Name : Sanket Chandrashekhar Harvande
Roll : 19
Sign :

DATE
PAGE NO. 1

Assignment No. 01

Q.1 Define
a) DFA
b) NFA
c) Moore Machine
d) Mealy Machine

a) DFA :

Definition :- DFA consists of finite set of state, one state is called start state and there can be one or more final states. In DFA from each state on each i/p symbol there is exactly one transition.

DFA is represented using five tuple representation

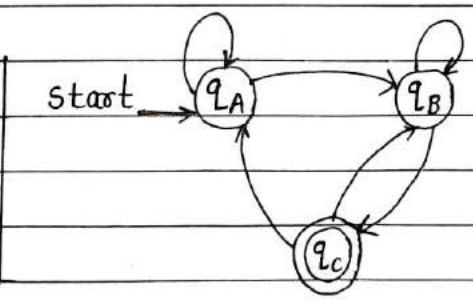
$$M = (Q, \Sigma, \delta, q_0, F)$$

where

Q = finite set of states
Σ = i/p alphabet
δ = transition function $\delta : Q \times \Sigma \longrightarrow Q$
q_0 = start state $q_0 \in Q$
F = Finite set of final states $F \subseteq Q$

eg :- $Q = \{q_A, q_B, q_C\}$
$\Sigma = \{0, 1\}$
$q_0 = q_A$

Q | E	0	1
$\rightarrow q_A$	q_A	q_B
q_B	q_C	q_B
q_C^*	q_A	q_B



Non-Deterministic Finite Automata (NFA):

Definition :- NFA consists of finite set of states. one state is called start state & there can be one or more final states

NFA is represented using five tupple representation and it is defined as follows.

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

Where,

Q = finite set of states

Σ = i/p alphabet

δ = transition function $\delta : Q \times \Sigma \rightarrow 2^Q$

q_0 = start state $q_0 \in Q$

F = finite set of final states $F \subseteq Q$

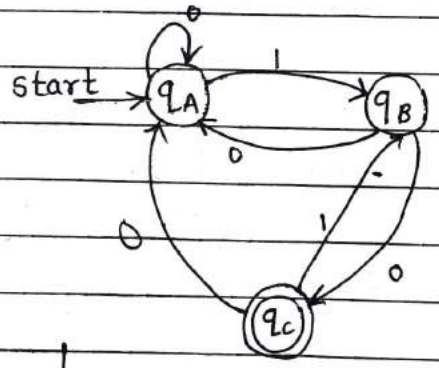
eg. $Q = \{q_A, q_B, q_C\}$

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

$q_0 = q_A$

$F = \{q_C\}$

δ :



$Q \setminus \Sigma$	0	1
$\rightarrow q_A$	q_A	q_B
q_B	$\{q_A, q_B\}$	$\{\}$
q_C^*	q_A	q_B

c) Moore Machine :-

Definition :-

It is a FA with no final state & it produces the o/p sequence for the given i/p sequence. In Moore m/c, the o/p symbol is associated with each state.

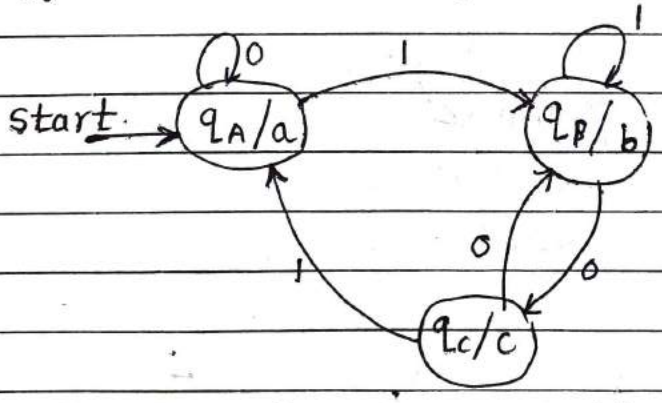
Moore machine is represented using six tupple representation & six tupple representation is given below

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

where,

Q = finite set of states
Σ = i/p alphabet
Δ = o/p alphabet
δ = transition function $\delta : Q \times \Sigma \to Q$
λ = o/p mapping $\lambda : Q \to \Delta$
q_0 = start state $q_0 \in Q$

eg.



$Q = \{q_A, q_B, q_C\}$ $\Delta = \{a, b\}$

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

$q_0 = q_A$.

$Q \backslash \Sigma$	0	1
$\to q_A$	q_A	q_B
q_B	q_C	q_B
q_C	q_B	q_A

$\lambda(q_A) = a$
$\lambda(q_B) = b$
$\lambda(q_C) = a$

d) Mealy Machine :-

It is FA with no final state & it produces the o/p sequence for the given i/p sequence.

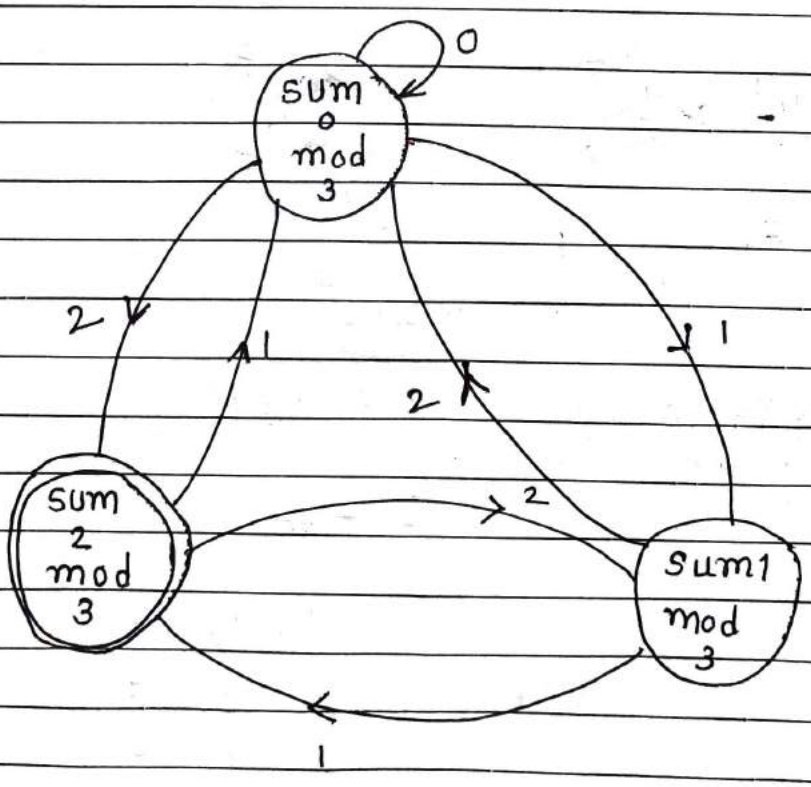
In mealy m/c, the o/p symbol is associated with each transition.

