AUTOMATA, LANGUAGES AND COMPUTATION

11/09/2023

→ **Automata**: Machines which can perform tasks automatically →
- → Finite devices
- → Infinite devices

→ Finite Automata.

UNIT-1

→ An automata (or) automaton is a machine designed to respond to encoded instructions (Robot).

Auto: self **Meta**: machine.

→ Code written should be compact.

Ex: Automatic washing machine, automatic machine tools; etc.

→ Automata Theory (or) Theory of computation describes the basic idea and models underlying computing.

I/P $\longrightarrow$ $\boxed{\text{Computation}}$ $\longrightarrow$ O/P

→ Each abstract computing machine recognizes formal language.

→ Formal language recognizes or contains encoded instructions

→ Applications:

- → LST for logical circuits
- → Robotics
- → Compiler Construction
- → Designing of Editors (Text Editors)
- → Natural Language Processing
- → Finite State Machines (FSM) etc.

Symbol, Alphabet, string/word, String operation, Language, Language operations, Formal Language, Grammar, Problems.

Chomsky hierarchy.

<u>Symbol</u>: abstract entity which is not defined (one or more characters).

<u>Alphabet</u>: non-empty finite set of symbols (Σ)

$\boxed{\Sigma = \{a, b, c\}}$ → String can only contain a, b, c.

<u>String/Word</u>: finite sequence of symbols over an alphabet 'Σ'.

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

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

<u>String operations</u>:

- → Concatenation : combining 2 strings with no space.
- → Length: length of a word/string. ($|w|$)
- → Empty String (ϵ w)
- → Reverse of a string (w^R)
- → substring
- → prefix (start)
- → suffix (suffix)
- → Proper prefix & Proper suffix.

Ex: $w = abc$

Prefix : ϵ, a, ab, abc Proper prefix : ϵ, a, ab

Suffix : ϵ, c, bc, abc Proper suffix : ϵ, c, bc

└→ Set of Strings : (Σ^*) 0 (or) more.

$\boxed{\text{Empty Set} : \emptyset}$ $\Sigma^+ = \Sigma^* - \{\epsilon\}$ (1 or more)

Language is a set of string of symbols from some one alphabet (Σ)

* A language L is a subset of Σ^*

<u>Language Operations</u>:

→ Compliment of Language (L')

$$\boxed{L' = \Sigma^* - L}$$

→ Reverse of Language (L^R)

$$L^R = \{w^R \mid w \in L\}$$

→ **Concatenation of 2 languages:**

L_1, L_2 : 2 languages

$L_1 \cdot L_2$ → concatenation

→ **Closure operation of a language**

language including null string : Star/Kleen Closure.

($L^* \approx \Sigma^*$)

language excluding null string : +ve closure

($L^+ \approx \Sigma^+$)

→ **Formal Language**

A set of strings of symbols from some alphabet (Σ) is called a formal language. (empty and finite)
↳ set can be

→ **Formal Grammar:**

Structure of a language.

$G = (V, T, S, P)$

(capital letters) V = non-empty set of variables

(lower case) T " " terminals

$S \in V$ = start variable

P = set of production rules

15/9/23

* Empty language $L = \{\phi\}$
* Empty string $= \epsilon$

⇒ L_1, L_2, L_3 → $\boxed{\text{Machine}}$ → Accept / reject

* $\boxed{M \leftrightarrow L}$ → for every machine there is a language and vice-versa.

General form of productions:-

$P : \alpha \to \beta$ where

$\alpha \in (V \cup T)^+$ and $\beta \in (V \cup T)^*$

$P : \alpha \to \beta$ → α derived β

α induced to β

where α . contains variables

Grammar or formal Grammar:-

Variables : Upper case

Terminals : a to z and digits 0 to 9 and some special operators

Eg:- $V = \{S, A, B\}$ and $T = \{a, b\}$ where S is a start variable, then productions P

