Date 08/09

Theory of Computation (TOC)

TOC is a study of mathematical computation problems & various computational models which are used to solve such problems.

It has 3 aspects:

1) Complexity Theory
2) Computability Theory
3) Automata & Language Theory.

1. **Complexity Theory**

It deals with the classification of computational problems as per the computational difficulties.

In other words it classifies the mathematical problems as easy or hard. It uses time 'n' space as measures for computational difficulties.

2. **Computability Theory**:- It deals with the classification of mathematical problems as solvable or unsolvable.

3. **Automata & Language Theory**:

It deals with various mathematical models of computation/ abstract models, there definations, properties & models.

→ These abstract machienes are known as automata.

→ These field also gives the formal definations for automobiles & their capabilities.

Based on their capabilities automata are of 4 types.

1) Finite Automata (FA) / Finite State Machines (FSM).
2) Push Down Automata (PDA)
3) Linear bounded Automata (LBA)
4) Turing Machine (TM)

Mathematical Notations

The concept of sets
 Unordered collections of objects
Sequences & Tuples
→ Ordered collections / set
→ Denoted → ( )
e.g $a = (2, 15, 23)$ Tuples = 3

Symbol:	Alphabet (Σ)	String (w)
a	Finite non empty	The finite sequence of
A	set of symbols	symbols taken from any
@	e.g $\Sigma = \{a, b\}$	alphabet usually written
;	$\Sigma = \{0, 1\}$	to each other without separated
+		by , or blank space.
		e.g $\Sigma = \{a, b\}$
		String (w) = ababa
		Length of string = $\lvert w \rvert$
		$\lvert w \rvert = 5$

Concatenation of a String/Symbol
→ Denoted by → '.'
w_1 & w_2 are two strings then there concatenation written as $w_1 . w_2$ or $w_1 w_2$ & obtained by appending w_2 to the end of w_1.
e.g. $\Sigma = \{a, b\}$
 $w_1 = aba$ $w_2 = ba$.
 $w_1 w_2 = ababa$

Reverse of a string (w^R)
→ It is obtained by writing symbols of String (w) in the opposite ordered.
e.g $w = ababa$, $w^R = ababa$.
 If $w = w^R$ then it is a palindrom over a given Σ.

Power set of Σ
Denoted as → 'Σ^n'

→ Set of all possible strings over $\Sigma = \{a, b\}$.

$\Sigma^n = 2^n$ $\quad$ n = length of strings ($n \in N$)

2 = no. of symbols in Σ.

→ Σ^n can be obtained by repeatedly concatenating the symbols from the Σ, 0 or more no. of times

$\Sigma^n = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots \cup \Sigma^n = \epsilon \cup (a,b) \cup (aa, ab)$.

$a^0 / b^0 = \boxed{\epsilon} \rightarrow$ null string $\quad = \{\epsilon, a, b, aa, ab, ba, bb, \ldots\}$

Σ^*: Set of all possible strings over Σ that can be obtained by concatenating symbols from 'Σ' zero / more no. of times.

→ The complete language over $\Sigma = \{a, b\}$:

⇒ $\boxed{\Sigma^*}$ – Klein Closure / Star Operation.

$\boxed{\Sigma^+}$ – +ve Closure of Σ.

Σ^+ : It is obtained by concatenating the symbols from 'Σ' 1 or more no. of times.

$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots \cup \Sigma^n$

$= \{a, b, aa, ab, ba, bb, \ldots\}$

⇒ $\boxed{\Sigma^* - \{\epsilon\} = \Sigma^+}$

Prefix & Suffix of Strings.

→ If $y = xz$ then the symbol x & string xz including the null string are its prefix

e.g. $w = abab$ $\quad$ Prefix $(w) = \{\epsilon, a, ab, aba, abab\}$

If length of w is n, then it has $n+1$ prefix ⇒ length$(w) = 4$, Prefix $= 5$

→ The set of all other prefixes of the given string excluding the string itself gives its proper prefixes.

If length of $w = n$, then its proper prefixes $= n$.

Proper $(w) = \{\epsilon, a, ab, aba\}$

Suffix

If $y = xz$, then the symbol & string xz includes the null string are its suffix. e.g. $w = abab$ $\rightarrow \epsilon.abab.\epsilon$

Suffix $(w) = \{\epsilon, b, ab, bab, abab\}$

Length $(w) = 4$, Suffix $(w) = 5$

classmate
Date
Page

Proper Suffix
All the suffixes of the string except for the string itself are the proper suffixes of the given string.

