Theory of Computation

Theory of computation is a branch that deals with how efficiently a problem can be solved in a model of computation using algorithm.

It will solve the problem in three method.

(i) Automata theory & language (FA, PDA)

(ii) Computability theory (TM)

(iii) Computational complexity theory (Reached Oriented)

Set Concepts

Similar types of variables collect in a single unit is called a set.

Eg. $S = \{1, 2, 3, 4, 5\}$, $S = \{A, B, C, D\}$

number, Alphabats

Empty Set $\{\}$ or ϕ no element.

Subset $\subseteq$ (one set element are there in the set)

Proper set $\subset$

Power Set (combination of all possible element)

Eg. $A = \{1, 2\}$

$P(A) = \{\phi, \{1\}, \{2\}, \{1, 2\}\}$

$P(A) = 4$

Set Operation

Cartesian Product - getting paired with all elements.

Eg $A = \{1, 2, 3\}$, $B = \{2, 3, 4\}$

$A \times B = \{(1,2), (1,3), (1,4), (2,2), (3,2), (2,3), (3,3), (1,4), (2,4), (3,4)\}$

Union Set - All element in the set.

$A \cup B = \{1, 2, 3\} \cup \{2, 3, 4\}$

$A \cup B = \{1, 2, 3, 4\}$

Intersection - $A \cap B = \{1, 2, 3\} \cap \{2, 3, 4\}$

$A \cap B = \{2, 3\}$

String (W) - It is the finite sequence of symbols from the alphabet (Σ) sigma

Eg. $W = \{01, 11, 00\}$ $\Sigma = \{0, 1\}$

ab, abab $W = 01011101$

Alphabet (Σ) = It is the finite, non empty set of symbols.

symbol = 0, 1, 2, 3 9

a, b, c, d . . . . . z

A, B, C, D . . . . . Z

@, #, ?, <, >, {, }, ! = Special symbol

+, −, /, %, *

Date- 04/06/25

Non-empty Symbol = $\{\Phi\}$

empty string = ε / ϵ (epsilon)

This symbol is used to represent empty set string.

Concatenation :- It is like the union set.

Eg. $u_1 = ab$

$u_2 = cd$

$u_1 . u_2 = abcd$

Length of a String :-

Eg. $w = abc$

$|w|$ = mod of $w = 3$

$w_1 = abbcd$

$|w_1|$ = mod of $w_1 = 5$

Structural Representation (Basic concept)

Language can be represent using
1. Grammer
2. Regular Expression

Grammer :-
It is mathematical model used to designing a ~~the~~ software.
(i) To identify the syntax of a language.
(ii) To identify the syntax of a statement K.
to check the expression is correct or not.

Regular Expression :-
It is used to represent the text strings.

Eg. Grammer → Verb, noun, adjective.

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

$V = \frac{\text{Variable / non-terminal}}{A, B, C, D \ldots Z}$

$T = \frac{\text{Terminal}}{a, b, c \ldots 0, 1, 2 \ldots ?, @, \# \ldots}$

$P = \frac{\text{Production}}{A \rightarrow \alpha}$
(A ← variable)

$S = \frac{\text{start symbol}}{A \rightarrow \alpha}$
(A ← start)

Regular Expression →

$L = \{0, 1, 01, 10, 101 \ldots\}$

$R.E = (0+1)^*$

Define FINITE Autometa ?

A finite autometa is a mathematical model for study of abstract machine or abstract computing device with discrete input & discrete output.

The abstract machine can solve the problem in the form of

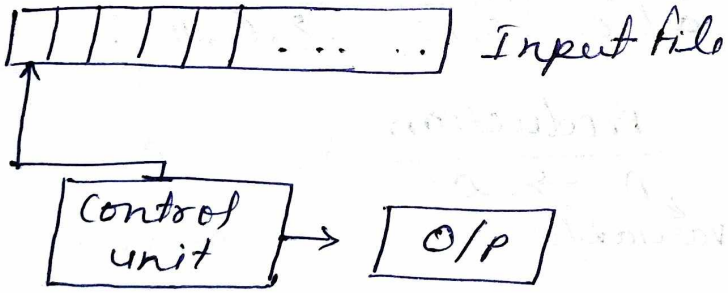
(i) Graphical (Transition , Diagram etc.)

(ii) Tabular (transition in table)

(iii) Mathematical (transition function & mapping function).

