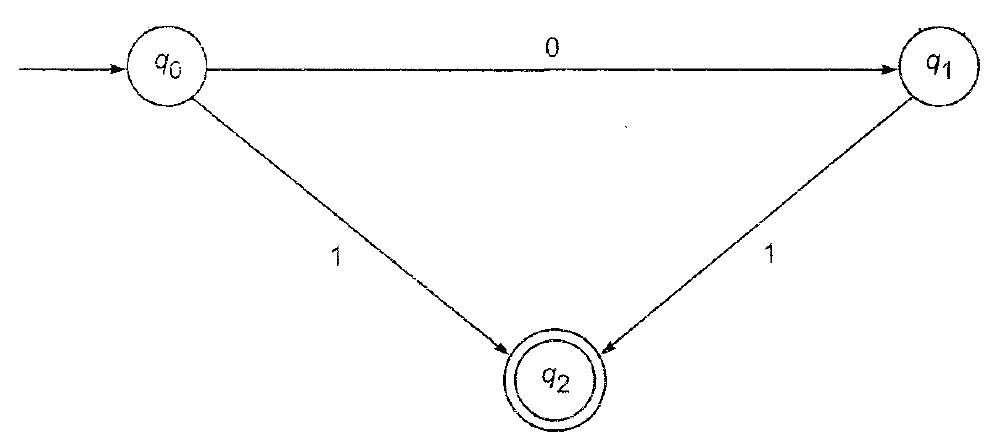
(iii) Finite control:-

* The input to the finite control will be symbol under the R-head, say 0, and present state of machine, say q0.
* By applying any input on present state, it moves to next symbol and state gets changed.
* Like δ(q0, 0) 🡪 q1
* **Formal Definition of a Finite Automaton or Finite State Machine**

**A finite automaton can be represented by a 5-tuple (Q, Ʃ, δ, q0, F), where**



Example:-



|  |
| --- |
| 5-tuple (Q, Ʃ, δ, q0, F) are shown as follows. |
| Q States:- {q0, q1, q2} |
| Ʃ Input:- {0,1} |
| δ State relation:- (q0, 0) q1 |
| Initial state q0 |
| F final state:- q2 |

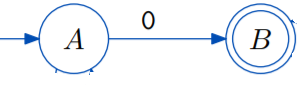
* **A finite automaton has a finite set of states with which it accepts or rejects strings.**
* **Operation of FA is as follows**

1. Set the machine to start state
2. If End-of-String, then ‘Halt’.
3. Read a symbol
4. Update state according to current state and symbol read.
5. Goto step 2

* Finite Automaton with Final state(s)

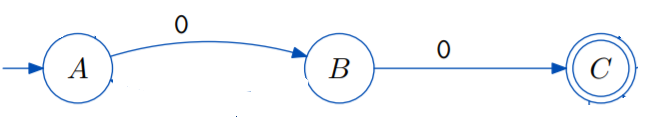
1. Final state is a state where FA finished reading the input string.
2. There are ‘accept’ states (double circled) and ‘reject’ states.
3. An FA ‘accepts’ input string if final state is accept state; otherwise it rejects.

Example:



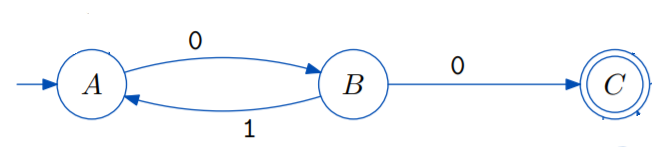
Language:- Final state accepts 0 as a string. This FA accepts 0 as a string. FA scans single length of 0.

Example:-



Sol:- 00 as a string.

Example:-



Sol:- 00, 0100, 010100,0101010100, 010101………00

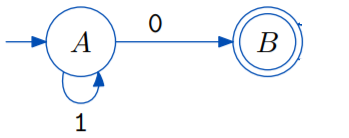
Language:-Final state accepts 00 as a string.

Final state accepts 0100 as a string.

Final state accepts 010100 as a string.

Can be written as **(01)\* 00** as a string.

