**Automata**

Introduction

Automata - “Self-acting”, abstract self-propelled computing device which follows a predetermined sequence of operations automatically.

Automaton – Plural

**Four Major Families of Automaton**

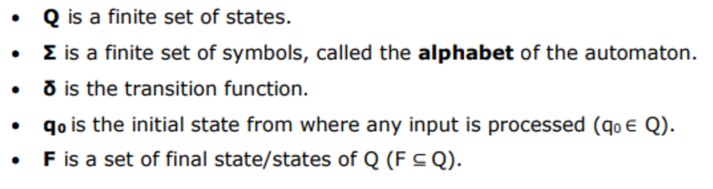
1. Finite State Machine - an automaton with a finite number of states
2. Pushdown Automata
3. Linear Bounded Automata
4. Turing Machine - a mathematical model of computation describing an abstract machine that manipulates symbols on a strip of tape according to a table of rules. Despite the model's simplicity, it is capable of implementing any computer algorithm.

Finite Automaton

Finite Automaton (FA) / Finite State Machine (FSM) - an automaton with a finite number of states

**Formal definition of a Finite Automaton**

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



**RELATED TERMINOLOGIES**

**Alphabet** - An alphabet is any finite set of symbols.

Example: Σ = {a,b,c,d} is an alphabet set where ‘a’,’b’,’c’, and ‘d’ are symbols.

**String** - A string is a finite sequence of symbols taken from Σ.

Example: ‘cabcad’ is a valid string on the alphabet set Σ = {a,b,c,d}

**Length of a String** - it is the number of symbols present in a string.

( Denoted by |S| )

**Kleene Star** - Σ\*, is a unary operator on a set of symbols or strings, Σ, that gives the infinite set of all possible strings of all possible lengths over Σ including **λ.**

Representation − ∑\* = ∑0 ∪ ∑1 ∪ ∑2 ∪……. where ∑p is the set of all possible strings of length p.

Example − If ∑ = {a, b}, ∑\* = {λ, a, b, aa, ab, ba, bb,………..}

**Kleene Closure/Plus**- The set ∑+ is the infinite set of all possible strings of all possible lengths over ∑ excluding λ.

Representation − ∑+ = ∑1 ∪ ∑2 ∪ ∑3 ∪…….

∑+ = ∑\* − { λ }

Example − If ∑ = { a, b } , ∑+ = { a, b, aa, ab, ba, bb,………..}

**Language** - is a subset of ∑\* for some alphabet ∑. It can be finite or infinite.

If the language takes all possible strings of length 2 over ∑ = {a, b}, then L = { ab, aa, ba, bb }

Deterministic and Nondeterministic Finite Automaton

Finite Automaton can be classified into **two types**:

* Deterministic Finite Automaton (DFA)
* Non-deterministic Finite Automaton (NDFA/ NFA)

**Deterministic Finite Automaton (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.

**Formal definition of a Deterministic Finite Automaton**

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

Q is a finite set of states.

∑ is a finite set of symbols called the alphabet.

δ is the transition function where δ: Q × ∑ → Q

q0 is the initial state from where any input is processed (q0 ∈ Q).

F is a set of final state/states of Q (F ⊆ Q).

**Graphical Representation of a DFA**

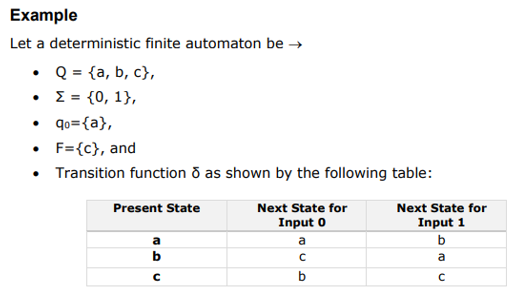
A DFA is represented by diagraphs called state diagram.

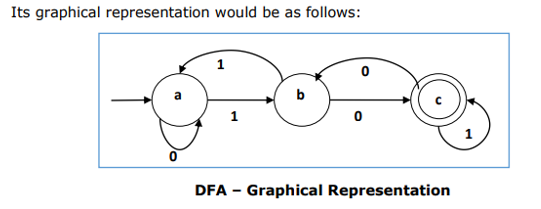
* The vertices represent the states.
* The arcs labeled with an input alphabet show the transitions.
* The initial state is denoted by an empty single incoming arc.

The empty string ε is just the string with no characters. The presence of a b-transition from q1 to q2 means that the automaton can move between those two states by reading character b but the presence of an ε-transition from q1 to q3 doesn't mean that it moves by reading the character ε.