In other word we can say finite Autometa is an abstract computing device. It is a mathematical model of a study with discrete input, output, state and set of transition from state to state that occurs on input symbols from alphabets (Σ).

Component of Finite Autometa (F.A.)

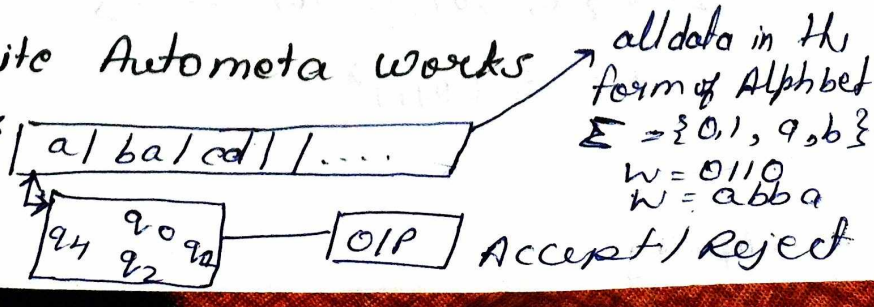


'Input file' - It contains cells where each cell contains a single I/P symbols

'Control unit' - Based on the I/P file. it determines the state of control unit.

'output unit' - accept / reject the result.

How does a Finite Autometa works



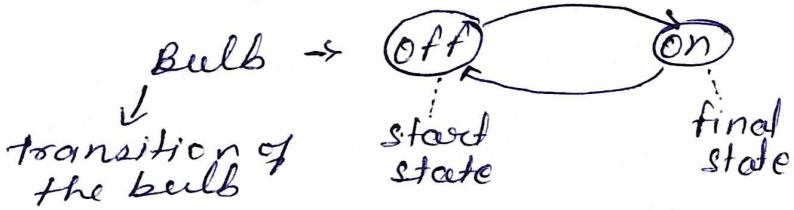
The FA solve the problem in ~~the~~ a three states
(1) Start state
(2) Non-final state
(3) ~~Finite state~~ Final State

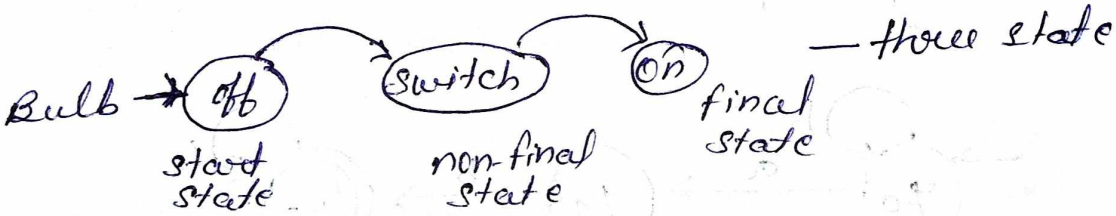
What are the different state / stages of the F.A.

There are three state / stages of the F.A.

1. Start state
2. non-final state
3. Final state

Eg. switching of Light.
It has two condition either ON/OFF





Notation of F.A.

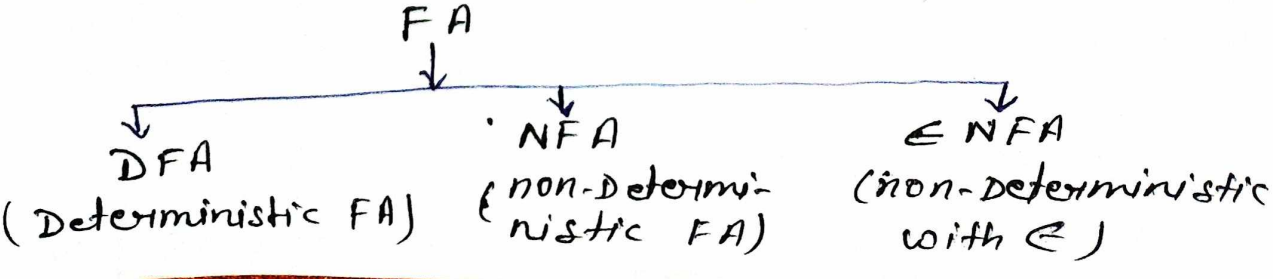
State – start state :- (q_0) (start state is always denoted by single circle)

Intermediate state :- (q_1), (q_2) (Normal state)

Final state :- (q_4) (final state is denoted by double circle)

Date - 05-06-2025

Finite autometa is of three types :-

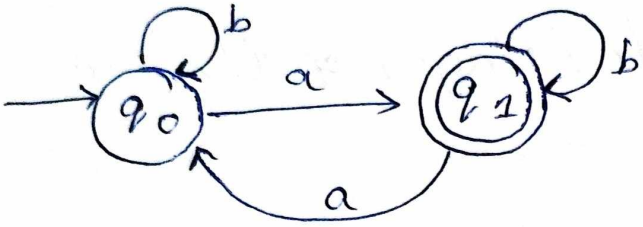


There are two formation of transition of FA.