Example:- LOOP

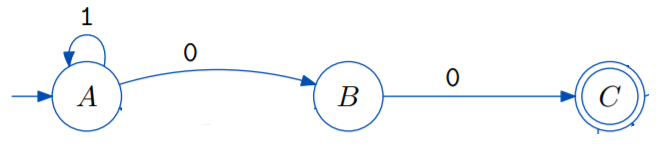


Language:- Final state accepts 0, 10, 110, 1110, 11111….0.

Means infinite length of 11111111…..followed by 0.

Can be written as **1\* 0**.

Example:-LOOP

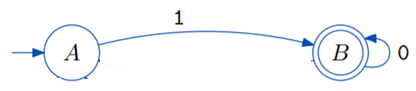


Language:- Final state accepts 00, 100, 1100, 11100, 1111…….00,

Means infinite length of 111111……followed by 00.

Can be written as **1\* 00**,

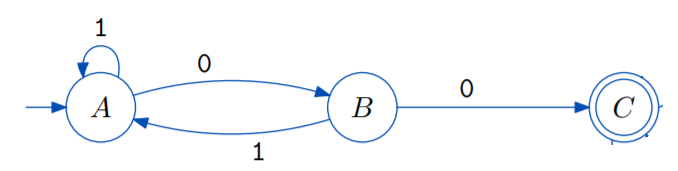
Example:-



Language:- FA accepts string 1, 10, 100, 10000…….0000, NULL,

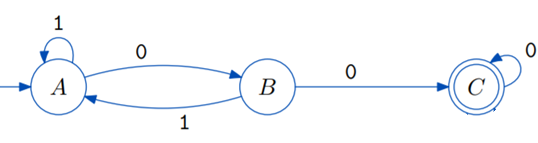
Can be written as **10\*,**

Question:-



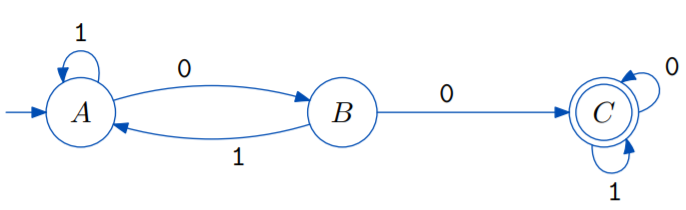
Language:- **(1\*(01)\*)\* 00**

Question:-

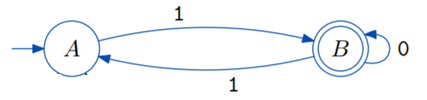


Language:-???

Question:-

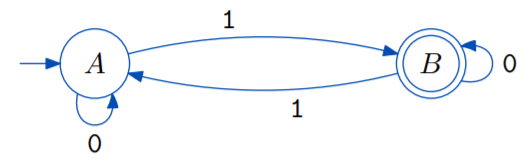


Question:-



Language:- ???

Question:-



Language:-???

Language:-Accepts all strings of 0’s and 1’s with odd numbers of 1’s.

**Transition diagrams:-**

* Each state is a node.
* For each state q Є Q and each symbol a Є Ʃ, let δ (q, a) = p, then the transition diagram has an arc from q to p, labeled a.
* There is an arrow to the start state q0.

Example:-

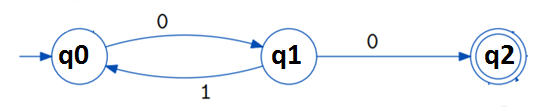


**Transition table:-**

* The transition function (state relation) of a finite accepter can be represented by transition table.
* The row corresponds to the states and the columns to the inputs in the table.
* The entry for the row corresponding to state q and the column corresponding to the input a is the state relation δ (q, a)

|  |  |  |
| --- | --- | --- |
| State | Input (0) | Input (1) |
| q0 | - | q1 |
|  | q1 | - |

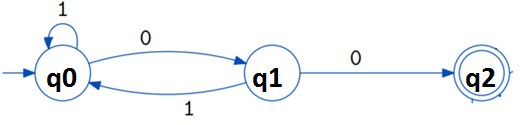
Example:-



Transition table

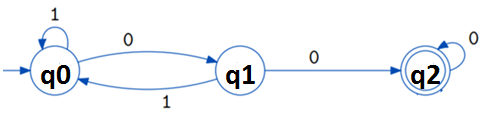
|  |  |  |
| --- | --- | --- |
| State | Input (0) | Input (1) |
| q0 | q1 | - |
| q1 | q2 | q0 |
|  | - | - |

Question:-



Transition table:-???