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

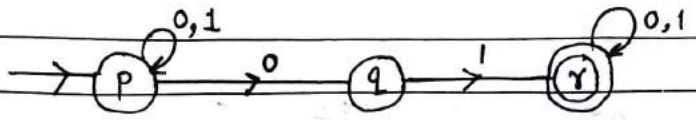
Where,

Q = finite set of states
Σ = i/p alphabet
Δ = o/p alphabet
δ = transition function $\quad \delta : Q \times \Sigma \to Q$
λ = o/p mapping $\quad \lambda : Q \times \Sigma \to \Delta$
q_0 = start state $\quad q_0 \in Q$

Q.2 Design FA for the language whose binary representation is x such that number is 2 mod 3.



Q.3 Convert NFA to DFA.

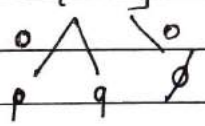


→ Intial state for DFA is p & we should always start with initial state.

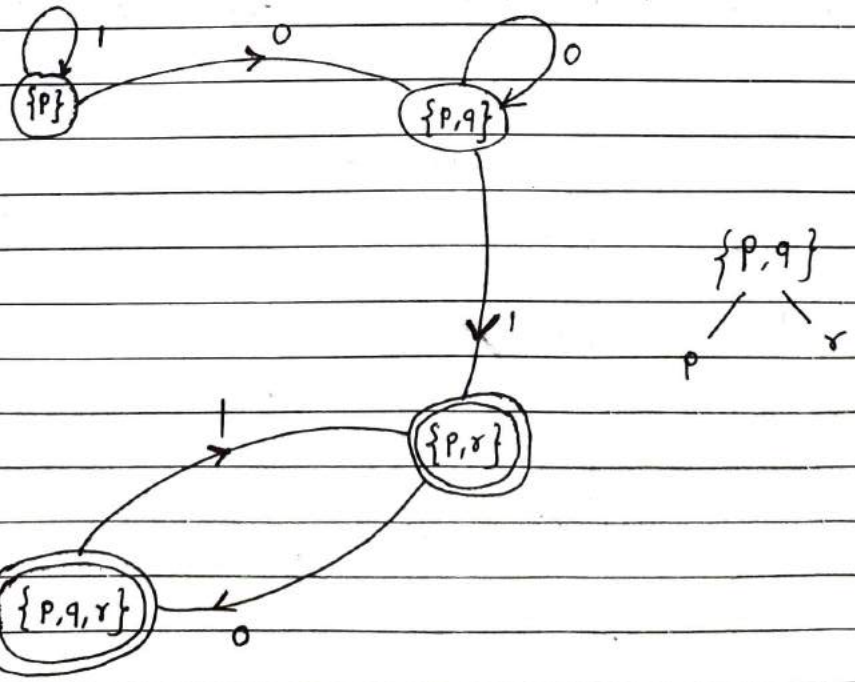
	0	1
→ {p}	{p,q}	{p}
{p,q}	{p,q}	{p,r}
{p,r}	{p,q,r}	{p,r}
{p,q,r}	{p,q,r}	{p,r}

So we found p another state which is reachable from p. so now consider that state p.

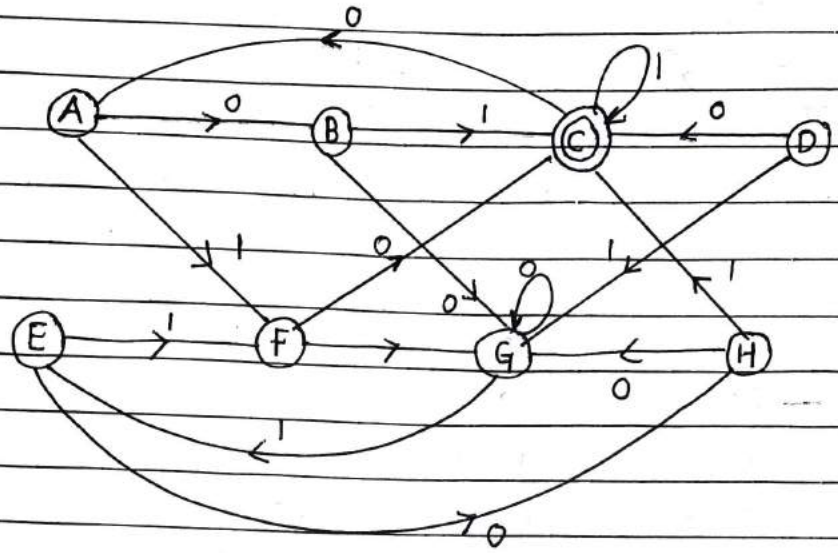
DFA is in {p,q}



Union = {p,q}



Q.4 Minimize the states of DFA.