Language (L)
The set of strings over a given alphabet Σ is known as Language.

e.g. $\Sigma = \{a, b\}$

$L = \{ab, abb, bab, b, \ldots\}$

$\Sigma = \{a\}$

$L = \{a, aa, aaa, aaaa \ldots\} = a^+$

$L = \{w \mid w \in \Sigma = \{a\}$ that contains one or more no of a's.$\}$

Types of Proof's
1. Proof by Construction
2. Proof by Contradiction
3. Proof by Induction.

Automata Language Theory

This field of TOC is the study of various computational machine /abstract machine which are used to solve various computational problems.
Also this field gives us the defination, properties & cababilities of such machine. These abstract machines also known as Automata (Automachine).

→ An Automata is an abstract computation machine that is used to solve certain mathematical problems.
Just like any other machine, an automation has essential or primary characteristics. It has 3 main characteristics

i) States: It gives the present state information of an automation at any point of time.

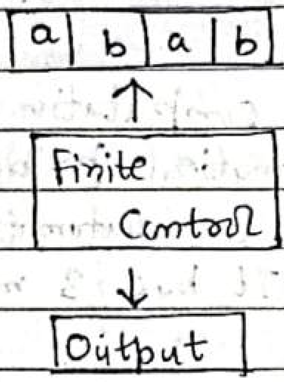
ii) Input: At any instance of time the input supplied to an automation is represented using this characteristics.

iii) Transitions / Behaviour: At any instance of time from any given state on receiving an input symbol how the automata in behaving is defined by the transition of the automata. Usually represented using a transition function(δ)

At any point of time an automaton can be in any one the states, reads the input from the input Tape, may use some kind of temporary storage or may not use any storage elements. In the next step it may change to some other state based on the input receive or it may remain in the same state or in the end of the input processing it gave it output.
Based on their capabilities automation are of 4 types

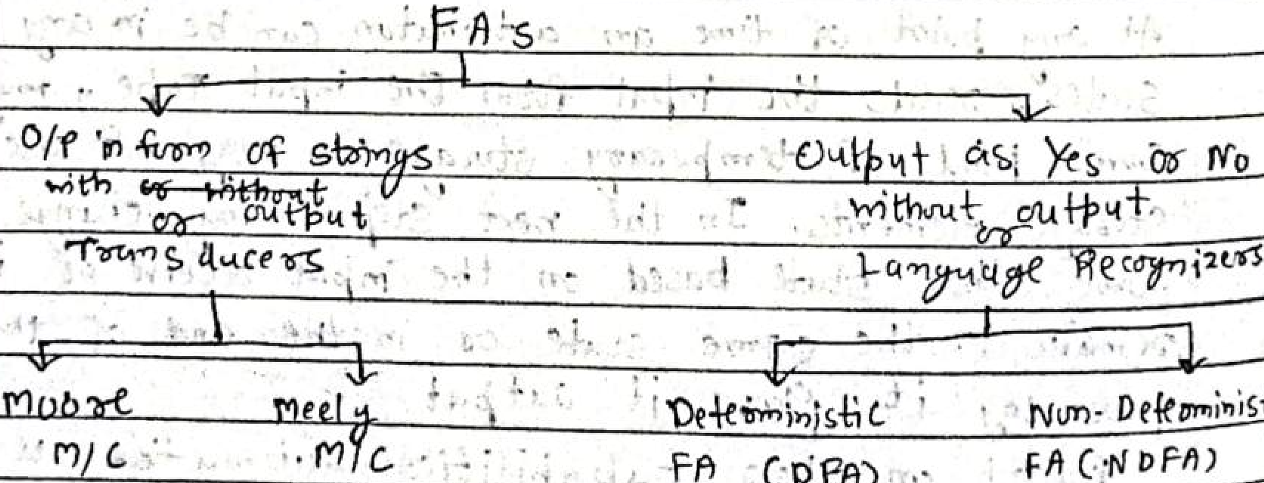
1. Finite Automata

This is the simplest type of automata that has no additional storage elements. This type of automata has finite no. of states Thus otherwise known as Finite states Machine (FSM)



At any point of time, a FA can be in any 1 of the finite no. of states, reads the input symbol from the I/p Tape from left to right, one symbol at a time. In the next step the finite automata may remain in the same state or may change state based on the I/p received. at the end of the I/p string the FA gives it O/p.