(i) Transition table

(ii) Transition diagram.

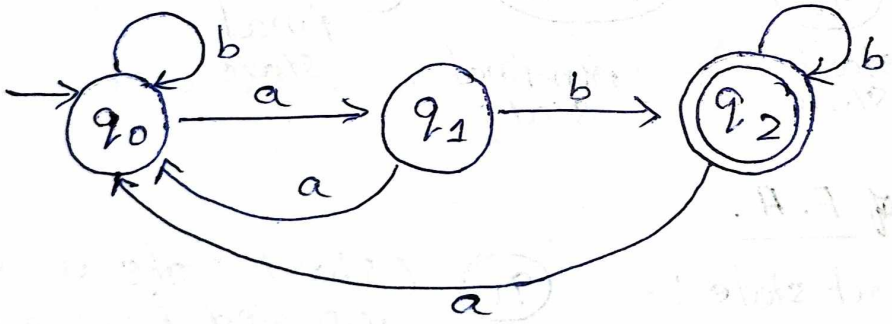
Eg. Construct the table of the following diagram.



Transition diagram I/P Symbols.

Input Symbols

δ	a	b
q_0	q_1	q_0
q_1	q_0	q_1



Transition diagram I/P symbols

I/P Symbols

δ	a	b
q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_0	q_2

Deterministic Finite Automata

The DFA is 5-tuples indicating five component.

$$M(Q, \Sigma, \partial, q_0, F)$$

M = Machine

Q = Finite Set of states

Σ = Finite set of input symbols/alphabets

∂ = transition function ($\partial : Q \times \Sigma \to Q$)

q_0 = start state

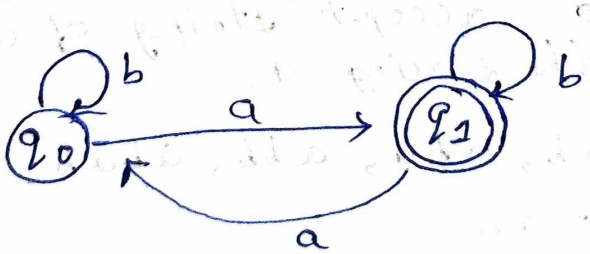
F = Set of final state

Define the DFA ?

The deterministic FA is exactly one transition for every input symbol from the state. So it is possible to determine exactly to which state the machine enters into after consuming the input symbol. (Next state).

The DFA has the finite and automata for calculation the finite means Machine has finite number of states and area and automation means a machine which may accept the string or reject the DFA has two notation to display the result. they can transition table and Transition diagram.

fig :-



Transition diagram.

Transition table

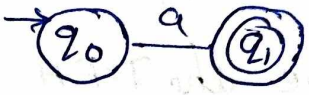
Input symbols

∂	a	b
q_0	q_1	q_0
q_1	q_0	q_1

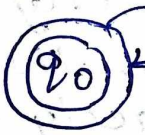
What do you understand by Acceptance of Language.

A Language acceptance is defined by " if a string W is accepted by the machine (M). i.e. if it is reaching the final state (F) by taking the string W" and if no accepted it will not reach the final state.

Eg. $L = \{\phi\}$ 

DFA to accept "a" ⇒ 

DFA to accept zero or more a

⇒ 

$L = \{\in, a, aa, aaa \ldots\}$

Date - 06-06-25

H.W

1. Draw a DFA to accept string of a's having at least one a (min string)
   (i) min string = a
   (ii) $\Sigma = \{a\}$

2. Draw DFA to accept string of a's and b's string with the string a.

   $L = \{ab, aba, abb, abab \ldots\}$

   (i) min string = ab
   (ii) $\Sigma = \{a, b\}$

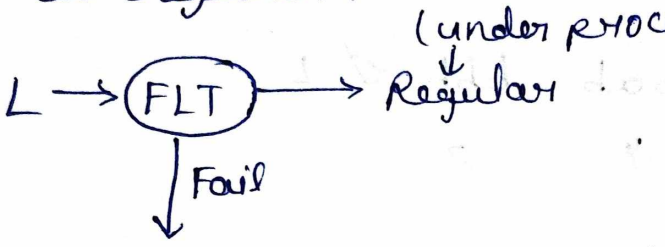
3 (a) Write the short notes on Transition Diagram (Transition graph)

(b) Transition table.