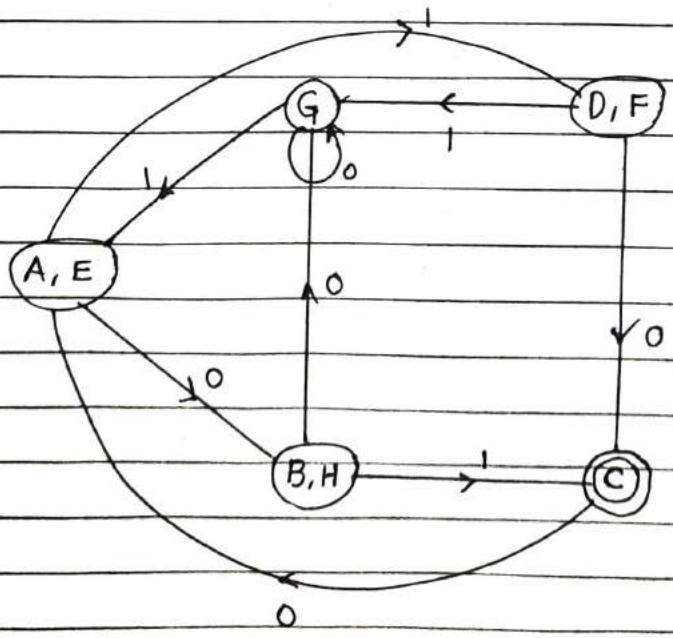
→

B	X						
C	X	X					
D	X	X	X				
E		X	X	X			
F	X	X	X		X		
G	X	X	X	X	X	X	
H	X		X	X	X	X	X
	A	B	C	D	E	F	G

→ Equivalent states (A, E)

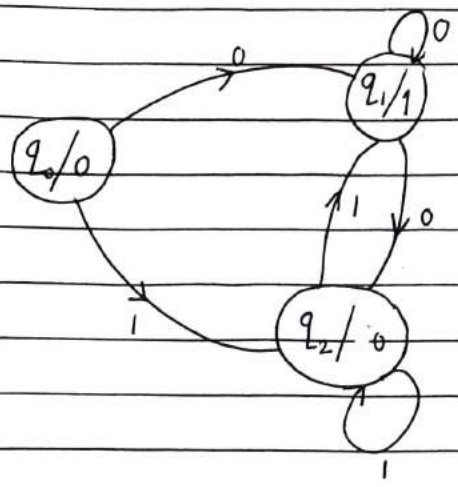
X - represents distinguishable states.

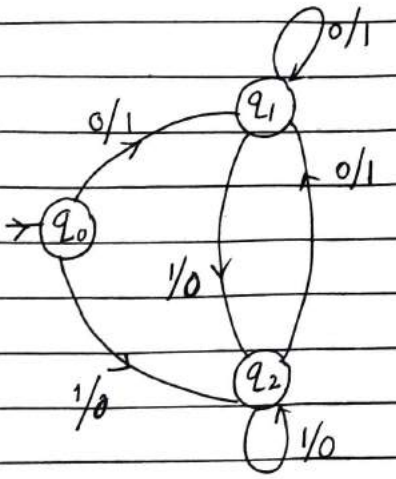
Q = {(A, E), (B, H), (D, F), C, G} ⇒ Equivalent states merge equivont state.



Q.5 Design a Moore & Mealy Machine to generate 1's complement of given binary number.

⇒





for instance

Take one binary number : 1011

Input :-

Input		1	0	1	1
state	q_0	q_2	q_1	q_2	q_2
output	0	0	1	0	0

Current state	Next	state	Output
$\rightarrow q_0$	q_1	q_2	0
q_1	q_1	q_2	1
q_2	q_1	q_2	0

Name : Sanket Chandrashekhar Harvande (19) :

DATE

PAGE NO. 01

ASSIGNMENT NO. 02

Q.1 Define
a) Regular Expression
b) Regular language

→ a) Regular Expression :-

R.E. is used for specifying the strings of regular language & is defined as
(1) 'ϕ' is R.E. for specifying {} (null set)
(2) 'ϵ' is R.E. for specifying {ϵ} (Epsilon)
(3) 'a' is R.E. for specifying {a}
(4) Let R & S be two R.E. for specifying LR & LS respectively
i) (R)/(S) is R.E. for specifying LR U LS
(ii) (R)(S) is R.E. for specifying LR·LS
(iii) (R)* is R.E. for specifying LR*

(b) Regular language :-

It is a language that can be expressed with a regular expression or a deterministic or non-deterministic finite Automata or state machine A language. A language is a set of strings which are made up of characters from a specified alphabet, or set of symbols. Regular languages are a subset of the set of all strings.

Closure properties of regular ~~expression~~ languages
The class of regular languages is closed under
i) Union ii) Intersection
iii) Complimentation (iv) set difference
v) Concatination
vi) Kleen Closure
vii) Reversal

Name: Sanket Chandrashekhar Harvande : 19 :

DATE

PAGE NO. 02

Q.2 Write short Note on:

a) Arden's Theorem :-

i) The Arden's theorem is used in regular longuages to determine whether a given expression has a unique solution

ii) It states that for two regular expressions out of which one does not contain e as its input, there exist a unique solution.

iii) Let P & Q be two regular expressions so if P doesn't have ϵ as its input, $R = Q + RP$ will have a unique solution represented by $R = QP^*$

Proof :-

We need to consider the fact that a regular expression r can be represented as $\epsilon + r^2 + r^3 + \dots$

So, we begin the proof using

$$R = Q + RP$$

$$R = Q + (Q + RP)P$$

$$= Q + QP + RP^2$$

Again replacing $R = Q + RP$ in RHS gives as

$$R = Q + QP + QP^2 + RP^3$$

$$R = Q(\epsilon + P + P^2 + \dots)$$

But $(\epsilon + P + P^2)$ can be replaced by P^*

Hence, $R = QP^*$ a unique solution hence exists.

Name: Sanket Chandrashekhar Harvande
Roll : 19
Sign :

DATE 03

b) Pumping lemma for RL :-

Statement :-

- Let L be a regular language & let z be a word of L such that $|z| \geq n$, where n is the minimum number of states required for recognizing L.
- Then as per pumping lemma, we can write z as $z = uvw$, where $|uv| \leq n$ & $1 \leq |v| \leq n$ such that all the strings of the form $uv^i w$ where $i \geq 0$ would belong to L.

