Finite Automata

Beulah A.
AP/CSE

Introduction

- FA recognizes regular languages only. It was developed by "Scott Robin" in 1950 as a model of a computer with limited memory.
- Input → string, reads from an input tape.
- Output → no output. Instead an indication of whether the input is acceptable (or) not.
- Hence used for decision making problems

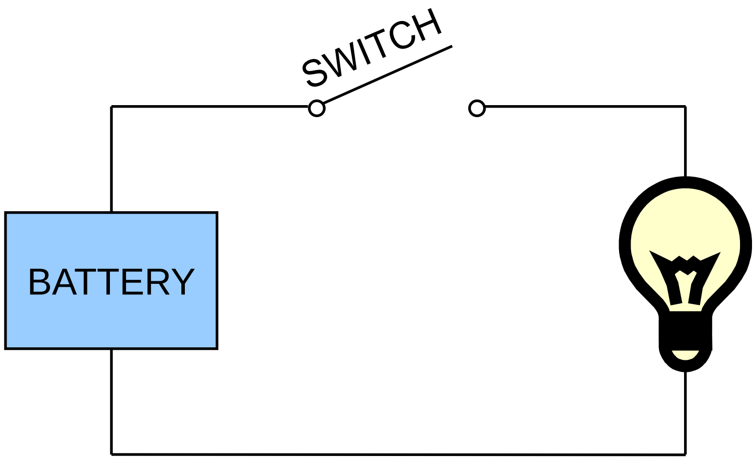
Applications of FA

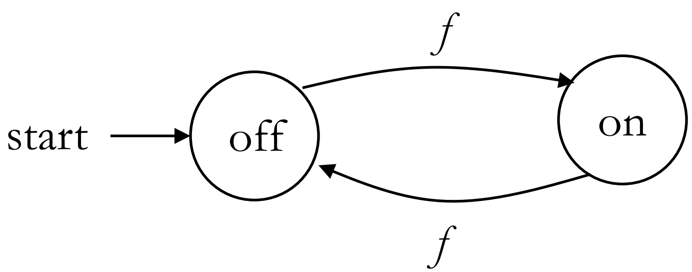
- It is an useful tool in the design of Lexical analyzer - a part of compiler that groups characters into tokens, indivisible units such as variable name and keyword.
- Text editor
- Pattern matching
- File searching program
- Text processing (searching an occurrence of one string in a file)

Limitations

- It can recognise only simple languages (regular)
- FA can be designed only for decision making problems.