(c) Transition function

Define pumping Lemma and closure properties of regular language?

The language accepted by finite automata is known as regular language. Pumping lemma is used to prove that language is not regular. It can not be used to prove that a language is regular.



The term Pumping Lemma is made up of two words. i.e pumping and Lemma.

Pumping :- The word pumping refers to generating many input string by pushing a symbol in an input string repeatedly

Lemma :- The word Lemma refers to the intermediate theorem in a proof.

There are two pumping Lemma that are defined for -

- Regular Language
- Context - Free Language

Eg :- If L is a infinite language then there exists some positive integer 'n' (pumping length) such that any string $W \in L$ has length greater then equal to 'n'. i.e $|W| \geq n$, then W can be divided into three parts. $W = XYZ$ satisfy by following condition.

i) for each $i \geq 0$ $XY^iZ \in L$

ii) $|Y| > 0$ and (iii) $|XY| \leq n$

eg. $a^n b^n$ $\quad n \geq 0$ (infinite)

$\underset{x}{\underline{a\,a}}\ \underset{y}{\underline{bb}}\ \underset{z}{\underline{bb}} \in L$

$\underset{x}{aa}\ \underset{y}{bbbb}\ \underset{z}{bb} \notin L$

$\underset{x}{a}\ \underset{y}{abab}\ \underset{z}{bbb} \notin L$

Unit 3

Date - 21/07/2025

Context Free Languages

Context free grammar

Basic Concept

There are four types related to the context free grammer.

1. RG —uses→ RL —Recognize→ FA

   Regular Grammer — Regular Language — Finite Automata

2. CFG —uses→ CFL —Recognize→ PDA

   context free grammer — context free Language — Push Down Automata

3. CSG —uses→ CSL —Recognize→ LBA

   context Sensitive grammer — context sensitive Language — Linear Bounded Automata

4. REG —uses→ REL —Recognize→ TM

   Recursive Enogriable grammer — Recursive Enogriable Language — Turing Machine

Symbols

- Non-terminal Variable (V) — (A, B, A', A")
- Terminal Variable (T) — (a, b, 1, #, @)

Define Context Free grammar? with example.

context free grammer is used to specify the syntatic structure of programming language that consist of non-terminal, terminal, production and start symboles.

It is represent as 4 tupels :-

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

where,

G → stands for grammar
V → Finite set of Non-terminal symbols
T → Finite set of terminal symbol
S → start symbol
P → Finite set of production rules

$$\alpha \longrightarrow \beta, \alpha \in V$$

Eg :-

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

$S \rightarrow ABC$
$A \rightarrow aAb \mid ab$
$B \rightarrow \varepsilon B \mid \varepsilon$
$C \rightarrow d$

Ans :- $V = \{S, A, B, C\}$

$T = \{a, b, \varepsilon, d\}$

$S = \{S\}$

$P =$

$S \rightarrow ABC$
$A \rightarrow aAb$
$A \rightarrow ab$
$B \rightarrow \varepsilon B$
$B \rightarrow \varepsilon$
$C \rightarrow d$

H.W

Design Context free grammer for the following language.