Applications of Pumping lemma :-

1) Select a string z in the language L.
2) Breaks the string z into x, y & z accordance with the above conditions imposed by the pumping lemma.
3) Now, check if there is any contradiction to the pumping lemma for any value i.

c) Closu Properties of RL :-

i) Closure & Union :-

If L & m are regular languages so in LUM

proof :- Let L & M be the languages of regular expression R & S respectively

Then R+S is a regular expression whose language is LUM

ii) Concatenation :-

Let R & S be two regular expressions specifying regular languages L(R) & L(S) resp.
Now if we perform contatination operation on languages L(R) & L(S). Then resulting language after

the operation is performed will also be a regular language which be specified by a regular expression

R-5 is the regular expression for specifying L(R) . L(S)

iii) Intersection :-

Regular languages are closed under intersection i.e. if we perform intersection between two regular languages then the resulting language will be also a regular language

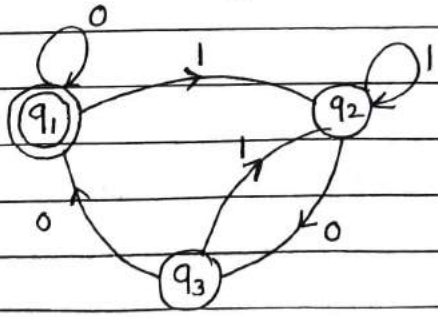
iv) Difference :-

Regular languages are closed under set difference operator i.e. if we perform set difference operator between / on the two regular languages then the resulting language will be also a regular language

v) Reversal :-

Regular languages are closed under reversal i.e. if we take a reversal of a regular language then the resulting language will be regular

Q.3 Convert the following DFA to RE



$\Rightarrow q_1 = 0q_1 + 1q_2 + E$

$q_2 = 0q_3 + 1q_2$

$q_3 = 0q_1 + 1q_2$

$q_2 = 0q_3 + 1q_2$

Name: Sanket Chandrashekhar Harvande
Roll : 19
Sign:

$= 1q_2 + 0q_3$

$q_2 = 1 * 0q_3$ .... using Arden's Theorem.

$q_3 = 0q_1 + 1q_2$
$= 0q_1 + 11 * q_3$
$= 11 * 0q_3 + 0q_1$
$= (11*0)*0q_1$ .... Using Arden's Theorem

$q_1 = 0q_1 + 1q_2 + \epsilon$
$= 0q_1 + 11 * 0q_3 + \epsilon$
$= 0q_1 + 11 * 0(11*0) * 0q_1 + \epsilon$
$= (0 + 11*0(11*0)*0)q_1 + \epsilon$
$= [(0 + 11*0(11*0)*0)] * \epsilon$

$\boxed{q_1 = (0+11*0(11*0)*0)^*}$

Q.4 Prove : $L = \{a^p \mid p \text{ is prime}\}$ is not regular

→ Let's assume L is a regular & p is a prime number
prime numbers - 2, 3, 5, 7 .....
- aa
aaa
aaaaa.....

Now consider $x = \underbrace{aaaaa}_{u\ v\ w}$, $|x|$ is prime say p

$i = 2$
$\underbrace{a}_{u}\underbrace{aa}_{v}\underbrace{a}_{v}\underbrace{aa}_{w}$

but p+1 is not a prime number. Hence Whatever we have assumed becomes contradictory. Hence L is not regular.

Name :- Sanket Chandrashekhar Harvande
Roll : 19
Sign :

DATE
PAGE NO. 6

Q.5 Show that $L = \{ww \mid w \in \{0,1\}^*\}$ is not regular.

→ Let us assume L is regular, consider he is pumping lemma, constant K represent length of w consider

$$x = \underbrace{0101}_{w} \underbrace{0101}_{w}$$
$$\underbrace{\quad}_{k} \quad \underbrace{\quad}_{k}$$

$$x = \underbrace{0101}_{u} \underbrace{0101}_{w}$$
$$\uparrow v$$

According to pumping lemma there exist uvw

$$\therefore |uv| \leq k \quad , |v| > 0$$

$i = 2$

$uvvw$

$$\underbrace{0110}_{k+1} \underbrace{101}_{k}$$

$\therefore uvvw \notin L$

∴ We have a contradiction therefore our assumption is wrong.

Hence it is not regular.

Name : Sanket Chandrashekhar Harvande
Roll : 19
Sign :

Page no.: 01
Date: / / /

Sub : Theory of Computer Science

Assignment No. : 03

Q.1 Define :-

a) Context free Grammer

⇒ A context free grammer G is a quadruple
$G = (V, \Sigma, P, S)$ where
V : set of non-terminal
Σ : set of terminal
P : set of rewriting rule or production rule.
S : The start symbol.

G is a context free grammer if each element of P is of the form
$X \rightarrow \alpha$ where $x \in V$ & $\alpha \in (V \cup \Sigma)^*$

b) Regular Grammer :-

The language accepted by finite automata be described using a set of production known as regular grammer. The production of regular grammer are of the following form.

$A \rightarrow a$
$A \rightarrow aB$
$A \rightarrow Ba$
$A \rightarrow e$

where $a \in T(\Sigma)$ & $A, B \in V$

A language generated by a regular grammer is known as regular language.

A regular grammer can be written in two forms
i) Right Linear form :-

A right linear form will have production of the given form

$A \to a$
$A \to aB$
$A \to \epsilon$

} Variable B in $A \to aB$ is the second symbol on the right.

ii) Left-linear form :-

A left linear form will have production of the following form

$A \to a$
$A \to Ba$
$A \to \epsilon$

} Variable B in $A \to Ba$ is the first symbol on the right.

c) Context free language :-

Context free grammer $G = (V, \Sigma, P, S)$ associated with language $L(G) \subseteq \Sigma^*$ as follows

$$L(G) = \{ x \in \Sigma^* \mid S \overset{*}{\Rightarrow} x \}$$

$L(G)$: Language generated by grammer G.
x represents set of all terminals strings which can be derived from start symbol of G.

