Automata theory and computation

01/09/2020

Introduction:

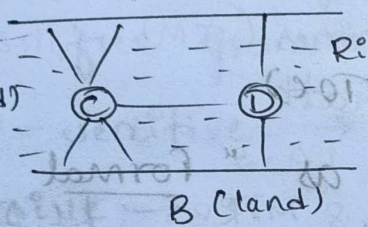
Algorithm: Time complexity and space complexity.

Two different kinds of problem.

a) Solvable problem
b) Unsolvable problem

a) Solvable problem: Here, the problem will be solved easily but we cannot solve some problem, in that time we should prove it by mathematical model / computation

Ex:- Konsbag's bridge problem

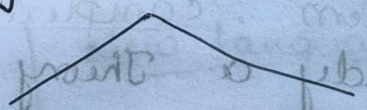


* "Euler theory" → Converted particular problem into graph → proved by mathematical model / computation

b) Unsolvable problem: Here we cannot solve a problem

Ex:- division by zero.

Theory of computation.



a) Solvable problem b) Unsolvable problem.

- a) decidability (decidable)
- b) undecidability (undecidable)

i) Decidability:- The problem will have both algorithm and procedure.

* We know the how much time is required to solve a given problem.

Diff blw algorithm and procedure.

Algorithm	procedure
* Time Complexity & Space Complexity	* Step by instruction cohich has to followed
* We know that how much time complexity & space Complexity is required to solve a problem	* Donot know that how much time required to Solve a problem.

ii) Undecidability :- Here, the problem having only procedure.

* Here, we donot know the how much time is required to complete a given problem.

4/9/2020

Theory of computation :- (TOC)

It is also known as "Formal language and automata theory (FLAT)".

IMP

* Stdy study of mathematical machine on systems Called "Automata" (OR) what is the complexity of solving a problem.

:> Why we need to study a Theory of computation?

=> What kind of limitation for solving a problem (OR) what is the complexity of solving a problem.

4 Parts of machine :-

a> finite automata :- It is a simplest machine, we can solve some kind of problem.

2021/02/04 12:59

b> PDA (post down automata):-

c) Linear bounded Automata (LBA) :-

d) Turning machine (TM) :- It is a hypothetical mode → Here any problem can be solved by using any compu -ter Science / machine

Three basic things :- LAG → Expand as

* Language
* Automata
* Grammer → basic building block of Theory of autom -ata (TOC).

four different parts of grammer :-

a) Regular grammer (RG) → LBA
b) content free grammer → finit
c) Content Sensitive → PDA
d) Reccursive → TM

LAG :- Language :- Communicate in natural way } 7/9/20.

V. IMP (def^n)

9) Symbol :- Smallest element of any particular language is called a "symbol" (basic building block)

English	New language design
a to z	A - Z } known as symbol
A to Z	0₁ - - - 0₁₀
Alphabet (a, b, - - - z)	Σ (a, b)
	Σ {} x

Imp

def^n of Symbol :- It is a basic building block which can be any character or token are called Symbol

Eg :- A - Z (or) 0 - 10

02/04 12:59

IMP

ii) Alphabets: finite non Empty set of Symbol are called alphabets.

* It Should be unique and non duplicate are allowed.
* It can be represented by sigma (Σ)

Ex:- $\Sigma(a,b,c)$ – Alphabet

$\Sigma()$ – Not an alphabet

IMP

iii) String: finite Sequence of Symbol from alphabet are called String.

* Collection or Sequence of alphabet.

String $\omega \Rightarrow \Sigma\{a,b\}$.

ab, ba, aa, bb;

Create a language of length 3, $\Sigma(a,b)$

$\Sigma\Sigma\Sigma(a,b)(a,b)(a,b)$

Length, $L_3 = \{aa, bb, ab, ba\}\{a,b\}$

$L_3 = \Sigma x - \{aaa, bbb, aab, aba, abb, baa, bab, bba\}\{a,b\}$

$\Sigma(a,b)$

a, b, ab, ba, aa, bb

Length, $L_1 = \{a,a\}$

$L_2 = \{aa, bb, ab, ba\}$.

