CSC-257
Theory Of Computation
(BSc CSIT, TU)

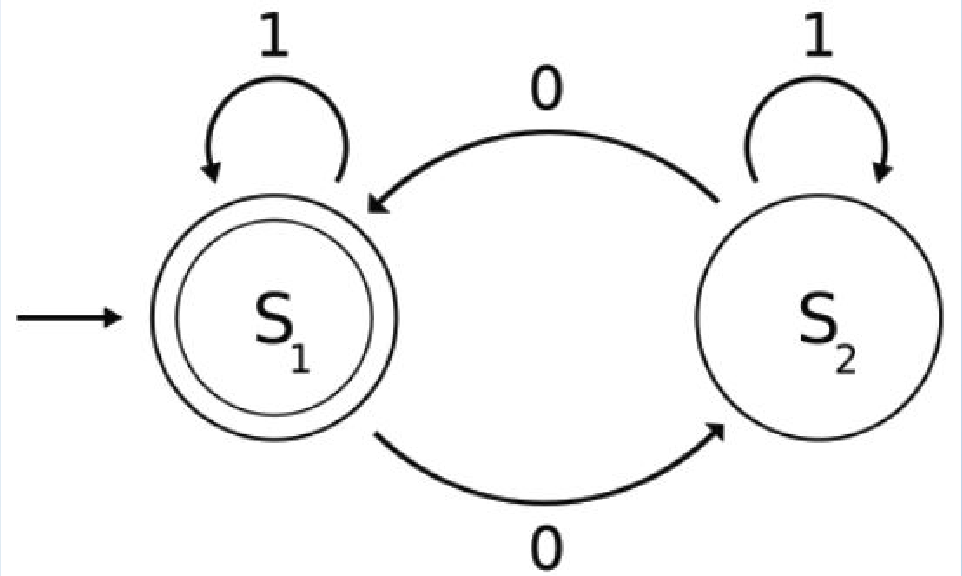
Ganesh Khatri
kh6ganesh@gmail.com

Automata Theory

- The study of the mathematical properties of abstract machine or automata is automata theory
- In theoretical computer science, automata theory is the study of abstract machines (or more appropriately, abstract 'mathematical' machines or systems, as they are described in mathematical terms) and the computational problems that can be solved using these machines.
- These abstract machines are called automata
- Automata come from the Greek word, which means "self-acting"

Automata Theory

- The figure above illustrates a finite state machine, which belongs to one well-known variety of automatons
- 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 (or jump) to another state, according to its transition function (which takes the current state and the recent symbol as its inputs)
- Alphabet = {0, 1}



Automata Theory

- Automata theory is also closely related to formal language theory
- An automaton is a finite representation of a formal language that may be an infinite set
- Automata are often classified by the class of formal languages they are able to recognize
- Automata play a major role in theory of computation, compiler design, parsing and formal verification

Finite-State Machine

- A finite-state machine (FSM) or finite-state automaton (plural: automata), or simply a state machine, is a mathematical model used to design computer programs and digital logic circuits
- It is conceived as an abstract machine that can be in one of a finite number of states
- The machine is in only one state at a time;
- the state it is in at any given time is called the current state
- It can change from one state to another when initiated by a triggering event or condition, this is called a transition.
- A particular FSM is defined by a list of the possible transition states from each current state, and the triggering condition for each transition.
- Finite-state machines can model a large number of problems, among which are electronic design automation, communication protocol design, parsing and other engineering applications.
- In biology and artificial intelligence research, state machines or hierarchies of state machines are sometimes used to describe neurological systems and in linguistics - to describe the grammars of natural languages

Finite Automata

- An automaton with a set of states, and its "control" moves from state to state in response to external "inputs" is called a finite automaton
- A finite automaton, FA, provides the simplest model of a computing device
- Finite automata are used for pattern matching in text editors, for compiler lexical analysis