(I) $L = a^n / n \geq 1$

(II) $L = a^n / n \geq 0$

(III) $L = a^n / n \geq 2$

Date - 22/07/2025

(i) Ans :- $L = a^n / n \geq 1$

$L = a, a^2, a^3, a^4, a^5 \ldots$

$T = \{ a, a^2, a^3, a^4, \ldots \}$

(III) $L = a^n / n \geq 2$

$n = 2$. aa
$n = 3$. aaa
$n = 4$ aaaa

~~$S \rightarrow aaS / aa$~~

$S \rightarrow aS / aa$

(ii) $L = a^n / n \geq 0$

$n = 0$. $\in$
$n = 1$. a
$n = 2$. aa

$S \rightarrow aS / \in$

(i) $L = a^n / n \geq 1$

$n = 1$. a
$n = 2$. aa
$a = 3$. aa

$S \rightarrow aS / a$

(iv) $L = a^n b^n / n \geq 1$

$n = 1$. ab
$n = 2$. ~~abab~~ aa bb
$n = 3$ ~~abab~~ aaa bbb

~~$S \Rightarrow abS / ab$~~

$S \rightarrow aSb / ab$

(a) $L = a^n b^n \mid n \geq 0$

$n = 0 \quad \in$

$n = 2 \quad ab$

$n = 2 \quad aabb$

$S \rightarrow aSb \mid \in$

(b) $L = a^n b^n \mid n \geq 2$

$n = 2 \quad aabb$

$n = 3 \quad aaabbb$

$n = 4 \quad aaaabbb$

$S \rightarrow asb \mid aabb$

(c) $L = a^n b^{n+1} \mid n \geq 1$

$n = 1 \quad abb$

$n = 2 \quad aabbb$

$n = 3 \quad aaabbbb$

$S \rightarrow aSb \mid abb$

(d) $L = a^{2n} b^n \mid n \geq 1$

$n = 1 \quad aab$

$n = 2 \quad aaaabb$

$n = 3 \quad aaaaaabbb$

$S \rightarrow aaSb \mid aab$

Q) Define Parse Tree or Derivation ? Explain with Example.

Ans :- The process of generating string from given grammer is called as derivation or parse tree.

Order of derivation

1. Left most derivation (LMD)

A derivation is said to be left most if in each steps the left most variable Non-terminal in a sentential from is replaced is called as LMD.

Date- 23/07/22

$L = a^{2n} b^n \mid n \geq 1$

$n = 1$ aab

$n = 2$ aaaabb

$n = 3$ aaaaaabbb

$S \longrightarrow aaSb \mid aab$

2. Right Most Derivation (RMD)

A derivation is said to be right most if in each steps the right most variable or non-terminal in a sentential form is replaced is called as RMD.

eg. $E \longrightarrow E+E \mid E*E \mid id$

Solve

string (W) = "id + id * id"

~~RMD~~

$E \rightarrow E+E$

$E \rightarrow E+id \quad \{E = id\}$

$E \rightarrow E*$

RMD

$E \longrightarrow E+E \mid E*E \mid id$

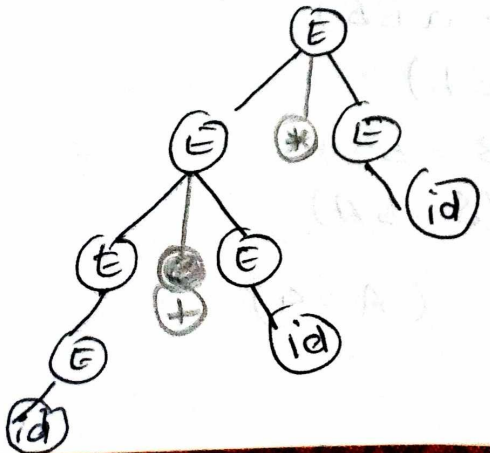
$E \rightarrow \cancel{E+E} \; E*E \quad \{E = id\}$

$E \rightarrow \cancel{E+E} \; E*id$

$E \rightarrow E+E*id \quad \{E = E+E\}$

$E \rightarrow E+id*id \quad \{E = id\}$

$E \rightarrow id+id*id \quad \{E = id\}$



LMD

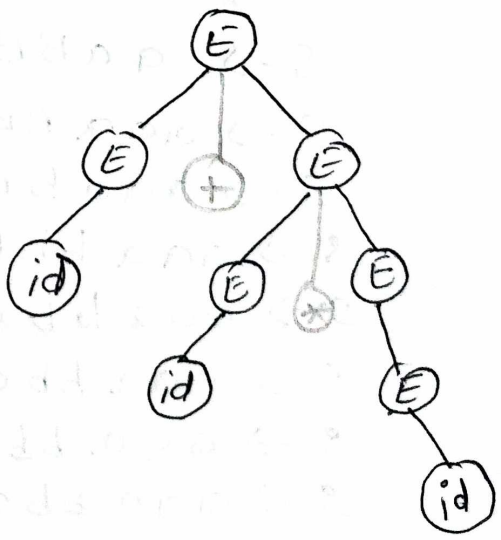
$E \rightarrow E+E$

$E \rightarrow id+E \quad \{E = id\}$

$E \rightarrow id+E*E \quad \{E = E*E\}$

$E \rightarrow id+id*E \quad \{E = id\}$

$E \rightarrow id+id*id \quad \{E = id\}$



Date - 29/07/25

Pushdown Automata (Deterministic and non-deterministic)

A pushdown automata (PDA) is a computational model that recognizes context free languages which are language generated by context free grammers.