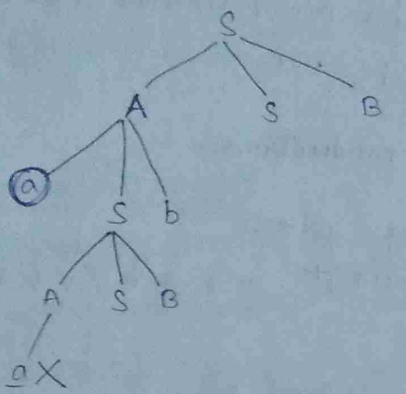
$S \rightarrow ASB$ → 1 production

$A \rightarrow aSb \mid \epsilon$ → 2 productions either aSb or ε

$B \rightarrow bSa \mid \epsilon$ → 2 productions. either bSa or ε

Total productions $P = 1+2+2 = 5$

→ if i/p string $\boxed{w = \text{"abb"}}$ (better start deriving from left most)



already 'a' is present, 'b' is not derived so it is not accepted.

Problem:-

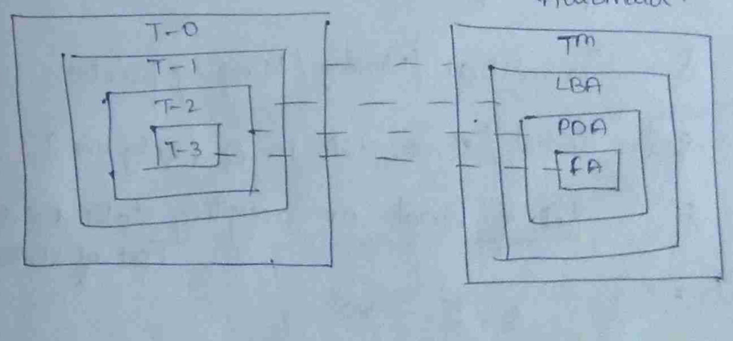
"Given a string w in Σ^*, decide whether or not w is in L"

Chomsky Hierarchy of Formal Languages:-

Grammar Type	Grammar Accepted	Language Accepted	Automaton
Type-0	Unrestricted grammar	Recursively enumerable language	Turing machine
Type 1	context-sensitive grammar	Context-sensitive language	Linear-bounded automaton
Type-2	Context-free grammar	Context-free language	Pushdown automaton.
Type-3	Regular grammar	Regular Language.	Finite state automaton.

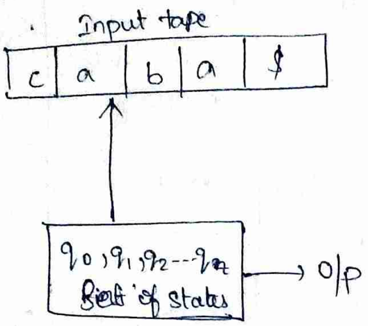
Type 3 < Type 2 < Type 1 < Type 0.

Automata.



Finite Automata:-

FA or Finite state Machine (FSM) represents a machine that takes input and produces output



FA: Quin Tuple or 5-Tuple denoted by M

$M = (Q, \Sigma, \delta, q_0, F)$

Q :- Finite or non empty set of states or internal states

Σ : input Alphabet

δ : Transition / Moving / Mapping function

q_0 : Initial / start state in Q (only one)

F : Set of Final or accepting states, $F \subseteq Q$ (set of states)

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

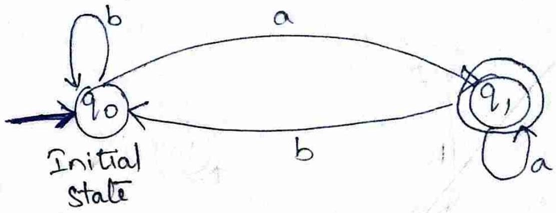
Example:-

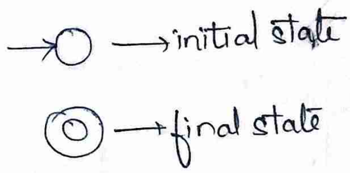
$M = \{Q, \Sigma, \delta, q_0, F\}$

$Q \rightarrow \{q_0, q_1\}$, $\Sigma \rightarrow \{a, b\}$, $q_0 \rightarrow q_0$, $F \rightarrow \{q_1\}$