Finite Automata : Formal Definition

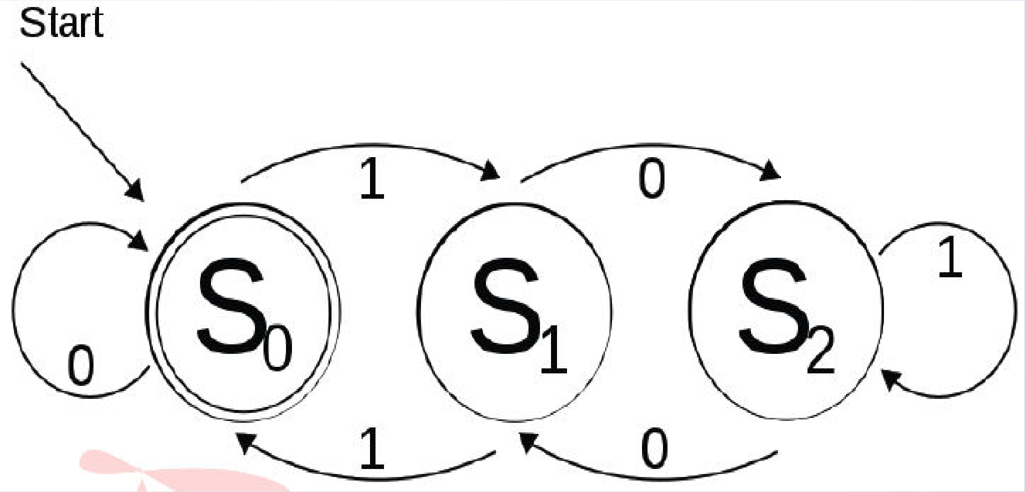
- An automaton is represented formally by
  - a 5-tuple $(Q, \Sigma, \delta, q0, F)$, where:
- Q is a finite set of states.
- Σ is a finite set of symbols, called the alphabet of the automaton
- δ is the transition function, that is, $\delta: Q \times \Sigma \rightarrow Q$
- $q0$ is the start state, that is, the state of the automaton before any input has been processed, where $q0 \in Q$
- F is a set of states of Q (i.e. $F \subseteq Q$) called accept states

Applications

- Finite automata are used in text processing, compilers, and hardware design
- Context-free grammar (CFGs) are used in programming languages and artificial intelligence. Originally, CFGs were used in the study of the human languages
- Cellular automata are used in the field of biology, the most common example being John Conway's Game of Life etc.

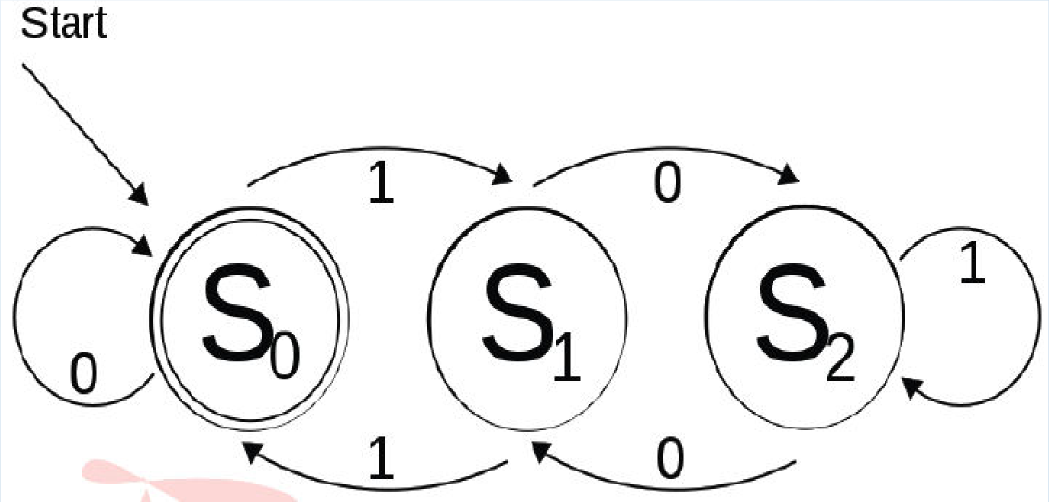
Deterministic finite automata(DFA)

- It is a finite state machine that accepts/rejects finite strings of symbols and only produces a unique computation (or run) of the automaton for each input string
- The figure at above illustrates a deterministic finite automaton
- there are three states: S0, S1, and S2
- automaton takes finite sequence of 0s and 1s as input
- For each state, there is a transition arrow leading out to a next state for both 0 and 1
- Upon reading a symbol, a DFA jumps deterministically from a state to another by following the transition arrow { 0, 1 }