A language is context free language if there is a context free grammer G such that $L = L(G)$

d) Chomsky Hierarchy of grammer :-

According to Noam Chomsky, there are four types of grammers & they are as follows :

(1) Type 0 → unrestricted grammer :-

Unrestricted grammer generates recursively enumerable language (REL). The productions have no restrictions hence it is termed as unrestricted grammer.

They generate the language that are recognized by a Turing Machine (TM.)

(2) Type 1 → Context sensitive Grammer (C S G)

CSG generates context sensitive languages (CSL). They generate the languages that are recognized by linear bounded automata. (LBA).

The production must be in the form

$$\alpha A \beta \rightarrow \alpha Y \beta$$

(3) Type 2 → Context free Grammer (CFG)

CFG generates context free language (CFL) they generate the languages that are recognized by push down automata (PDA).

(4) Type 3 → Regular Grammer :- (RG)

Regular Grammer generates regular language. They generate the language that are recognized by finite state machine. (FSM).

Q.2 Simplify the following grammer.

$S \rightarrow ASB \mid \epsilon$
$A \rightarrow aAS \mid a \mid aA$
$B \rightarrow sbS \mid bS \mid sb \mid b \mid aAS \mid a \mid aA \mid bb$

→

i) Remove ϵ production

$S \rightarrow ASB \mid AB$
$A \rightarrow aAS \mid a \mid aA$
$B \rightarrow sbS \mid bS \mid sb \mid b \mid A \mid bb$

ii) Remove unit production

$S \rightarrow ASB \mid AB$
$A \rightarrow aAS \mid a \mid aA$
$B \rightarrow sbS \mid bS \mid sb \mid b \mid aAS \mid a \mid aA \mid bb$

iii) Lemma 1 :-

$G' = (V', \Sigma, P', S)$
$V' = \{A, B, S\}$

iv) Lemma 2 :

$G'' = \{V'', \Sigma'', P'', S)$

$V'' = \{A, S, B\}$
$\Sigma'' = \{a, b, bb\}$

$P'' = S \rightarrow ASB \mid AB$
$A \rightarrow aAS \mid a \mid aA$
$B \rightarrow sbS \mid bS \mid sb \mid b \mid bb \mid aAS \mid aA \mid b$

Q.3 Convert CFG to CNF.

$S \rightarrow asb \mid ab$

→ for every terminal symbol introduce a new non-terminal

If $A \rightarrow B1B2 \ldots\ldots Bm$ is a rule CFG then the rule in CNF

$A \rightarrow B1D1$ $\qquad$ $D_{n-3} \rightarrow B_{n-2} D_{n-2}$

$D_1 \rightarrow B2D2$ $\qquad$ $D_{n-2} \rightarrow B_{n-1} B_n$

$D_2 \rightarrow B3D3$

CNF : step 1 :-

$S \rightarrow AsB$

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b$

step 2 :-

$S \rightarrow AD1$

$D1 \rightarrow SB$

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b$

Q.4 Convert CFG to GNF.

$S \rightarrow AA \mid 0$

$A \rightarrow SS \mid 1$

⟹ i) Not necessary

ii) Consider, $S = A1$, $A = A2$

$A1 \rightarrow A2A2 \mid 0$ ——①

$A2 \rightarrow A1A1 \mid 1$ ——②

iii) Rule 2 is not in GNF form so,

$A2 \rightarrow A2A2A1 \mid 0A1 \mid 1$ $\qquad$ $[A1 \rightarrow A2A2 \mid 0]$

Using Lemma 2

$B1 = 0A1$, $B2 = 1$ $\qquad$ $a_1 = A2A1$

$\therefore A2 = 0A1 \mid 1$

$A2 \rightarrow 0A1Z \mid 1Z$

$Z \rightarrow A2A1 \qquad A1 \rightarrow A2A2 \mid 0$

$Z \rightarrow A2A \mid Z$

iv) $A1 \rightarrow A2A2 \mid 0$

$A2 \rightarrow 0A1 \mid 1 \mid 0A1Z \mid 1Z$

$Z \rightarrow A2A1 \mid A2A1Z$

$A1 \rightarrow 0A1A2 \mid A2 \mid 0A1ZA2 \mid 1ZA2 \mid 0$

$A2 \rightarrow 0A1 \mid 1 \mid 0A1Z \mid 1Z$

$Z \rightarrow A2A1 \mid A2A1Z$

v) $A1 \rightarrow 0A1A2 \mid 1A2 \mid 0A1ZA2 \mid 1ZA2 \mid 0$

$A2 \rightarrow 0A1 \mid 1 \mid 0A1Z \mid 1Z$

$Z \rightarrow 0A1A1 \mid 1A1 \mid 0A1ZA1 \mid 1ZA1$

$Z \rightarrow 0A1A1Z \mid 1A1Z \mid 0A1ZA1Z \mid 1ZA1Z$

Now rewrite the rules by converting back $A1 = S$, $A2 = A$

$S \rightarrow 0SA \mid 1A \mid 0SZA \mid 1ZA \mid 0$

$A \rightarrow 0S \mid 1 \mid 0SZ \mid 1Z$

$Z \rightarrow 0SS \mid 1S \mid 0SZS \mid 1ZS$

$Z \rightarrow 0SSZ \mid 1SZ \mid 0SZSZ \mid 1ZSZ$

Q.5 Consider the grammer.

$S \rightarrow 0B \mid 1A$

$A \rightarrow 0 \mid 0S \mid 1AA$

$B \rightarrow 1 \mid 1S \mid 0BB$

For the string 00110101 find the following left most derivation, right most derivation & parse tree.