Transition function: $\delta(q_0, a) = q_1$, $\delta(q_0, b) = q_0$, $\delta(q_1, a) = q_1$, $\delta(q_1, b) = q_0$

State diagram:-





① Suppose, i/p is abb

$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_0 \xrightarrow{b} q_0$ ✗

Not accepted. ↳ since final state is not reached

② abba

$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1$ ✓

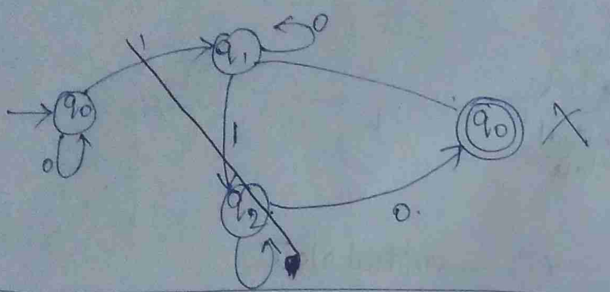
Since final state is reached accepted.

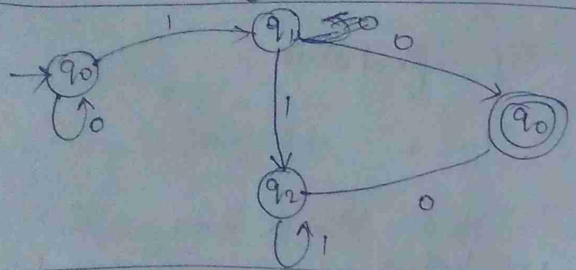
Q ababbab

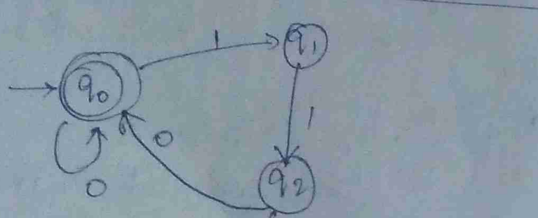
$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_0 \xrightarrow{b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_0$

→ Not reached to final state
→ So not accepted.
→ Current state is q_0

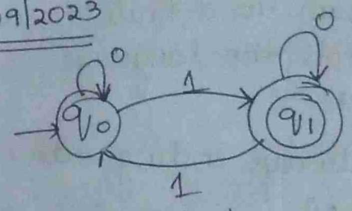
$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_0\})$







19/09/2023

* 

a) 01101101

$q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1$

⇒ accepted

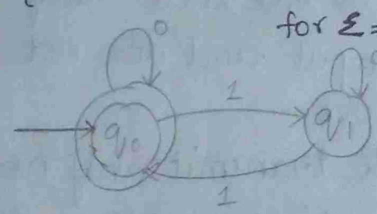
b) 10011010101

$q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_1 \rightarrow q_0$

⇒ Not accepted.

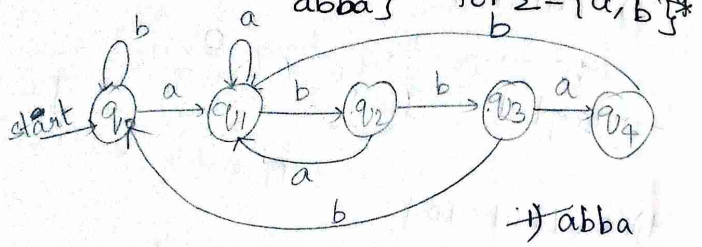
→ $L = \{w \mid w$ contains odd no. of 1's$\}$ for $\Sigma = \{0, 1\}^*$

→ $L = \{w \mid w$ contains even no. of 1's$\}$ for $\Sigma = \{0, 1\}^*$



→ construct state diagram for a finite automata that accepts the language which is defined as

$L = \{w \mid w$ is always ending with 'abba'$\}$ for $\Sigma = \{a, b\}^*$



1) abba
2) babba
3) ababbabba
4) ababbab

* **Deterministic Finite Automata:** $\boxed{DFA \subseteq NFA}$

→ Deterministic refers to the uniqueness of computation.