Example





input: switch

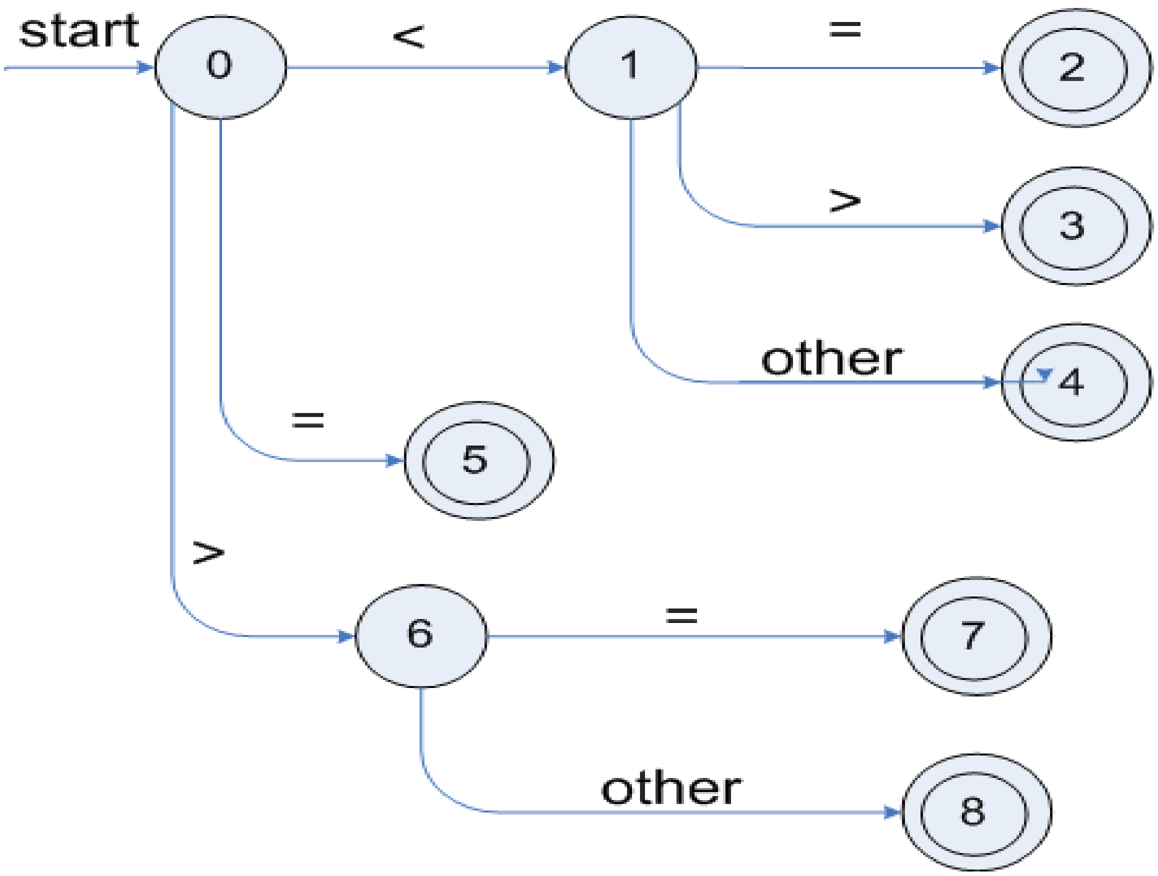
output: light bulb

actions: f for "flip switch"

states: on, off

bulb is on if and only if there was an odd number of flips

LA–Recognising relational operators



Context Free Languages

- It allows richer syntax than regular languages.
- Context free grammars can be recognised by computing devices like *pushdown automata*.
- Pushdown automata is a finite automata with an auxiliary memory in the form of a stack.
- It is immensely used in the design of parsers - another key portion of a compiler

Turing Machines

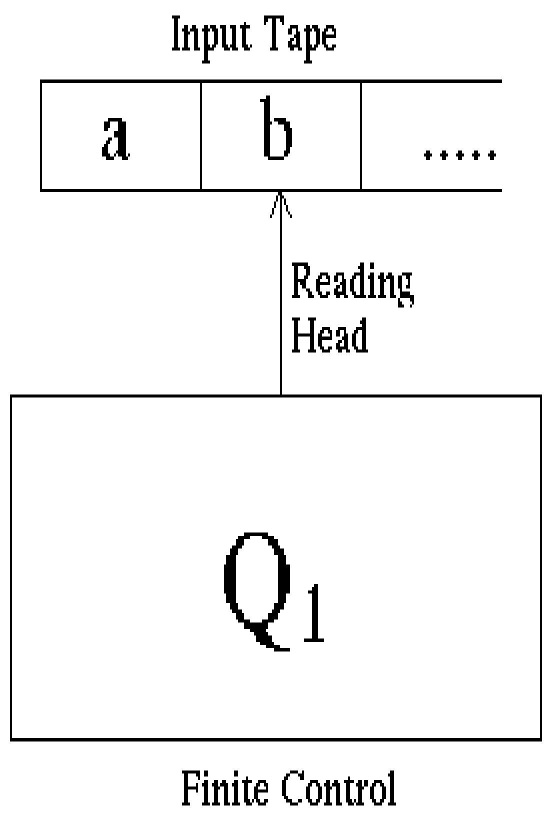
- It was invented by British Mathematician *“Alan Turing”*
- It is a most powerful abstract model.
- It has infinite amount of tape memory accessible in both directions, that is left (or) right.
- It can recognize recursively enumerable languages.
- It simulates digital computer in terms of power.
- If any function is not solvable by Turing machine, it cannot be computed by digital computer.

Finite state Systems

- The FA is a mathematical model of a system, with discrete inputs and outputs and a finite number of memory called as states and a set of transitions from state to state that occurs o input symbols from alphabet Σ
- The FA is classified as:
  - Deterministic Finite Automata (DFA)
  - Non Deterministic Finite Automata (NFA)

Deterministic Finite Automata

- DFA is a language recogniser that has :
- An input file containing an input string.
- A finite control - a device that can be in a finite number of states.
- A reader - a sequential reading device
- Program



How DFA works ?

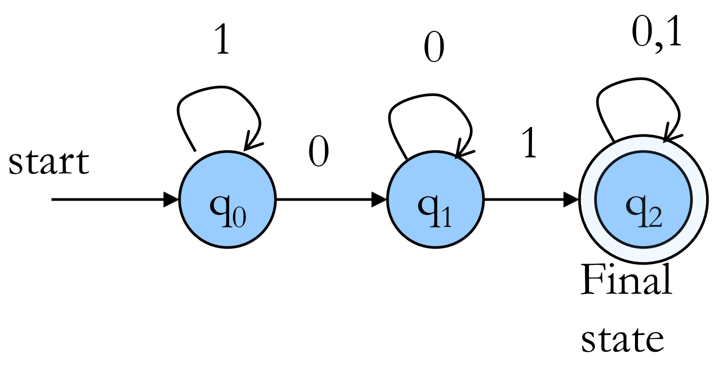
- Initialization :
  - Reader start reading from the leftmost symbol. Finite control is in start state.
- Single step :
  - Reader reads current symbol then, reader moves to the next symbol to the right. And Control enters a new state
- No current symbol :
  - All symbols have been read then, if control is in final state, the input string is accepted. Otherwise, the input string is not accepted.