$\therefore 2^n = 2^2 = 4$

IMP

iv) Language: Collection String is called as language

8/9/2020

* language can be null string.

sigma* ⇒ Σ^* → Representation {which consist all the string}

Σ^* { ∈, a, b, aa, bb, ab, ba ... }

↓ universal set

$\Sigma^+ = \{\Sigma^* - \in\}$

Imp

1) Grammer :- It is a basic building blocks of TOC (Theory of computation)

* It can be Represented by

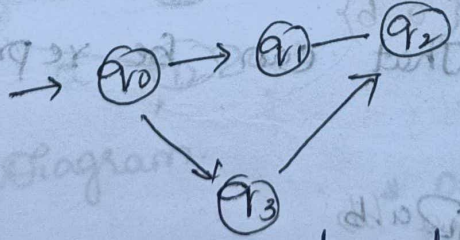
$G = \{V, T, P, S\}$

where :- G = Grammer, T = Terminal
V = variable, S = start
P = production

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

Language, $L = \{aa, bb\}$

ex :-



Two types of finite automata

1. Deterministic finite automata
2. Non-deterministic -11-

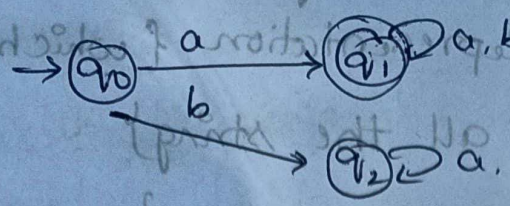
Deterministic finite automata :-

* we can determine and it is mathematically defined that

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

where :- Q = States / should be non empty / finite.

Ex :-

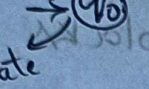


$Q = \{q_0, q_1, q_2\}$

* Any DFA we can draw without state.

2) q_0 = Initial State

* every Slm have only one { initial } state



3) Σ = finite State / non empty / Alphabet

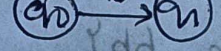
Ex :- $\Sigma = \{a, b\}$

4) F = Set of final state [any kind of set is may empty or non empty]

$0 \leq |F| \leq Q$

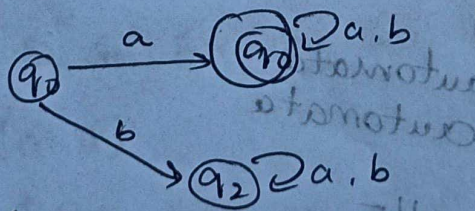
* set of final state can be represented by "0".

5) δ = The state transition.

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

* on any i/p symbol, it can be changing to one State to another That can be represented as δ.

Ex :- is



Solu'n :-

Transitional Table :-

	a	b
q_0	q_1	q_2
q_1 (self loop)	q_1	q_2
q_2	q_2	q_2

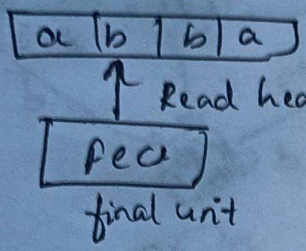
$\Sigma = \{a, b\}$.

If it is accepted by final stage

q_0 a q_1 b q_1 → String accepted

q_0 b q_2 b q_2 → String not accepted becoz the q_2 is not a final state.

* I/p String :-



Qthing Remember while drawing DFA.

① → q_0 Not more than one (atmost) one transition for i/p symbol

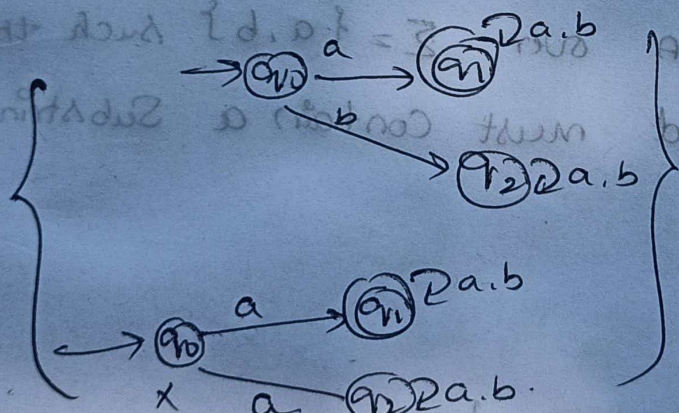
② DFA should be complete in nature

Imp

1) Design a minimal DFA (No. of State is minimum) (MDFA) over $\Sigma = \{a, b\}$ (i/p alphabet) such that String is accepted must start with w.

ⓐ w = 'a' ⓑ w = 'ba' ⓒ w = 'abb'.