Essentially a PDA is a finite state machine with the addition of a stack, allowing it to handle the unbounded memory required for recognizing these language.

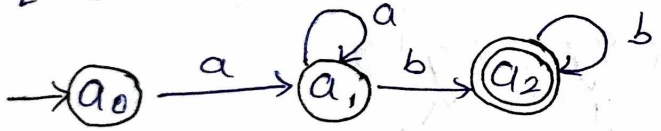
The limitation of finite automata is limited amount of memory hence it can not perform some mathametical operation. i.e comparison $(a^n b^n / n > 0)$

Hence to eliminate this drawback new types of automata, which consist of finite Automata and Memory elements (stacks).

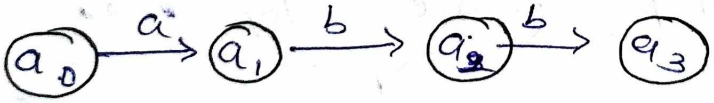
PDA = FA + stacks

Eg:- $L = a^n b^n / n > 0$

$L = \{ab, aabb, aaabbb, \ldots\ldots\ldots\}$



we take a string "abb"



Date - 30/07/2025

Mathematical defination

It is represented as 7 tupiles.

$M = (Q, \Sigma, \delta, q_0, F, Z_0, \gamma)$ $\quad \gamma = \Gamma a_0$

where,

Q = finite set of states

Σ = finite set of inputes alphabet.

q_0 = initial state

F = Finite set of Final state

Z_0 = Initial state Top

δ = Transition function

Γ = Stack Aphabets.

Representation of PDA (Pushdown Automata)

It will represent in two way :-

(i) Transition Function.

(ii) Transition Diagram.

1. Transition function :-

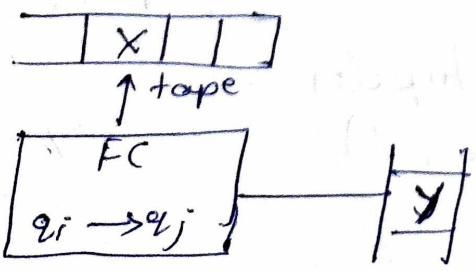
A transition function is a rule that dicte how a system changes from one state to another based on its current state and some input. The transition function is a fundamental component of finite automata, push down automata and other computational models.

General format

$$\delta(q_i, X, Y) = (q_j, x_y)$$

- q_i : State
- X : Tape Symbol
- Y : stack top
- q_j : next state
- x_y : stack top

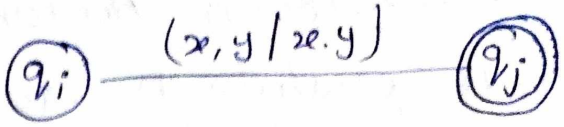
Before transition | After transition



Date - 31/07/2025

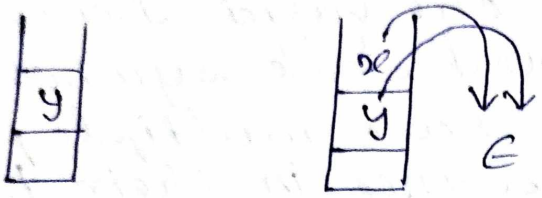
for Push

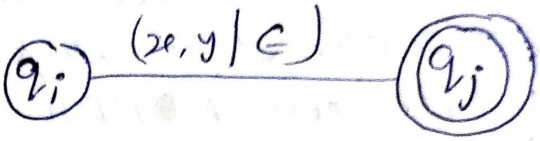
$\delta(q_i, x, y) = (q_j, xy)$



for Pop

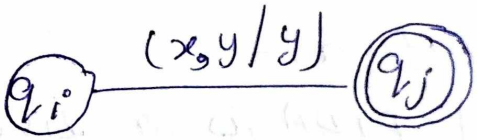
$\delta(q_i, x, y) = (q_j, \epsilon)$





for read

$\delta(q_i, x, y) = (q_j, y)$



Transition Diagram

A transition diagram, also known as a state transition diagram, is a graphical representation of a finite state machine (FSM). It depicts how a system or object change from one state to another based on inputs or events.