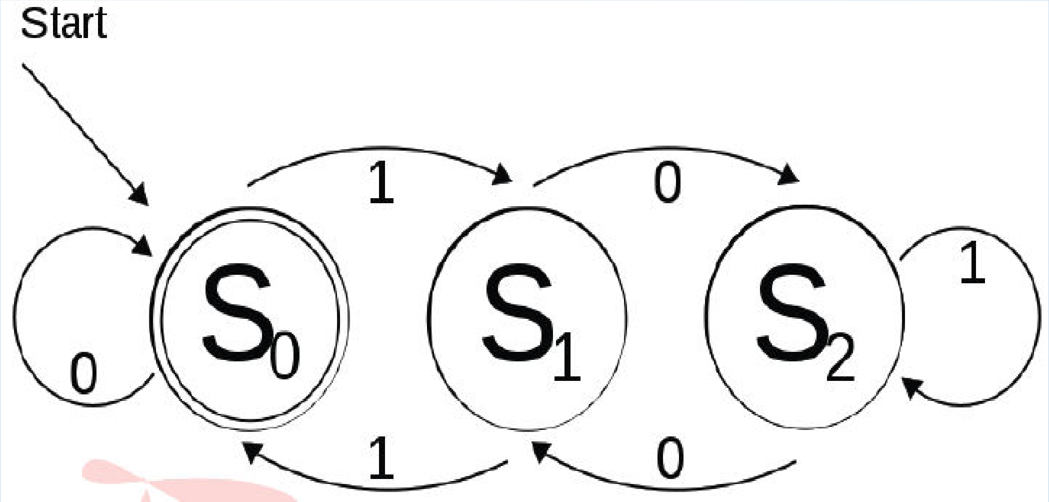
Deterministic finite automata(DFA)

- For example, if the automaton is currently in state S0 and current input symbol is 1 then it deterministically jumps to state S1.
- A DFA has a start state (denoted graphically by an arrow coming in from nowhere) where computations begin, and a set of accept states (denoted graphically by a double circle) which help define when a computation is successful
- A DFA is defined as an abstract mathematical concept, but due to the deterministic nature of a DFA, it is implementable in hardware and software for solving various specific problems



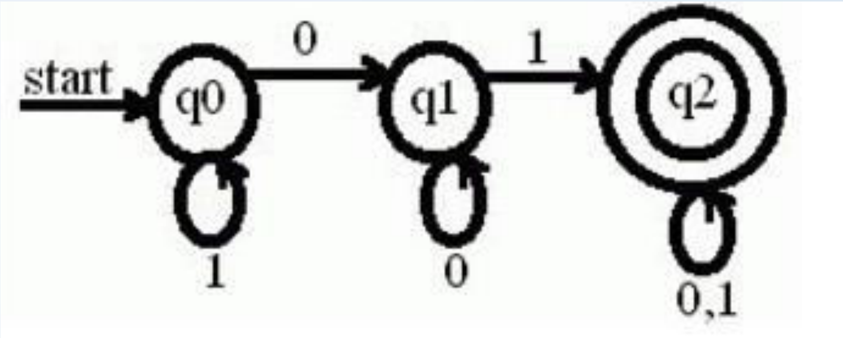
Deterministic finite automata(DFA) : Formal Definition

- deterministic finite automaton M is a 5-tuple, (Q, Σ, δ, q0, F), consisting of
- a finite set of states (Q)
- a finite set of input symbols called the alphabet (Σ)
- a transition function ($\delta : Q \times \Sigma \rightarrow Q$)
- a start state ($q0 \in Q$)
- a set of accept states ($F \subseteq Q$)



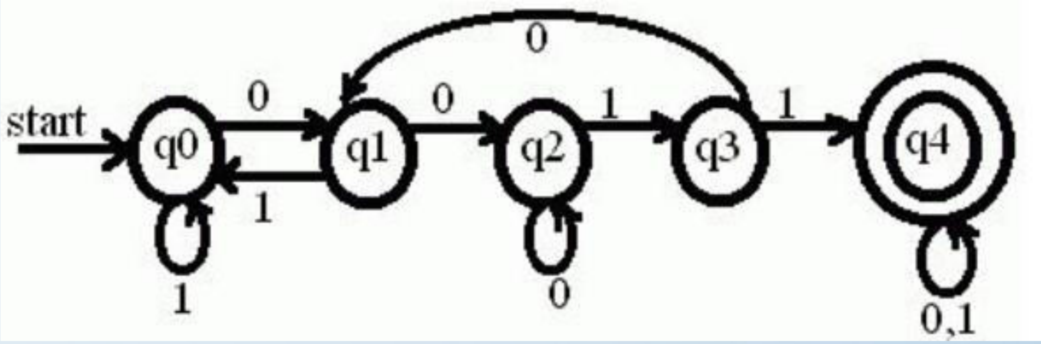
Exercise

- 1. Construct a DFA to accept a string containing a zero followed by a one. Or which contains 01.