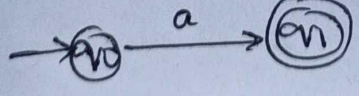
Language, L = { a, aa, ab, aab, ...}



→ Not started with b is not valid.

Not Required.

DFA Diagram.



Minimum String is a.

* only Staring state is matter then it can be anything

q_0 —a→ q_1.

final State.

* So String is accepted

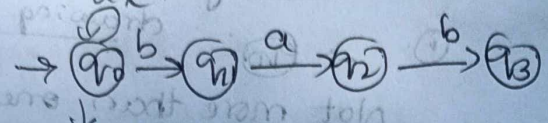
* In DFA we remember something so that we can rechanging.

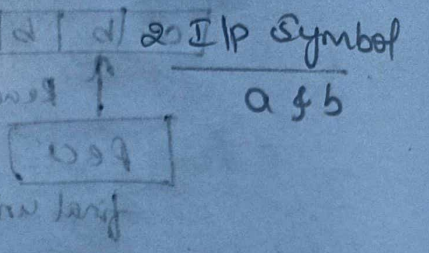
Ex:- 1) $L = \{w \in \{a,b\}^* \mid$ every a in w is immediately preceded & followed by b $\}$.

(After ba any b may came, it not matter)

$\Rightarrow$ { bab → String

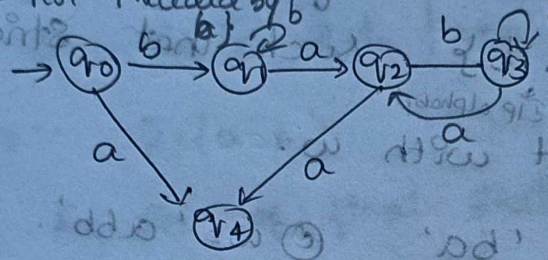
min. string: babb, bbbab, babbb, ----- }





{ Because we cannot make it a self loop, becoz it is not preceded by b }

Structure of DFA.



2) Substring

Ex:- Design a DFA over $\Sigma = \{a,b\}$ such that every string accepted must contain a substring w

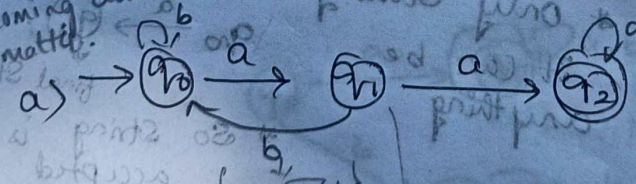
a) aa

b) ba

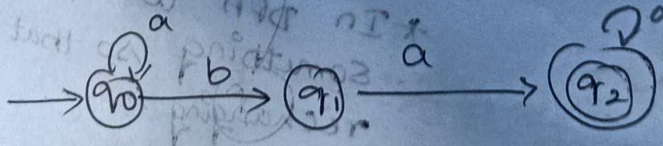
c) abb

$\Rightarrow$ now many a & b coming don't matter. $L = \{aa, baab, \ldots\}$

a)

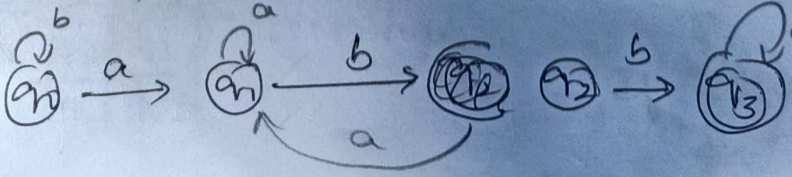


b)



bba
bbba

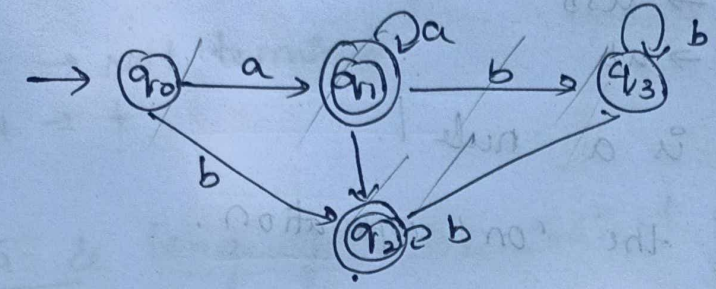
c)