. Based on the type of O/P provided by the FA they are divided into two categories.

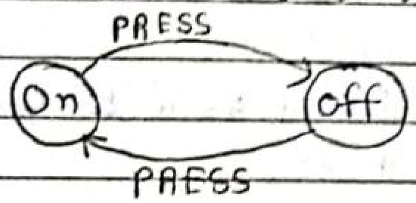


Notation to represent FA

There are two notations used to represent any type of Automata:-

1. State - Transition Diagram.

In this representations circles are used to represent states & directed edges labelled the input symbol used to represent the transition from 1 to other state.



Always a FA (every other automata as well) starts processing the inputs from a state known as the initial state or start state.

→ In this method a circle / state pointed by an directed edge with no origin is represented as an initial state

And a FA always accepts the input string if and only if at the end of the input string it enters to an accepting state. In this method an accepting state / final state is represented using a double circle ◎.

There is exactly one initial state for every type of automata. Where as there can be more than one final state for every type of automata.

2. Transition Table representation.

In this method the automata are represented using tables where the rows of the table represents the states of automata & the columns of the table represents the input symbols. The value under every (row, column) pair represents the transition from one state to other.

In this method the initial state / start state is marked by using an directed edge pointing from no where to a particular state and an accepting state is represented using an * astrisk mark over the given symbol.

States \ Input	PRESS
→ ON	OFF
OFF*	ON

A finite automata is formally defined as a 5 tuple as follows
$FA = (Q, \Sigma, \delta, q_0, F)$.

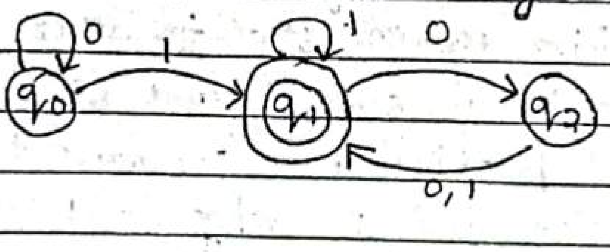
Q: Finite non empty set.
Σ: Input alphabet.
δ: Transition function.
q_0: Initial / start state ($q_0 \in Q$)
F: Finite set of Accepting / Final states ($F \subseteq Q$)

Deterministic Finite Automata

A DFA is the simplest FA that uses Deterministic computation. in other words in a DFA from any single state on any single input symbol at any point of time there is exact one path of computation / exactly one transition defined.

Acceptance Mechanism of DFA

Starting from the initial state, reading the input string from input tape from left to right one symbol at a time at the end of processing the input string if a DFA enters accepting / final state then the input string is said to be accepted otherwise rejected



formally a DFA is represented as a 5 tuples.

$D = (Q, \Sigma, \delta, q_0, F)$.

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

$Q = \{q_0, q_1, q_2\}$

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

$\delta : \delta(q_0, 0) \rightarrow q_0$ $\quad \delta(q_2, 0) \rightarrow q_1$

$\delta(q_0, 1) \rightarrow q_1$ $\quad \delta(q_2, 1) \rightarrow q_1$

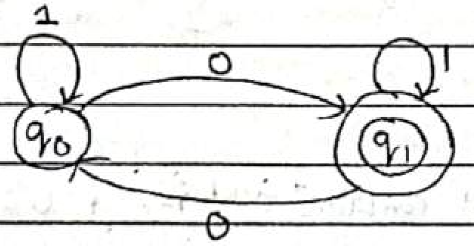
$\delta(q_1, 0) \rightarrow q_2$

$\delta(q_1, 1) \rightarrow q_1$

$q_0 : \{q_0\}$

$F : \{q_1\}$

m1



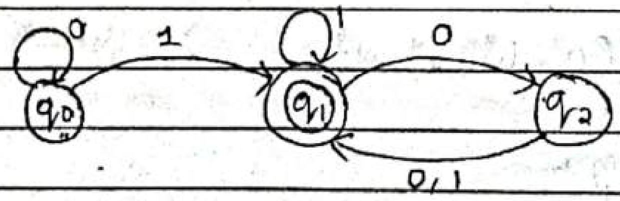
$m_1 = (\{q_0, q_1\}, \{0, 1\}, \delta, \{q_0\}, \{q_1\})$

$\delta : \delta(q_0, 0) \rightarrow q_1$

$\delta(q_0, 1) \rightarrow q_0$

$\delta(q_1, 0) \rightarrow q_0$

$\delta(q_1, 1) \rightarrow q_1$



$w = 1$ trace fro i/p $w = 1$

$\delta(q_0, 1) \rightarrow q_1$

$w = 01$.

