Theory of Computation

Automata TOC :-

Theoretical branch of Computer Science and Mathematical Studying abstract machines and problems they can solve.

Abstract machines :- simplified models used to study and understand computationals.

Abstract Machines

Finite Automata | Pushdown Automata | Turing Machines

* Deals with logic of computation with respect to simple machines.

Basic Terminologies of TOC :-

Symbol :-

A symbol is also called a character, alphabet.

a, b, c, 0, 1 ....

Alphabets :-

Denoted by (Σ) :-

set of finite symbols.

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

$\Sigma = \{a, b, c\}$

$\Sigma = \{1, 2, 3 ...... 9\}$

String :-

A string is a finite sequense of symbol s from some alphabets.
Denoted with w and its length is denoted by |w|

* Empty string is the string with zero symbols denoted by 'ε'

Eg:- No. of strings with length 2 generated over alphabet {a, b}:

{a, a} {a, b} {b, b} {b, a}

⇒ (aa, ab, bb, ba)

Length of string |w| = 2
Number of string = 4

Formula generated a string with m no. of alphabets with n length ⇒

$$\boxed{M^n}$$

Closure Representation in TOC :-

L^+ :

Positive closure that represents a set of all strings except Null or ε-strings.

L^* :

Kleene Closure, represents a set off all strings including null or ε-strings.

$$\boxed{L^* = \varepsilon L^+}$$

Eg:- Regular expression for language accepting all combination of g's over $\Sigma = \{g\}$:

$$R = g^*$$

$$R = \{\varepsilon, g, gg, ggg, ggg, gggg, \ldots\}$$

Regular Expression for language accepting all combination of g's over $\Sigma = \{g\}$:

$$R = g^+$$

$$R = \{g, gg, ggg, gggg \ldots\}$$

Kleene star is an infite set but if we provide any grammer rules then it can work as a finite set. (ϵ is included in Kleen closure)

Language :-

A language is a collection of appropriate string.

$L1 = \{\text{Set of Strings}\}$

Finite Automata

* automata used to recognize patterns.
* It takes the strings of symbol as input and changes its state accordingly when desired symbol is found, then the transition occurs.
* at the time of transition, the automata can either move to the next state or stay at the same state.
* FA has two states accept or reject state.
  when string is accept → accept; reject → when its rejected.

Finite automata is a collection of 5 tuples $(Q, \Sigma, \delta, q_0, F)$, where:
Q = finite set of states
Σ = finite set of input symbol
q_0 = initial state
F = Final State
δ = Transition function

Types of automata :-

Finite Automata
→ Deterministic Finite Automata (DFA)
→ Non-deterministic finite automata (NDFA) or (NFA)

Deterministic Finite Automata (DFA)
* Machine goes to one state only for a particular input character.
* Does not accept null move

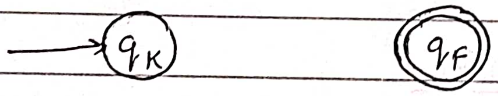
Non-deterministic finite automata (NDFA) or (NFA)
* Any number of states for a particular input.
* can accept null move.

* Every DFA is NFA, but NFA is not DFA
  (Every apple is a fruit but all fruits are not apple)
* Both NFA and DFA can have multiple final states.

Transition Diagram

A transition diagram or state transition diagram is a directed graph which can be constructed as follows :-

* A finite set of states, at least one of which is designed the start state and some of which are designated as final states.
* An alphabet Σ of possible input symbols from which the input Strings are formed.
* A finite successful path through the transition graph is a series of edges forming a path beginning at start state and ending at one of the final states.
* A finite set of transitions that show the change of state from the given state on a given input.



Start State and states other than final states

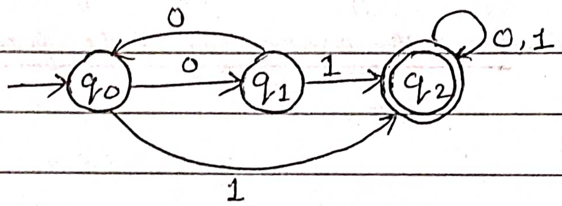
Final State

Transition Table

Tabular representation of transition function. It takes two arguments State and symbol and returns a state (the "next state").

* Column corresponds to input symbols
* Rows corresponds to states
* Entries corresponds to the next state.
* The start state is denoted by an arrow with no source.
* The accept state denoted by star.
  ↳ final

Eg:-



Transition Table of given DFA :-

Present State	Input	
	0	1
→ q_0	q_1	q_2
q_1	q_0	q_2
* q_2	q_2	q_2