# Theory Of Computation

## UNIT-1

→ What is TOC?

In theoretical computer science, the theory of computation is the branch that deals with whether and how efficiently problems can be solved on model of computation using an algorithm.

Theory of computation

- **Automata theory**: It deals with the defination & properties of various mathematical model of computer
- **Computability theory**: It deals with what can and cannot be computed by model
- **Complexity theory**: It groups the computable problem based on their hardness.

It enables to understand how machines compute the functions and solve problems.

Eg- Set of binary strings that ends with 0.
Set of binary strings that represent a legal java code - Yes.

TOC tells -
- What we can compute
- What we cannot compute
- What is model of computation
- How we can compute.

TOC is also called as "Automata Theory".

# Automata theory :-

(1) Symbols - smallest building block, which can be any alphabet, letter or any picture.
a, b, c, 0, 1 ....

(2) Alphabets ($\Sigma$) - It is set of symbols, which are always finite.
$\Sigma = \{1\}$ → Unary
$\Sigma = \{0, 1\}$ → Binary no's
$\Sigma = \{0-9\}$ → decimal digit

(3) String - Finite sequence of symbols taken from a particular alphabet.
Denoted by $w$ & length of string is denoted as $|w|$.

$\Sigma = \{0, 1\}$ $\quad$ $\Sigma = \{a, b\}$
01101 $\quad$ abba

Eg - If $|w| = 2$ then no. of strings = 4
For alphabet $\{a, b\}$ with length n, number of strings can be generated $= 2^n$.

(4) Language - Set of string that are always defined on alphabet.
A language is a subset of $\Sigma^*$
A language formed over $\Sigma$ can be finite or infinite.
Eg - Ka$^{H}$ - neither eng. nor hindi
$L_1 = \{$ Set of all strings of length 2$\}$
$= \{aa, ab, ba, bb\}$ - finite language

→

$L_2$ = {Set of all strings which starts with 'a'} (2)
= {a, aa, ab, aba, aba, aaa, abb, ---} → Infinite language.

(5) Powers of alphabet ($\Sigma$) –

Say $\Sigma = \{a, b\}$ then

$\Sigma^0$ = Set of all strings over $\Sigma$ of length 0.
$\{\varepsilon\}$

$\Sigma^1$ = Set of all strings over $\Sigma$ of length 1.
$\{a, b\}$

$\Sigma^2$ = Set of all strings over $\Sigma$ of length 2.
$\{aa, ab, ba, bb\}$

(6) Length of string – Counting the frequency / occurance of symbol in a string.

(7) Kleene Closure ($\Sigma^*$) – It contains all possible strings of any length defined over an alphabet including null string.

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \text{---}$$

Eg – {00, 01} → Subset of {0, 1}.

"A language is always a subset of Kleen Closure".

Grammar —(Generates)→ Language ←(Recognize)— Automa

Automata : A algorithm or program that automatically recognize if particular string belongs to the language or not, by checking the grammar of the string.

The automata consist of
States (represented by circles)
& transitions (represented by arraws)
(jump)

Uses –
Compiler design and parsing

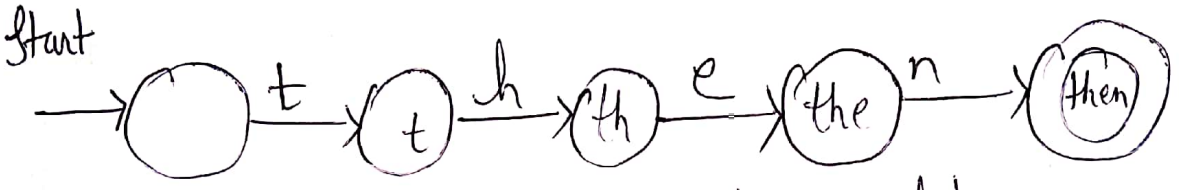



Fig. Finite automata model

As the automata sees a symbol of input, it makes a transition or jumps to another state, according to its transition function (which takes the current state & recent symbol as its inputs).

States – is an instantaneous description of that system which gives all relevant information.

Transition – are changes of states that can occur spontaneously or in response to inputs to the states.