$\delta(q_0, 01) \rightarrow \delta(q_0, 1) \rightarrow q_1$

$w = 00$ X

$\delta(q_0, 00) \rightarrow \delta(q_0, 0) \rightarrow q_0$

w = 1011 ✓

$\delta(q_0, 1011) \to \delta(q_1, 011) \to \delta(q_2, 11) \to q_1$

w = 1010100 ✓

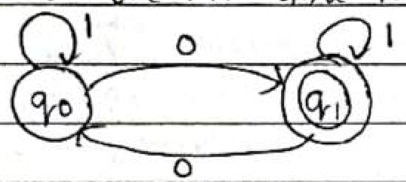
w = 10111010 ×

w = 000100 ✓

w = 11101 ✓

$L(m_1) = \{w \mid w \in \{0,1\}$ that contains atleast one '1' and the last 1 in the string w will always followed by even no. of 0's }.

Q Consider the DFA and identify the language recognised.



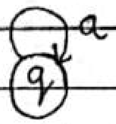
$L(m_2) = \{w \mid w \in \{0,1\}$ that contains odd no. of 0's }.

w = 0

w = 10

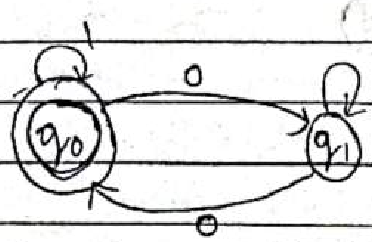
w = 01

w = 101



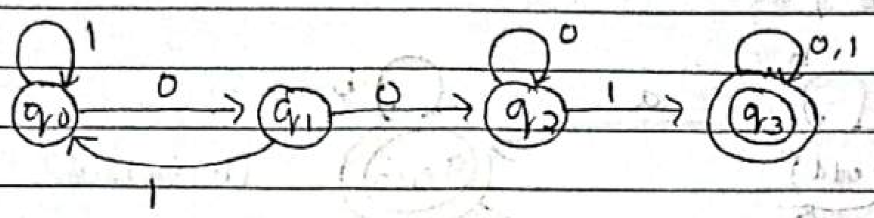
$a^* \leftarrow \{a^n \mid n \geq 0\}$ ϵ, a, aa.

|odd '0's| $(0^0, 0^2, 0^4, 0^6, \_\_\_\_) = (00)^*$



$L(m_3) = \{w \mid w \in \{0,1\}$ that contains even no. of 0's }

Note → Any Automata in which the initial state itself a final state excepts ϵ by default or vice versa.



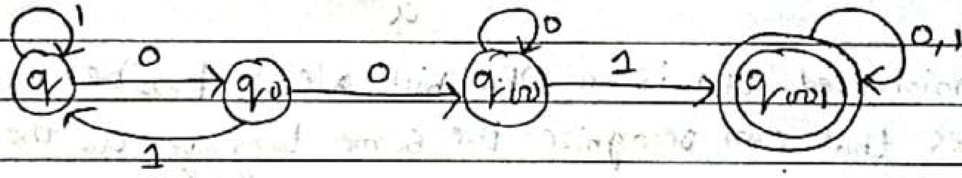
$(0+1)^* \; 001 \; (0+1)^*$

Designing of DFA.

Q Design a DFA for a given language.