Question:-



Transition table:-???

**How string is processed by Finite automaton?**

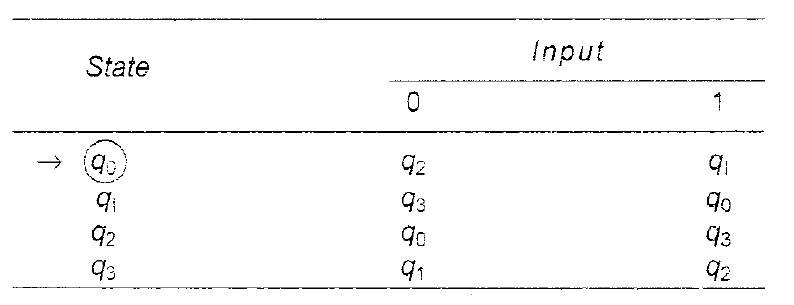
Example:- Consider the finite state machine whose transition function δis given by following transition table. Here,

Q = {q0, q1, q2, q3}

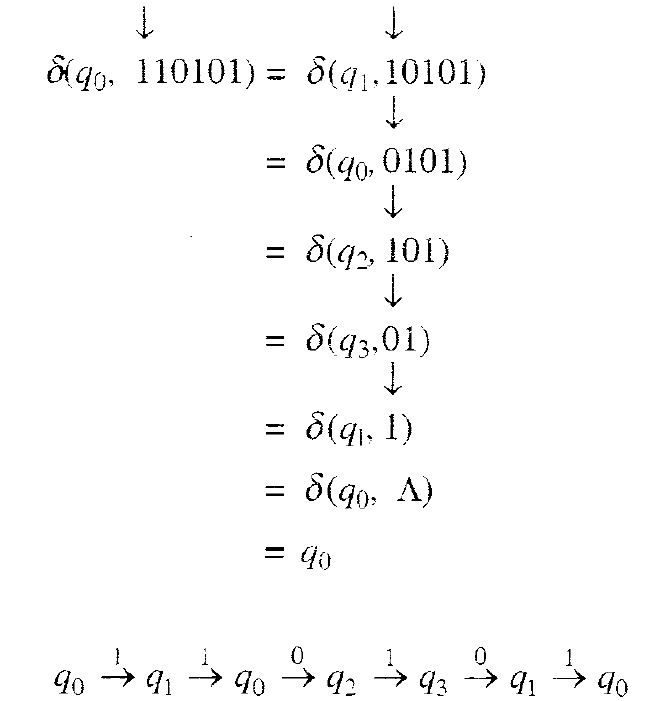
Ʃ= {0, 1},

F = {q0}

δ = (Transition function) in form of table is given as



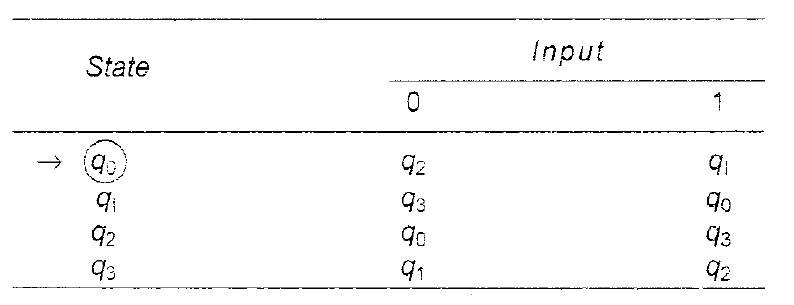
**See the entire sequence of states for the input string 110101.**



Q. Consider the finite state machine whose transition function δis given by following transition table. Here,

Q = {q0, q1, q2, q3}, Ʃ= {0,1}, F = {q0}

δ = (Transition function) in form of table is given as



**Q1. Draw the entire sequence of states for the following input string: 011101**

**δ (q0, 011101) = δ ( ). Will this string be accepted by Finite state machine?**

**Q2. Draw the entire sequence of states for the following input string: 01101**

**δ (q0, 01101) = δ ( ). Will this string be accepted by Finite state machine?**