## → FINITE AUTOMATA :-

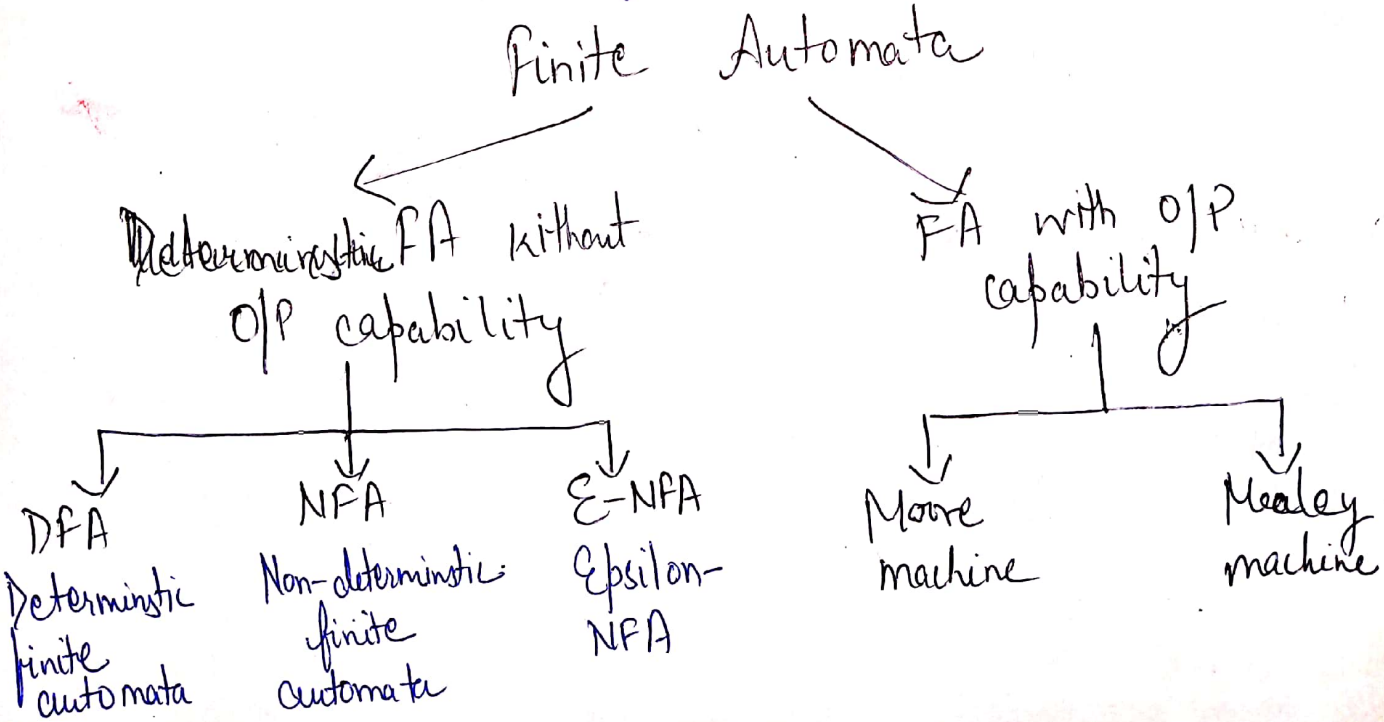
Finite automata is the simplest machine to recognize patterns.
It is a machine that will answer Yes/No.
A system containing only a finite number of states and transitions among them is called as finite-state transition system or finite system.

A finite automata consist of the following :—

$$\{Q, \Sigma, q, F, \delta\}$$

where —

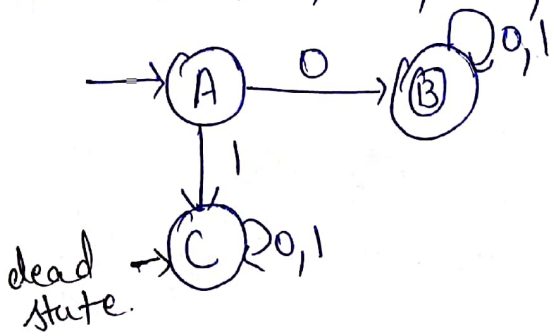
$Q$ = Finite set of states
$\Sigma$ = Set of Input symbols.
$q$ = Initial state
$F$ = Set of final states
$\delta$ = Transition function.



Examples of finite automata:—

(1) Set of all binary strings starting with zero.

$\Sigma = \{0, 01, 00, 001, 010, 011 \ldots\}$ — Infinite representation




→ Finite representation

* Any system as a m/c answered in a finite rep. as Yes/No but not in infinite representation

"001"

$A \xrightarrow{0} B \xrightarrow{0} B \xrightarrow{1} B$

So, when from initial we reach final state assumming all states then m/c says Yes otherwise No.

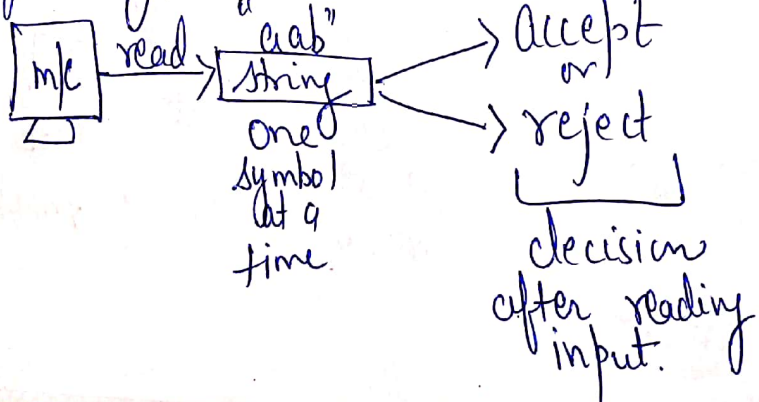
"101"