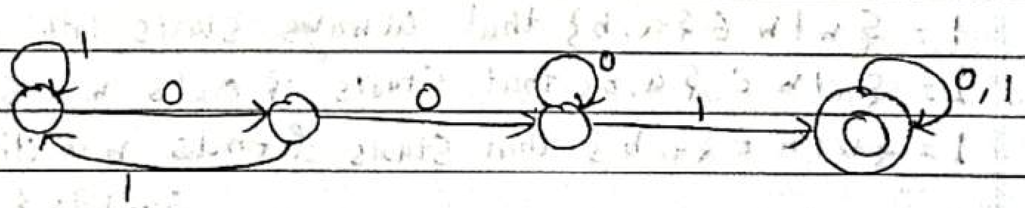
$L = \{ w \mid w \in \{0,1\}$ that contains 001 as substring $\}$.

$= \{ 001, 0001, 1001, 00011$



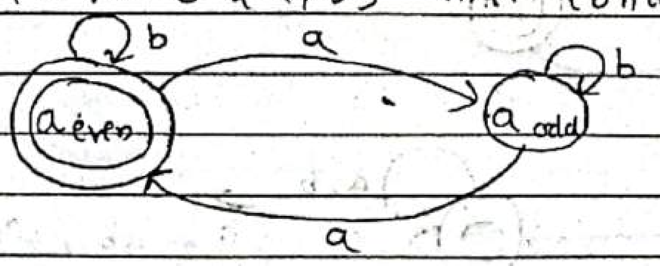
$W = 001$

$|W| = 3$, No. of state $= 3 + 1 = 4$.



Design a DFA for a given language.

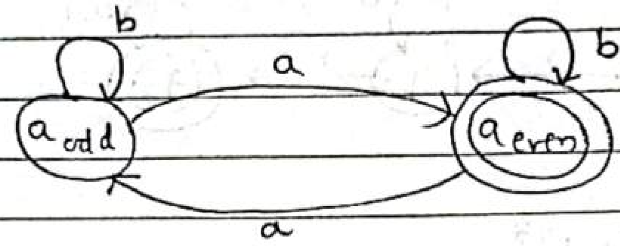
$L = \{ w \mid w \in \{a,b\}$ that contains even a's $\}$



Design a DFA for the given language.

$L = \{w \mid w \in \{a,b\}$ that contains odd a's $\}$

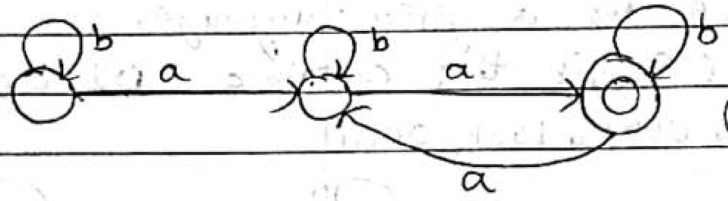
$= \{\epsilon, aa$



(Minimized)

$w = aa$

$|w| = 2$, No. of States $= 2+1 = 3$



(Regular)

A minimized DFA is a DFA with a least possible no. of states that can recognize the same language as the regular.

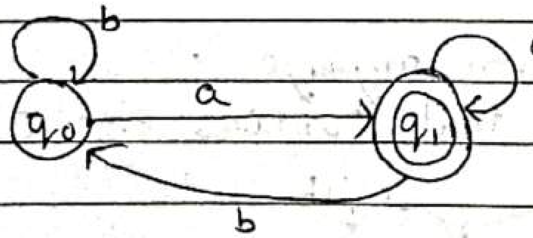
Design a DFA for following Languages.

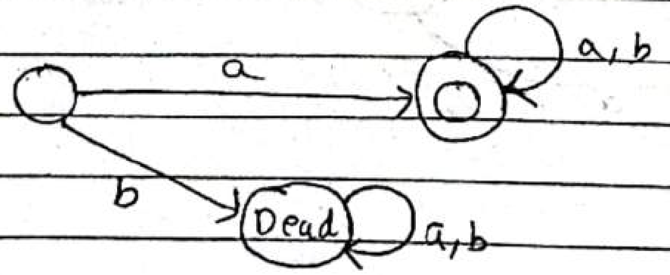
$L = \{w \mid w \in \{a,b\}$ that ends with 'a' $\}$

$L = \{w \mid w \in \{a,b\}$ that always starts with an 'a' $\}$

$L = \{w \mid w \in \{a,b\}$ that starts & ends with same symbol $\}$

$L = \{w \mid w \in \{a,b\}$ that starts & ends with different symbols $\}$





a, ab, $a^n b^n$ ✓

$b^n a^n$ ✗