H.W

Q) Design a PDA for

$L = a^n b^n \mid n \geq 1$

1. Analysis
2. Algorithm (a) Push (b) Pop (c) Read

Date-04/08/2025

Pushdown Automata (Deterministic & non-deterministic)

A pushdown automata is a computational model that extends a finite automata with a stack. It can be either deterministic (DPDA) or non-deterministic (NDPDA). The key difference lies in the transition function is In a DPDA has at most one valid transition for each state input and stack symbol, while an NDPDA can have multiple possible transitions. This difference in their transition function leads to different capabilities and expression power.

Deterministic Pushdown Automata

Definition:-

A DPDA is a five tuples $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where,

Q - It is a finite set of state

Σ = It is a finite set of input symbol)

Γ = It is a finite set of stack symbol.

δ = It is a transition function (map a state, input and stack)

q_0 = It is the initial state

F = It is the set of accepting states.

Transition

for each state, input and stack symbol, there is at most one possible transition (either Pushing / Poping or remaining unchange)

Expression Power :-

DPDA recognize the class of deterministic context free Language (DCFLs), which is a a proper-subset of the context free language.

Date - 06/08/2025

Eg:-

Consider the language

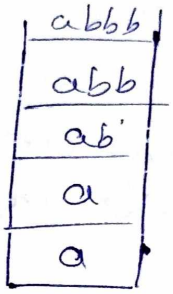
$L = (a^n b^n / n \geq 1)$

A DPDA can be designed to recognize this Language by pushing 'a' onto the stack and popping them when encountering 'b'.

$n = 1$ ab

$n = 2$ aabb

$n = 3$ aaabbb



aaabbb

Non-Deterministic Pushdown Automata (NDPA)

A non-deterministic pushdown Automata (NDPA) is a type of automata that extends the concept of a finite automata of in corporating a stack for memory. Unlike deterministic PDA, NPDAs can have multiple possible transition per a given state and input symbol, making them more powerful in recoagnizing language. An NPDA accept a string if at least one of its possible computation paths leads to an accepting state.

Stack -

An additional memory component with push and pop operation allowing the automata to store and retrieve data.

Formal Definition

An NPDAs is a formally defined as a 7-tuples.

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

Q = is a finite set of state

Σ = is a finite set of input symbol

Γ = is a finite set of stack

δ = is the transition function mapping (q, a, X) to set of (q', γ') where

→ q is the current state.

→ a is the input symbol or $\in$ (empty string)

→ X is the top of the stack

→ q' is the next state

→ γ' is the string of stack symbol to be pushed onto the stack.

q_0 = is the initial state

Z = is the initial stack symbol.

F = is the set of accepting states.

Date - 07/08/2025

Q. Construct a PDA for a language.

$L = \{ww^R \mid w = \{0,1\}^*\}$ even length palindrome

$w = 0110$

$w^R = 0110$

Sol^n = 010 | 010

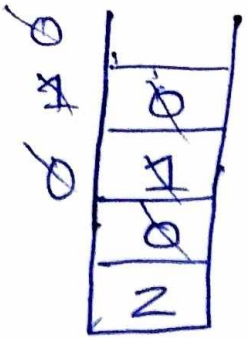
0110 | 0110

w w^R

case 1 :- If input symbol & stack symbol is same, then remove it.

010	010

we have to pop the values in the stack



$\in$

∴ the string is palindrome.