It is a sequence of symbols from the alphabet with length zero. The empty string is never a symbol in the alphabet

* The final state is indicated by double circles.





**Non-Deterministic Finite Automaton (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 a NDFA**

An NDFA can be represented by a 5-tuple (Q, ∑, δ, q0, F) where −

Q is a finite set of states.

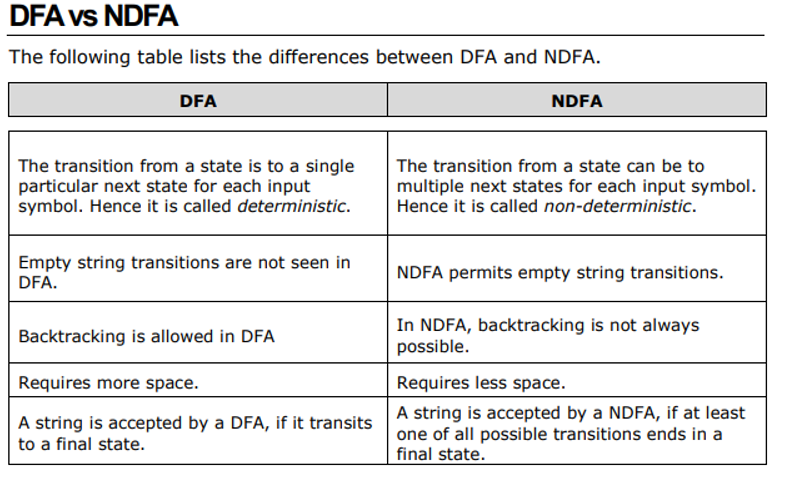
∑ is a finite set of symbols called the alphabets.

δ is the transition function where δ: Q × ∑ → 2Q

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

q0 is the initial state from where any input is processed (q0 ∈ Q).

F is a set of final state/states of Q (F ⊆ Q).



Acceptors, Classifiers, and Transducers

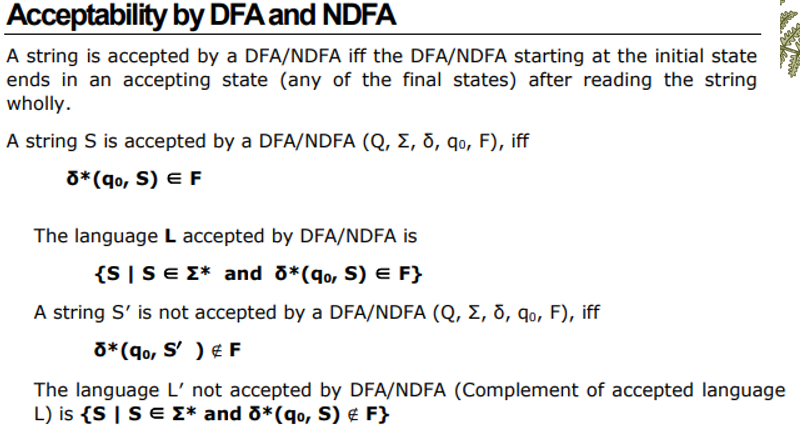
**Acceptor** (Recognizer) - An automaton that computes a Boolean function is called an acceptor. All the states of an acceptor is either accepting or rejecting the inputs given to it.

**Classifier** - has more than two final states and it gives a single output when it terminates.

**Transducer** - An automaton that produces outputs based on current input and/or previous state is called a transducer. Transducers can be of two types:

* Mealy Machine - The output depends both on the current state and the current input.
* Moore Machine - The output depends only on the current state.

Acceptability by DFA and NDFA



Automata Real-life Examples, Applications, and Importance

**Real-life Examples**

* Automatic Printing Machines
* Artificial Card Punching Machine
* Human Detection

**Where are Automata used in real-life?**

* Modeling and Verification of Software
* Distributed Systems
* Real-time Systems
* Structured Data

**Why do we need Automata Theory?**

Automata theory is important because it allows scientists to understand how machines solve problems. An automaton is any machine that uses a specific, repeatable process to convert information into different forms. Modern computers are a common example of an automaton.

*Verbatim:*

Automata enables programmers to understand how machines compute the functions and solve problems. Through automata, we see the logic, and if something is doable.

The main motivation in developing automata theory is developing methods to describe and analyze the dynamic behavior of discrete systems.

Computer Scientists were able to understand how machines compute functions and solve problems, and more importantly, what it means for a function to be defined as computable or for a question to be described as decidable.

**Others**

Greek Symbols: <https://skylinecollege.edu/boo/greek.php>