→ Leftmost derivation:-

$S \rightarrow 0B \qquad [S \rightarrow 0B]$

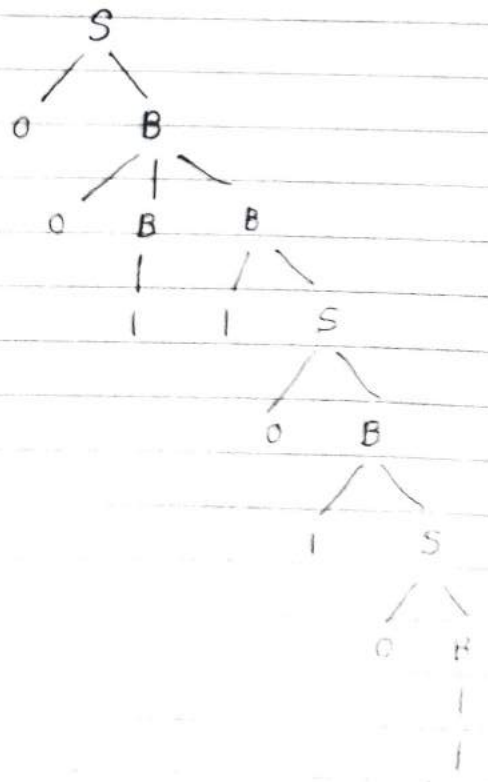
$\rightarrow 00BB \qquad [B \rightarrow 0BB]$

→ 001B [B → 1]
→ 0011S [B → 1S]
→ 00110B [S → 0B]
→ 001101S [B → 1S]
→ 0011010B [S → 0B]
→ 00110101 [B → 1]

Rightmost derivation :-

S → 0B [S → 0B]
→ 00BB [B → 0BB]
→ 00B1S [B → 1S]
→ 00B10S [S → 0B]
→ 00B101S [B → 1S]
→ 00B1010B [S → 0B]
→ 00B10101 [B → 1]
→ 00110101 [B → 1]

Parse tree



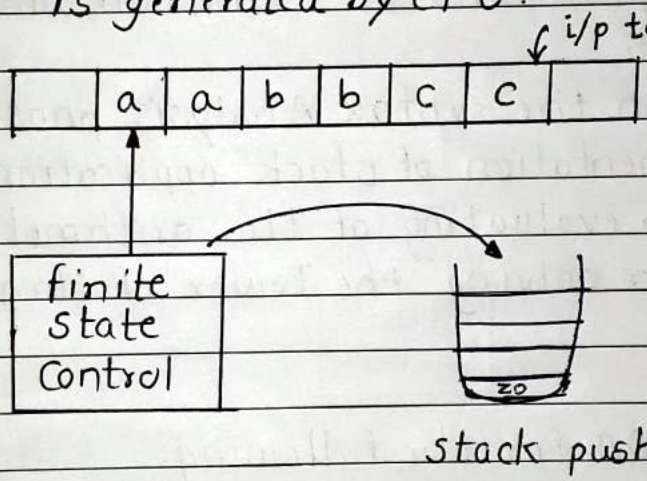
Name: Sanket Chandrashekhar Harvande
Roll : 19
Sign :

PAGE No. 01
DATE / /

TCS - Assignment No. 04

Q.1 Define PDA :-

→ Definition :- PDA is used for recognizing CFL which is generated by CFG.



stack push/down state

Components of PDA

PDA consists of finite set of states, input tape and read head and stack.

Working :-

PDA can be represented using 7 tuple representation and it is defined as follows

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

Where,

Q = finite set of states
Σ = i/p alphabet
Γ = Stack alphabet
δ = Transition Function

$\delta = Q \times \Sigma \cup \{\epsilon\} \times \Gamma \rightarrow$ subsets of $Q \times \Gamma^*$
q_0 = start state $q_0 \in Q$
Z_0 = Initial stack top symbol $Z_0 \in \Gamma$
F = Finite set of final states $F \subseteq Q$

Q.2 Write a short note on Applications of PDA.

→ The applications of Pushdown automata are as follows

- Used in the syntax Analysis phase.
- Implementation of stack applications
- Used in evaluating of the arithmetic expressions
- Used for solving the Tower of Hanoi Problem.

Q.3 Design PDA for the following.

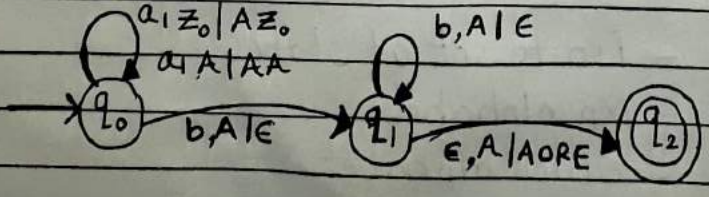
i) $L = \{a^m \; b^n \mid n < m\}$

ii) $L = \{a^n \; b^n \; c^m \; d^m \mid n, m >= 1\}$

iii) $L = \{a^n \; b^m \; a^n \mid n, m >= 1\}$

→

$L = \{a^m b^n \mid n < m\}$

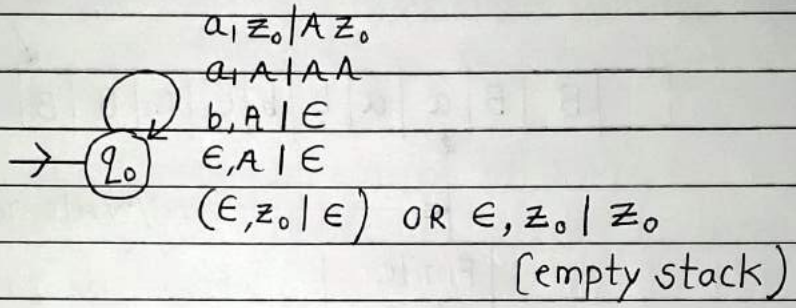


$\delta(q_0, a, Z_0) = (q_0, A Z_0)$ $\quad$ $\delta(q_0, b, A) = (q_1, \epsilon)$