Date- 12/08/2025

Turing Machine

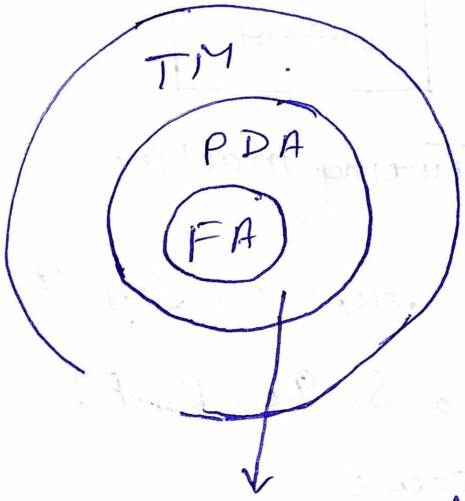
FA = Regular language

$R = \{abbabb\}$

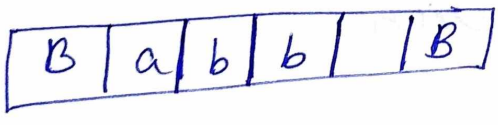
PDA = Context free language

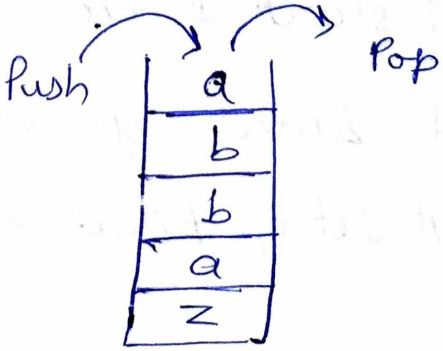
$R = \{a^n b^n\}$ if $n \leq 1$

$TM = a^n b^n c^n$



memory location
stack





Define Turing Machine

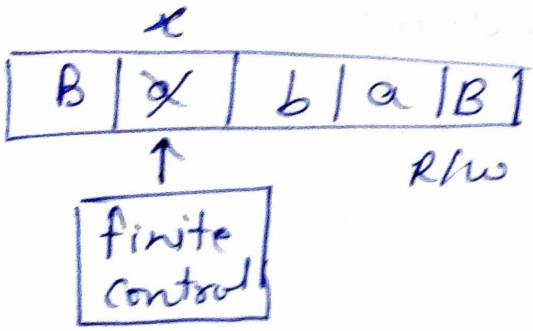
A turing machine is a theoritical model of computation process by Alan Turing in 1936. It is not a physical machine but a mathematical concept used to understand what problem can be computed and how. It is used to accept recorsive Enumerable language.

A turing machine is a mathematical model of computation describing and abstract machine that manupulates symbols on a strip of tape according to the table of rools. Despite the models simplicity, it is capable of implimenting any computer algorithm. A physical turing machine model constructed by mike Davey.

Special Symbol

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

B ↓ Blank space.



- Read, write allowed (update)
- Read, write, head can move left & right direction (with help of ε)
  (Vertical for one input variable only)

A turing machine consist of -

1. In finite tape - devided into cells, each holding a symbol (eg. 0, 1 or Blank)
   This act as memory.

Top head - can read and write symbols on the tape and move left or right on right at a time.

Finite control unit - contains a finite set of states ~~no~~ including a start state and possible ~~no~~ node more halting states.

ii) Transition function - A set of rules that tell the machine based on the current state and the symbol it ~~is~~ read.

Date - 14/08/2025

A turing machine consists of a tape of infinite length on which read and writes operations can be performed. The tape consist input symbols or a special symbol called blank. It also consist of a head pointer which point to cell currently being read and it can move in both direction.

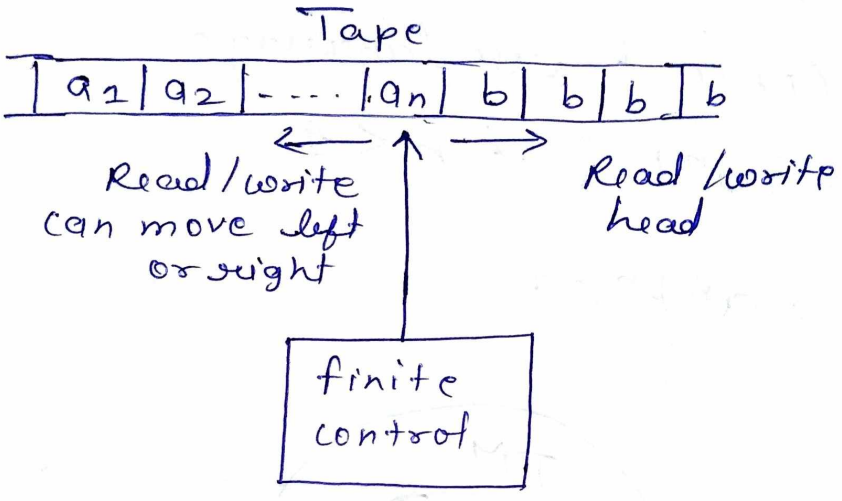


fig:- Turing machine

Turing machine is defined by 7-tuples :-

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

Q = A finite set of state

Σ = An input alphabet

r = A tape alphabet that includes Σ ($\Sigma \in r$)

δ = A transition function

q_0 = A start state

B = A blank symbol

F = A set of final state