- eg 1001

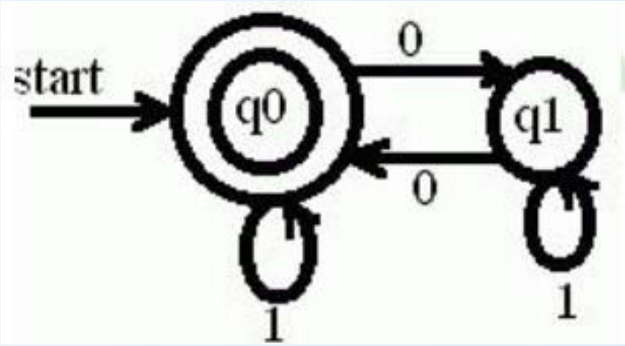
- 2. Construct a DFA to accept a string containing two consecutive zeroes followed by two consecutive ones or which contains 0011.



- Eg. 101000010011

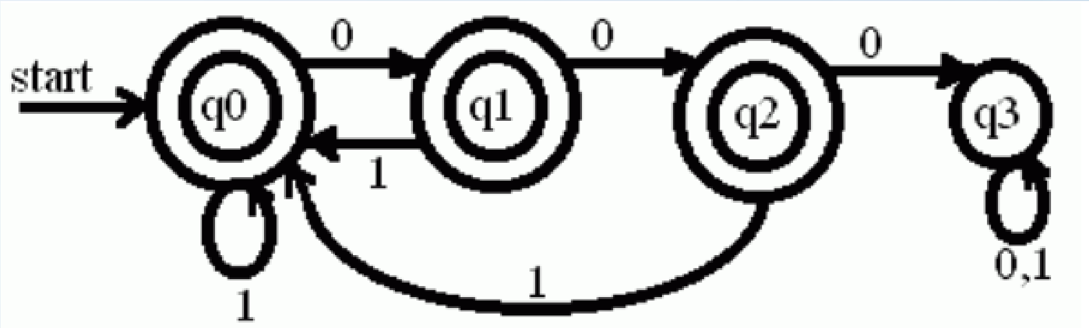
Exercise

- 3. Construct a DFA to accept a string containing even number of 0’s and any number of 1’s.



- e.g. 1, 00, 001, 100, 010, 0011, 00111, 01010101

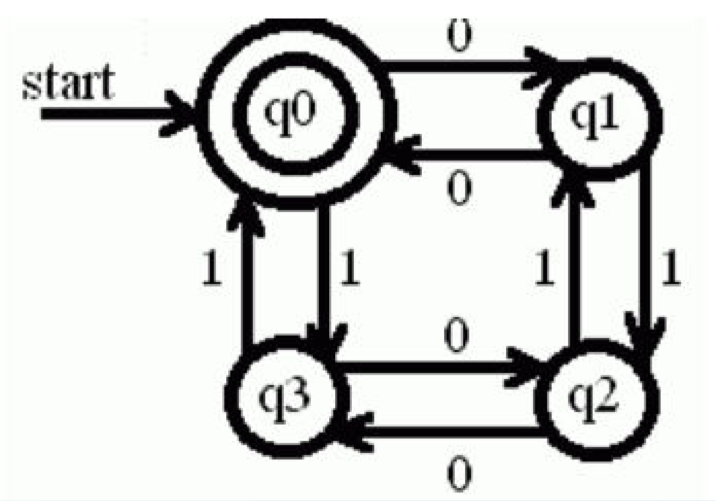
- 4. Construct a DFA to accept all strings which do not contain three consecutive zeroes.



- 001010, 001001110

Exercise

- 5. Construct a DFA to accept all strings containing even number of zeros and even number of ones.
- 1010, 0101, 1100, 0011, 1001, 0110, 11110000 etc



- 6. Construct A DFA to accept strings containing equal number of 0's and 1's