$\delta(q_0, a, A) = (q_0, AA)$ $\quad$ $\delta(q_1, b, A) = (q_1, \epsilon)$

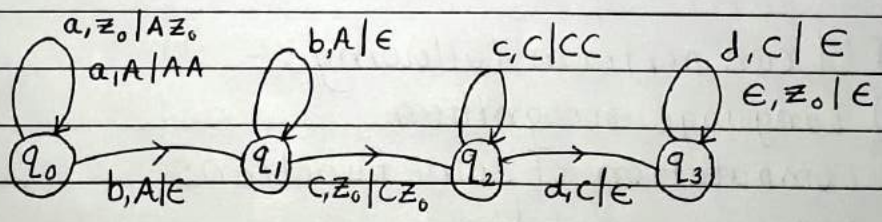
$\delta(q_1, \epsilon, A) = (q_2, A)$ OR (q_2, ϵ)

$M = (\{q_0, q_1, q_2\}, \{a, b\}, \{A, B, Z_0\}, \delta, q_0, Z_0, \{q_2\})$



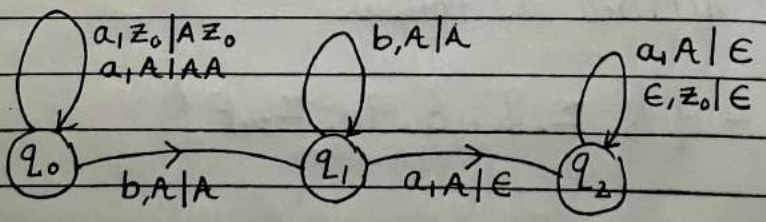
ii)

$L = \{a^n b^n c^m d^m \mid n, m \geq 1\}$



$M = (\{q_0, q_1, q_2, q_3\}, \{a, b, c, d\}, \{A, B, C, D\}, \delta, q_0, Z_0, \phi)$

iii) $L = \{a^n b^m a^n \mid m, n \geq 1\}$



$\therefore M = (\{q_0, q_1, q_2\}, \{a, b\}, \{A, Z_0\}, \delta, q_0, Z_0, \phi)$

Name : Sanket Chandrashekhar Harvande
Roll : 19
Sign : Sanket

PAGE No. 01
DATE / /

TCS - Assignment No. : 05

Q.1 Define TM

→ TM is a simple model of a computer and it is considered to be more powerful machine.

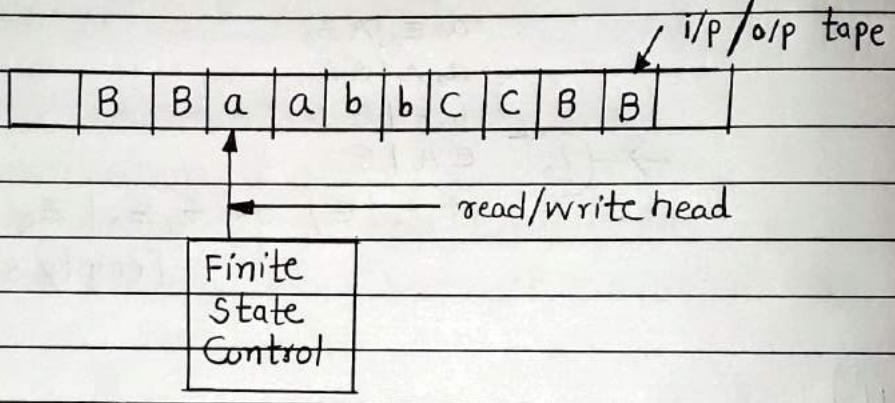


fig : Model of TM

TM can perform following s/-
1) Language recognition
2) Computation of some functions
3) Language generation

Representation of TM :-

- TM is represented using seven tupple representation & it is defined as follows

$$M = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$$

Where Q = Finite set of states
Σ = Input alphabet

PAGE No. 02
DATE / /

Γ = tape alphabet
δ = Transition function
q_0 = start state
B = Blank symbol
F = Finite set of final states.

- TM can change the state / remain in same state
- TM can change the tape symbol / keep it same.

Q.2 Write short note on variants of TM.

→ The different variants of Turing Machine (TM) are mentioned as follows :-

1. Turing Machine with Two-way Infinite Tape :-

In this variant of TM the input/output tape is a two way indefinite tape, i.e. there are unlimited blank cells on the left as well as right on the current non-blank portion on the tape.

2. Multi-tape Turing Machine :-

- In this variant of TM the machine has k tapes with k-heads i.e. each tape is controlled by seperate head.
- On single move depending on the state of the finite control & the symbol.

3. Multi-track turing Machine :-

- Multi-tracking turing machine is a specific type of multi tape turing machine.

- Multi-track turing Machines consists of multiple tracks but just one tape head which reads and writes K symbols from K tracks one by one.

4. Multi-Dimensional TM :

- Multi-dimensional turing machine has a multi-dimensional tape where head can move in any direction that is left, right, up & down.

5. Semi-Infinite Tape :-

- A TM with semi-infinite tape has no cells on the left hand side of the initial position & infinite cells on the right hand side of the initial position.

6. Non-deterministic Turing Machine :-

- Non-deterministic TM has a single one way infinite tape.
- In this variant of turing from each state on each tape symbol there can be multiple choices or paths hence it is cannot be deterministic.

PAGE No. 04
DATE / /

Q.3 Explain Applications, power, & limitations of TM.

→

Applications of TM :-

- For solving any recursively enumerable problem.
- For understanding complexity theory
- For implementation of neural networks
- For implementation of Robotics Applications
- For implementation of artificial intelligence.

