Finite Automata

Finite automata is an abstract computing device. It is a mathematical model of a system with discrete inputs, outputs, states and a set of transitions from state to state that occurs on input symbols from the alphabet Σ.

- Finite automata are used to recognize patterns.
- It takes the string of symbol as input and changes its state accordingly. When the desired symbol is found, then the transition occurs.
- At the time of transition, the automata can either move to the next state or stay in the same state.
- Finite automata have two states, **Accept state** or **Reject state**. When the input string is processed successfully, and the automata reached its final state, then it will accept.

Finite Automata Representation

The finite automata can be represented in three ways, as given below −

- Graphical (Transition diagram)
- Tabular (Transition table)
- Mathematical (Transition function)

Formal definition of Finite Automata

Finite automata is defined as a 5-tuples

M=(Q, Σ, δ,q0,F)

Where,

- Q: Finite set called states.
- Σ: Finite set called alphabets.
- δ: $Q \times \Sigma \rightarrow Q$ is the transition function.
- $q0 \in Q$ is the start or initial state.
- F: Final or accept state.

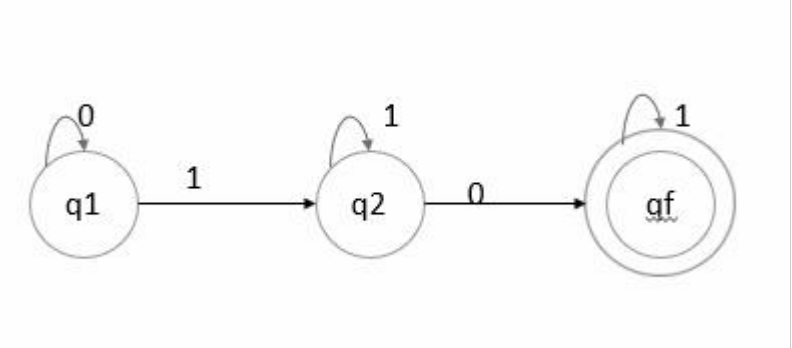
Finite Automata can be represented as follows −

- Transition diagram
- Transition table
- Transition function

Transition Diagram

It is a directed graph associated with the vertices of the graph corresponding to the state of finite automata.

An example of transition diagram is given below −



Here,

- {0,1}: Inputs
- q1: Initial state
- q2: Intermediate state
- qf: Final state

Transition table

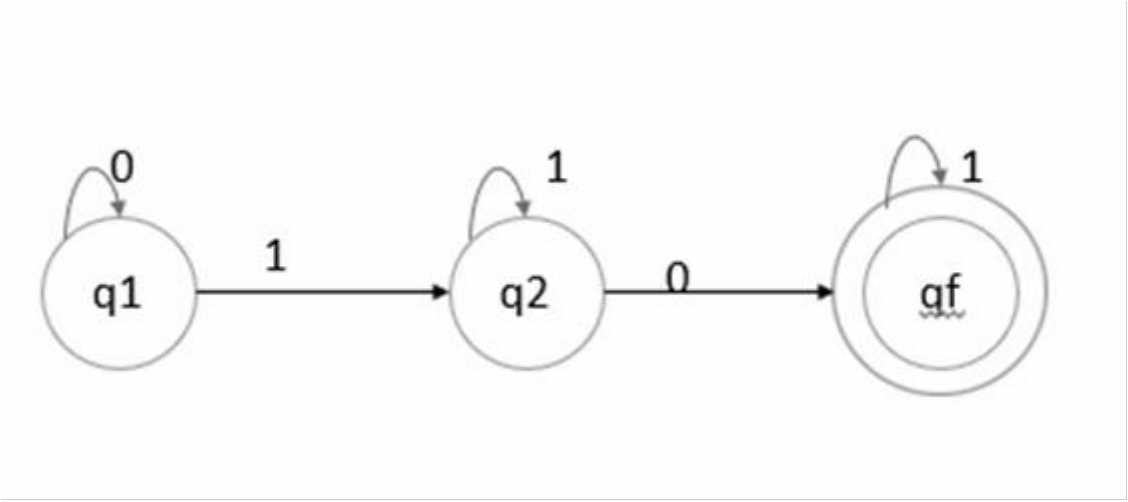
It is basically a tabular representation of the transition function that takes two arguments (a state & a symbol) and returns a value (the 'next state').

$\delta : Q \times \Sigma \rightarrow Q$

In transition table, the following factors are considered −

- Rows correspond to state.
- Column corresponds to the input symbol.
- Entries correspond to the next state.
- The start state is marked with ->.
- The accept state marked with *.