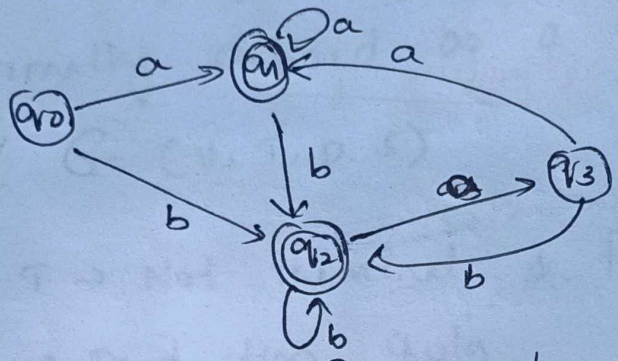


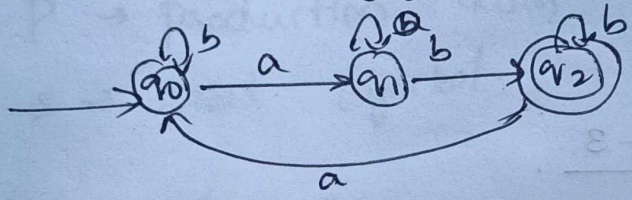
2/04 13:00

2> Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must start & ends with same symbol

=> $L = \{a, b, aaa, bbb\}$



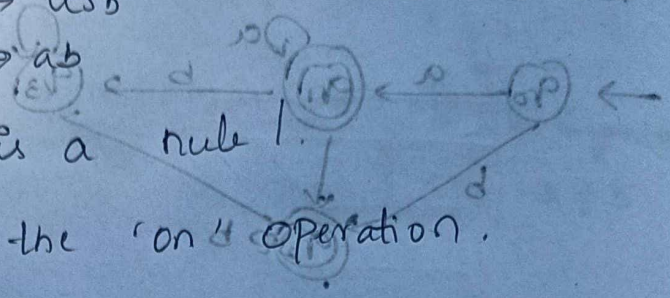




* Try to organize the language L for given CFG

$G = [\{S\}, \{a, b\}, P, \{S\}]$

When $P = \begin{cases} S \to asb \\ S \to ab \end{cases}$



$S \to asb \mid ab$ is a rule.

Which indicates the 'or' operation.

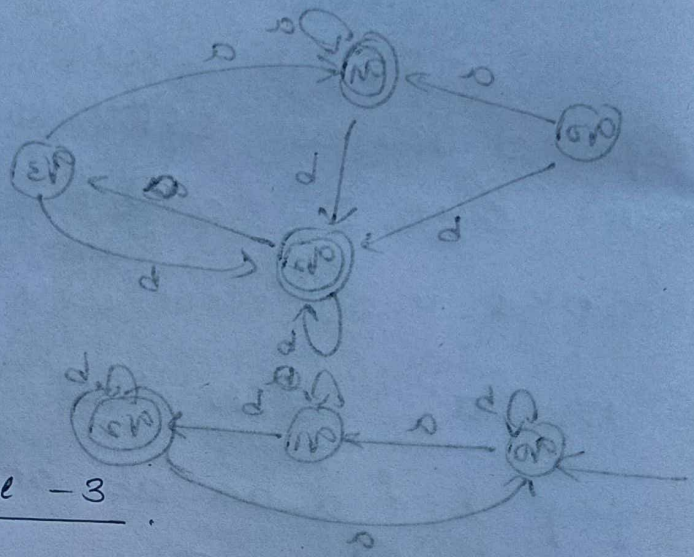
$S \to asb$

S

asb

$aasbb$

$aca\ sbbb$



Module - 3

IMP

FAQ (10)

Content of gramer.

1) Introduction to rewrite sim & gramer.
2) CFG & language.
3) Decegning CFG → 10M
4) Derivation and parse tree

IMP

10@5) Left & Right Most derivation

6) Ambiguity - 12M

* The rewrite system: It is a rule based sim in which there is a collection of rules and an algorithm form applying them.

Ex:- $S \to aS$
$S \to bA$
$A \to \varepsilon$ } production rules.

nt → Not terminal
nt → t

CFG & language: The context of free grammer can formally defined as a set denoted by

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

V & T → Not terminal & terminal.
P → Production rules
S → Start symbol.

nt → nt
nt → t

S is nt

$P = \{ S \to S+S$
$S \to S*S$
$S \to (S)$
$S \to 4 \}$

+, *, 4