→ Machine goes to one state only for a particular input and does not accept null move.

→ NFA is used to transmit any no. of states for a particular I/P and accept the null move.

→ NFA : multiple choices → Theoretical concept.

→ DFA : only one choices → Lexical Analysis

* $(Q, \Sigma, \delta, q_0, F)$

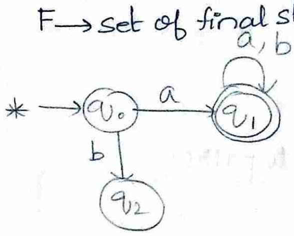
$Q \rightarrow$ set of finite states.

$\Sigma \rightarrow$ finite set of symbols (alphabet)

$\delta \rightarrow$ transition fn $\boxed{\delta : Q \times \Sigma \rightarrow Q}$

$q_0 \rightarrow$ initial state.

$F \rightarrow$ set of final states of Q.

* 

$L = \{w \mid w$ start with a$\}$ for $\Sigma = \{a, b\}^*$

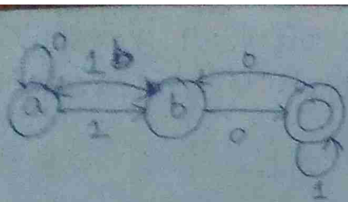
* Draw state diagram for given DFA.

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

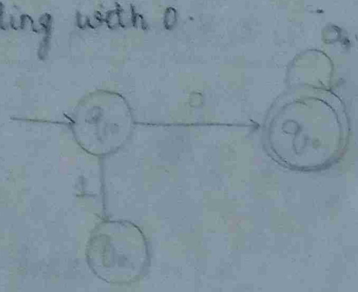
$\Sigma = (0, 1)$, $q_0 = \{a\}$, $F = \{c\}$

and transition function table:

Present state	Next state I/P = 0	Next state I/P = 1
a	a	b
b	c	a
c	b	c



* Draw state transition diagram for given DFA with $\Sigma = \{0, 1\}$ accepts all starting with 0.



* Extensions of transitions to paths:

→ Basis: $\hat{\delta}(q, \epsilon) = q$

→ Induction: $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$

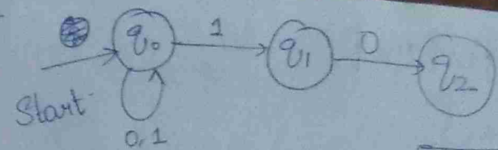
→ $L(A) = \{w : \delta(q_0, w) \in F\}$

* Non-deterministic Finite Automata:

Formal defn:

$(Q, \Sigma, \delta, q_0, F)$

*



δ	0	1
q_0	$\{q_0\}$	q_0, q_1
q_1	$\{q_0\}$	$\varnothing$
q_2	$\varnothing$	$\varnothing$

$\hat{\delta}(q_0, \epsilon) = \{q_0\}$

$\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \epsilon), 1)$

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

$\hat{\delta}(q_0, 10) = \delta(\hat{\delta}(q_0, 1), 0) = \delta(\{q_0, q_1\}, 0)$

$= \delta(q_0, 0) \cup \delta(q_1, 0)$

$= \{q_0\} \cup \{q_2\}$

$= \{q_0, q_2\}$

$\hat{\delta}(q_0, 101) = \delta(\hat{\delta}(q_0, 10), 1) = \delta(\{q_0, q_2\}, 1)$

$= \delta(q_0, 1) \cup \delta(q_2, 1)$

$= \{q_0, q_1\} \cup \varnothing$

$= \{q_0, q_1\}$

$\hat{\delta}(q_0, 1010) = \delta(\hat{\delta}(q_0, 101), 0) = \delta(\{q_0, q_1\}, 0)$

$= \delta(q_0, 0) \cup \delta(q_1, 0)$