An example of transition table is as follows −



The transition table is as follows −

State/input symbol	0	1
->q1	q1	q2
q2	qf	q2
qf	-	qf

Transition function

The transition function is denoted by δ. The two parameters mentioned below are the passes to this transition function.

- Current state
- Input symbol

The transition function returns a state which can be called as the next state.

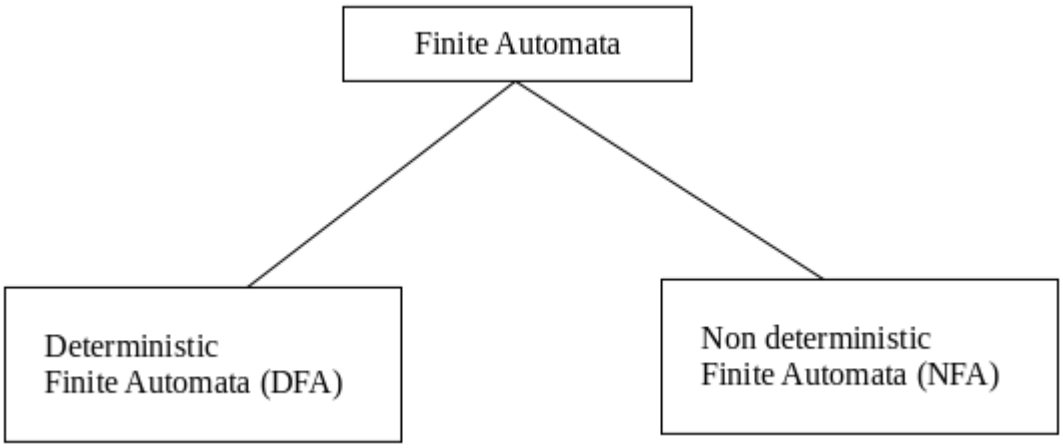
δ (current_state, current_input_symbol) = next_state

For example, δ(q0,a)=q1

Types of Automata:

There are two types of finite automata:

1. DFA(deterministic finite automata)
2. NFA(non-deterministic finite automata)



1. DFA

DFA refers to deterministic finite automata. Deterministic refers to the uniqueness of the computation. In the DFA, the machine goes to one state only for a particular input character. DFA does not accept the null move.

2. NFA

NFA stands for non-deterministic finite automata. It is used to transmit any number of states for a particular input. It can accept the null move.

Understanding Automata and Automaton: A Deep Dive

Automata is a complex and evolving field that intersects with various domains of computer science and artificial intelligence. This section delves into the nuances of automata and automaton, exploring their differences, applications, and implications in modern technology.

Automata vs Automaton

While the terms 'automata' and 'automaton' are often used interchangeably, they have distinct meanings. An automaton refers to a single computational entity, whereas automata is the plural form, encompassing multiple such entities. Understanding this distinction is crucial for grasping the broader concepts in computational theory.

Key Characteristics of Automata

Determinism vs Non-Determinism: Automata can be classified into deterministic and non-deterministic types. Deterministic automata have a single possible action for each state and input, while non-deterministic automata can have multiple possible actions. **Some important points about DFA and NFA:**

- Every DFA is NFA, but NFA is not DFA.
- There can be multiple final states in both NFA and DFA.

- DFA is used in Lexical Analysis in Compiler.
- NFA is more of a theoretical concept.

- **Finite vs Infinite**: Finite automata have a limited number of states, making them suitable for simpler tasks, whereas infinite automata can handle more complex scenarios.

- **State Transitions**: The behavior of an automaton is defined by its state transitions, which dictate how it moves from one state to another based on input.

Applications of Automata

Automata play a crucial role in various applications, including:

- **Compiler Design**: Automata are used in the lexical analysis phase of compilers to recognize tokens in source code.

- **Network Protocols**: They help in modeling and analyzing the behavior of network protocols, ensuring reliable communication.

- **Artificial Intelligence**: In AI, automata can model decision-making processes and learning algorithms, contributing to the development of intelligent systems.

Visual Representation

To better understand automata, consider the following state transition diagram:
State A --(input 1)--> State B
State B --(input 0)--> State A
This diagram illustrates a simple finite automaton where the states transition based on specific inputs. Such visual aids can significantly enhance comprehension of complex concepts.

Conclusion

In summary, the study of automata and automaton is foundational to understanding modern computational systems. Their applications span various fields, making them integral to advancements in technology and artificial intelligence.