About Automata

Automata Theory deals with designing abstract computing devices that follow an automatic sequence of operations.

The term "Automata" is derived from the Greek word "αὐτόματα" which means "self-acting". An automaton (Automata in plural) is an abstract self-contain computing device which follows a preplanned automatic sequence of operations.

An automaton with a finite number of states is called a **Finite Automaton** (FA) or **Finite State Machine** (FSM).

Formal definition of a Finite Automaton

An automaton can be represented by a 5-tuple (Q, ∑, δ, q0, F).

Where each symbol represents following state-

1. **Q** is a finite set of states.
2. **∑** is a finite set of symbols, called the alphabet of the automaton.
3. **δ** is the transition function.
4. **q0** is the initial state from where any input is processed ($q0 \in Q$).
5. **F** is a set of final state/states of Q ($F \subseteq Q$).

Finite Automaton

Deterministic Finite Automaton (DFA)

Non-deterministic Finite Automaton (NDFA / NFA)

Deterministic Finite Automaton (DFA)-

In DFA, for each input symbol, one can determine the state to which the machine will move. Hence, it is called Deterministic Automaton. As it has a finite number of states, the machine is called Deterministic Finite Machine or Deterministic Finite Automaton.

A DFA can be represented by a 5-tuple (Q, ∑, δ, q0, F).

Where Q, ∑, δ, q0, F represents following state-

1. **Q** is a finite set of states.
2. **∑** is a finite set of symbols called the alphabet.
3. **δ** is the transition function where $\delta: Q \times \sum \rightarrow Q$
4. **q_0** is the initial state from where any input is processed ($q_0 \in Q$).
5. **F** is a set of final state/states of Q ($F \subseteq Q$).

Graphical Representation of a DFA-

A DFA is represented by digraphs called **state diagram**.

- The vertices represent the states.
- The arcs labelled with an input alphabet show the transitions.
- The initial state is denoted by an empty single incoming arc.

The final state is indicated by double circles.

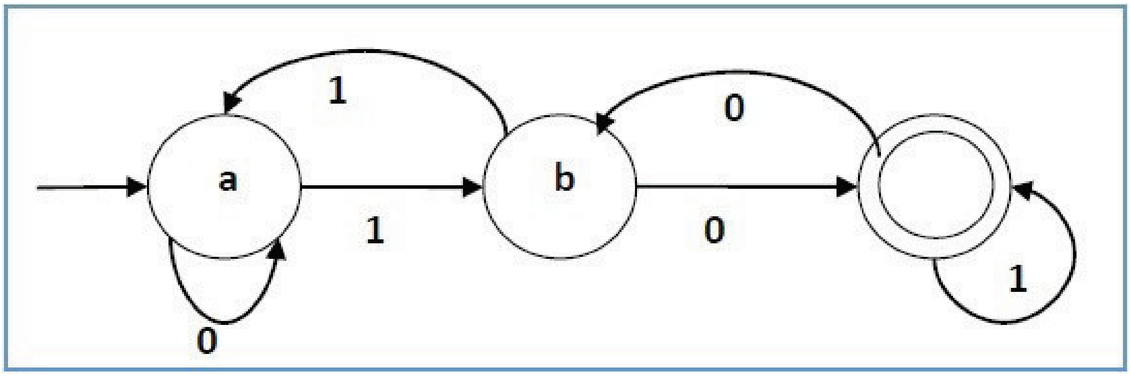
Example-

Let a deterministic finite automaton be →

- **Q** = {a, b, c},
- **∑** = {0, 1},
- **q_0** = {a},
- **F** = {c}, and Transition function **δ** as shown by the following table –

Present State	**Next State for Input 0**	**Next State for Input 1**
a	**a**	**b**
b	**c**	**a**
c	**b**	**c**

Its graphical representation would be as follows –



In NDFA, for a particular input symbol, the machine can move to any combination of the states in the machine. In other words, the exact state to which the machine moves cannot be determined. Hence, it is called **Non-deterministic Automaton**. As it has finite number of states, the machine is called **Non-deterministic Finite Machine** or **Non-deterministic Finite Automaton**.

Formal Definition of an NDFA-

An NDFA can be represented by a 5-tuple (Q, ∑, δ, q_0, F) where −

- **Q** is a finite set of states.
- **∑** is a finite set of symbols called the alphabets.
- **δ** is the transition function where δ: Q × ∑ → 2^Q

  (Here the power set of Q (2^Q) has been taken because in case of NDFA, from a state, transition can occur to any combination of Q states)

- **q_0** is the initial state from where any input is processed ($q_0 \in Q$).
- **F** is a set of final state/states of Q ($F \subseteq Q$).

Graphical Representation of an NDFA: (same as DFA)

An NDFA is represented by digraphs called state diagram.

- The vertices represent the states.
- The arcs labeled with an input alphabet show the transitions.
- The initial state is denoted by an empty single incoming arc.
- The final state is indicated by double circles.

Example-

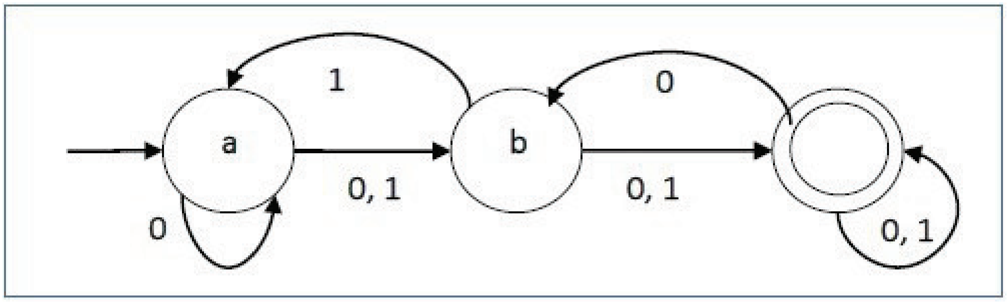
Let a non-deterministic finite automaton be →

- **Q** = {a, b, c}
- **∑** = {0, 1}
- **q_0** = {a}
- **F** = {c}

The transition function **δ** as shown below –

Present State	**Next State for Input 0**	**Next State for Input 1**
a	a, b	b
b	c	a, c
c	b, c	c

Its graphical representation would be as follows –



The following table lists the differences between DFA and NDFA.

DFA	NDFA
The transition from a state is to a single particular next state for each input symbol. Hence it is called deterministic.	The transition from a state can be to multiple next states for each input symbol. Hence it is called non-deterministic.
Empty string transitions are not seen in DFA.	NDFA permits empty string transitions.
Backtracking is allowed in DFA	In NDFA, backtracking is not always possible.
Requires more space.	Requires less space.
A string is accepted by a DFA, if it transits to a final state.	A string is accepted by a NDFA, if at least one of all possible transitions ends in a final state.

Source- Tutorialspoint.com