DFA Specification

- A Deterministic Finite Automata (DFA) is a 5-tuple $(\mathcal{Q}, \Sigma, S, F, \delta)$ where
  - $\mathcal{Q}$ is a finite set of states
  - Σ is an alphabet
  - S: $q_0 \in \mathcal{Q}$ is the initial state
  - $F \subseteq \mathcal{Q}$ is a set of accepting states (or final states)
  - δ: $\mathcal{Q} \times \Sigma \rightarrow \mathcal{Q}$ is a transition function

Example

DFA for strings containing 01



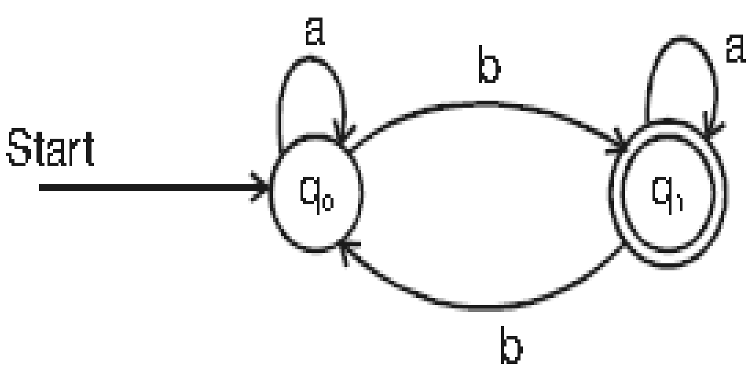
- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0,1\}$
- $S = q_0$
- $F = \{q_2\}$
- Transition table

δ	0	1
→ q_0	q_1	q_0
q_1	q_1	q_2
$*q_2$	q_2	q_2

What makes this DFA deterministic?

Transition diagram of DFA

- It is a directed graph whose vertices corresponds to states of DFA. The edges are the transitions from one state to another
- In the transition diagram, start state s is represented by → and the final states are represented by * or double circle.



Properties of Transition Function (δ)

1. $\delta (q, \varepsilon) = q$

This means the state of the system can be changed only by an input symbol else remains in original state.

2. For all strings *w and input symbol a*

$$\delta(q, aw) = \delta (\delta (q, a), w)$$

similarly $\delta(q, wa) = \delta (\delta (q,w), a)$

3. The transition function δ can be extended to $\bar{\delta}$(or) $\hat{\delta}$ that operates on states and strings (as opposed to states and symbols)

   Basis : $\bar{\delta} (q, \varepsilon) = q$

   Induction : $\bar{\delta} (q, xa) = \delta(\bar{\delta} (q, x), a)$

Language of a DFA

- A string x is said to be accepted by DFA $M = (Q, \Sigma, S, F, \delta)$, if $\delta(q_0, x) = p$, for some p in F.

Method :

- A finite automata accepts a string $w = a_1 a_2 \ldots a_n$ if there is a path in the transition diagram which begins at a start state ends at an accepting state with the sequence of labels $a_1 a_2 \ldots a_n$
- The Language accepted by finite automata (A) is

  $L(A) = \{w : \bar{\delta}(q_0, w) \in F\}$ where F is a final state.
- The language accepted by finite automata's are called "regular language."

Example

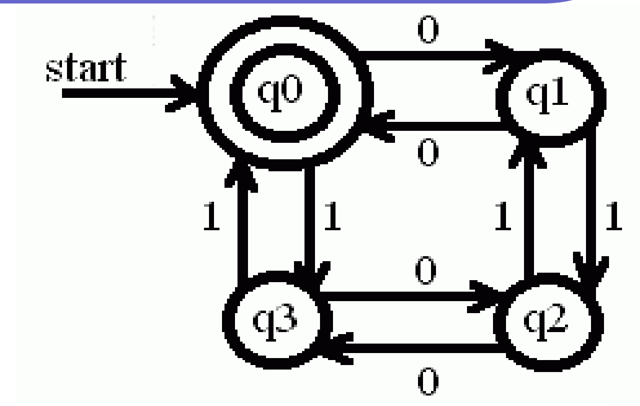
- The DFA for the above transition is represented as: M = $(Q, \Sigma, S, F, \delta)$ where

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0,1\}$

$S = q_0 \rightarrow$ Start State

$F = \{q_0\}$



$\delta \rightarrow$

States	Inputs	
	0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Example

- Suppose 110101 is input to M, check the validity of the input.
- Finite automata is in start state and reads from left most.

$\delta(q_0, 1) = q_1$

$\delta(q_1, 1) = q_0$ (Reader reads next symbols)

$\delta(q_0, 0) = q_2$ (Reader moves one position right)

$\delta(q_2, 1) = q_3$

$\delta(q_3, 0) = q_1$

$\delta(q_1, 1) = q_0$

since q_0 is a final state, the given string is accepted.

Summary

- Introduction to Finite Automata
- Definition of DFA
- Transition diagram, transition function and properties of transition function

Test Your Knowledge

- Design a DFA that accepts input string 0's and 1's that ends with 11
- Design a DFA over {0,1} to accept strings with 3 consecutive 0's

Reference

- Hopcroft J.E., Motwani R. and Ullman J.D, "Introduction to Automata Theory, Languages and Computations", Second Edition, Pearson Education, 2008