Reject as we didn't reach the final state

Finite automata without O/P capability

(1) Deterministic finite automata-

Informally,



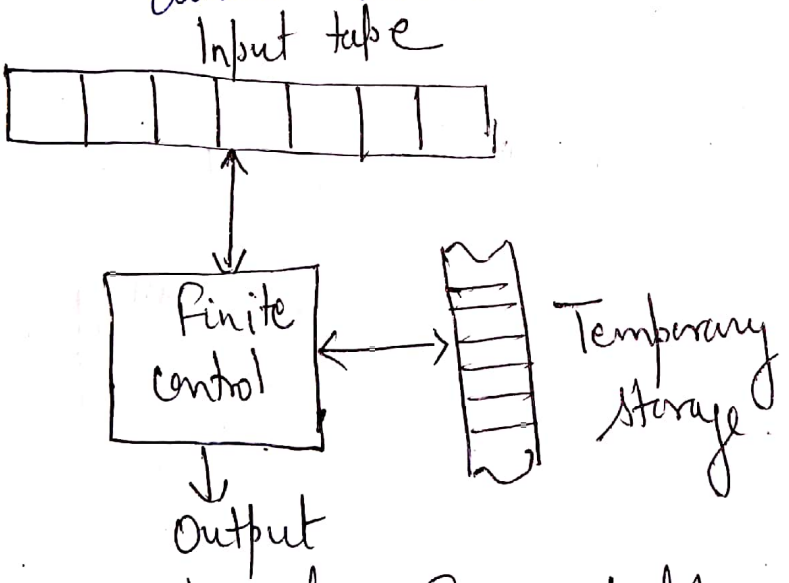
So, A m/c for which deterministic code can be formulated if there is only one unique way to formulate the code then m/c is called as DFA.

DFA consist of 3 parts:—

1) Tape — to hold input string.
   Divided into finite no of cells.
   Each cell holds a symbol from $\Sigma$

2) Tape head — For reading symbol from tape

3) Control —
   - Finite no. of states that m/c allows
   - A current state, initially a start state
   - State transition function for changing current state.




DFA consist of five tuples as—

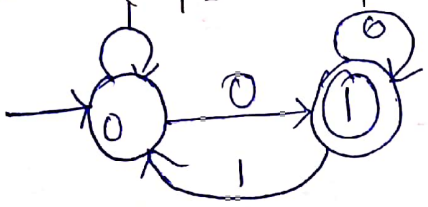
$$\{Q, \Sigma, q, F, \delta\}$$

where

$Q$ = set of all states
$\Sigma$ = Input symbols
$q$ = Initial states
$F$ = final states
$\delta$ = transition function

$\delta = Q \times \Sigma = Q$

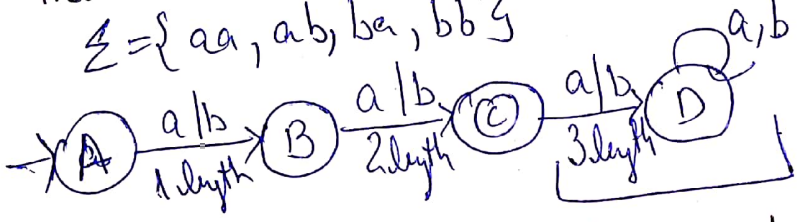
Eg- $\Sigma = \{0,1\}$ accepts all strings ending with 0.



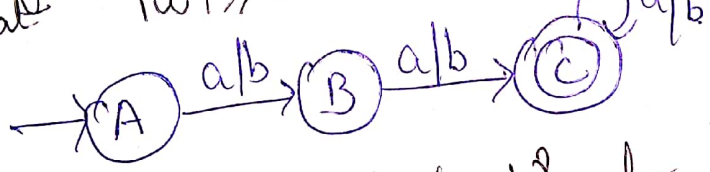

*Note- There can be many possible DFAs for a pattern but minimum no. of states is preferred.

Q- Design a DFA over the alphabet $\Sigma = \{a,b\}$ that will accept all strings w such that $|w| = 2$.

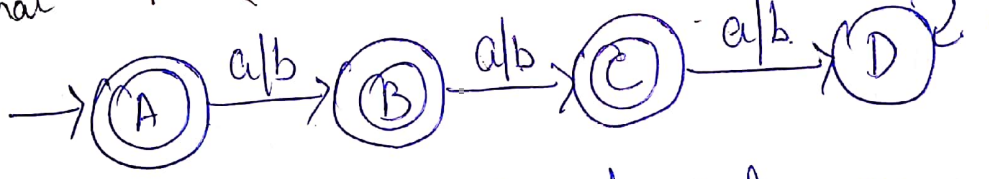
$\Sigma = \{aa, ab, ba, bb\}$




Q. Design a DFA $\Sigma = \{a,b\}$ for all strings such that $|w| \geq 2$




Q. Design a DFA $\Sigma = \{a,b\}$ for all strings such that $|w| \leq 2$ .




* If initial state is treated as the final state then it means $\varepsilon$ move is acceptable.

$\{\varepsilon, a, b, aa, ab, ba, bb\}$