Power of TM :-

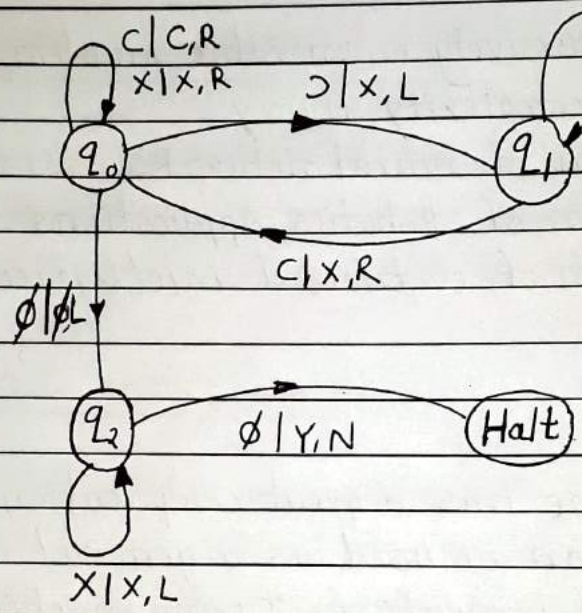
The turing machine has a great computational capabilities so it can be used as a general mathematical model for modern computers. Turing machine can model even recursively eumerable languages. Thus the advantage of turing machine is that it can model of all the computable functions as well as the languages for which the algorithm is possible.

Limitations of a Turing Machine :-

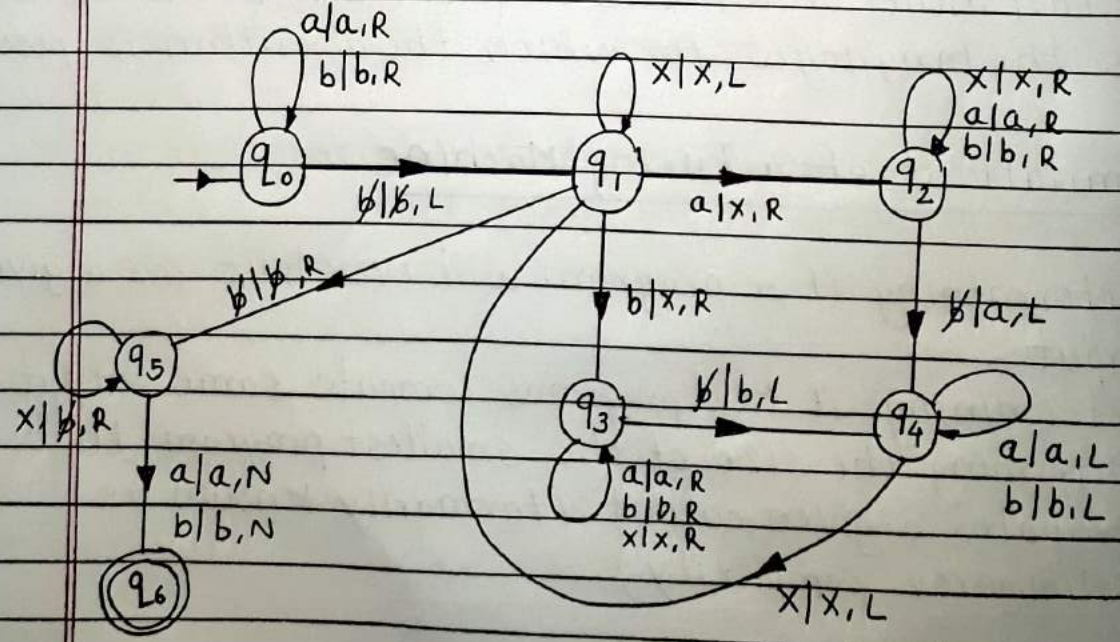
- Determining if a program will ever halt on a given input
- Determining if two programs compute same output.
- Determing the size of the smallest program that computes a given output (formarlly known as Kolmogorov complexity).

Q.4 Design TM for the following.

a. TM to check well formedness of paranthesis



b. TM to reserve string over {a, b}



Name: Sanket Chandrashekhar Harvande (19)

Sign:

Page no.: 01

Date: __/__/____

Assignment No. 06

Q.1 Write short Note on

a) Recursive & recursively enumerable language

⇒

* Recursive language :-

A language L is Turing decidable if the exist a halting Tm m such that

$$L = L(M)$$

for every I/p m/c has to halt.

M halts on w if either M accepted w or M reject w M is said to be a halting TM if $\forall w \in \Sigma^*$ M halts on w Decidable languages are subset of Turing recognizable.

* Recursively enumerable language :-

A language L is Turing recognizable, if there exists a TM M such that

$$L = L(M)$$

It means, TM for input

1) if $w \in L$, then Tm accept w & halt.
2) if $w \notin L$ then Tm either reject the w & halt

OR

it may goes into infinite loop.

∴ If $w \notin L$ there is no guarantee of halting the TM, language of these catagory called Recursively Enumerable languages

Recursively enumerable language for which it is not sure that on which input the TM will ever halt.

Name: Sanket Chandrashekhar Harvande (19)
Sign:

b) **Halting Problem :-**

i) The halting problem is a special kind of a problem wherein a Machine is proved to be undecidable in its behaviour at one point where its composition is changed.

ii) For this we assume a universal machine for this example & a couple of similar machines.

iii) We consider two simple machines, one (A) which performs addition of two numbers & another (B) that converts a number from decimal to binary

iv) Machine A cannot execute the output of Machine B neither can machine B execute machine A's output.

v) Now, we have a Machine D which executes any machine's (here A or B) output taking that Machine's blueprint and that Machine's individual input set as the input

vi) So, machine D will execute machine A's output on gaining Machine's A's blueprint & so it is the case for Machine B.

vii) Now, we construct a universal Machine V that has Machine D (to identify the Machine to be executed) & a negator machine N (to reserve the output).

viii) If we give machine B's blueprint & machine A's input set to V, we should be getting an error, which doesn't happen as the negator reserve the false output to true.

ix) If we give machine B's blueprint & machine B's input set to V, we should be getting the correct, which doesn't happen as the negator reserves the 'true' output to 'false'

x) This problem of undecidability is often what is termed as the Halting problem.

Name: Sanket Chandrashekhar Harvande

Sign :

Page no.: 03

Date: / /

c) Rice's Theorem :-

⇒ i) Rice's theorem helps to explain one aspect of the pervasiveness of undecidability. Here is the theorem & its proof, following the needed definition.

ii) A property of languages is a predicate $P : P(\Sigma^*) \Rightarrow \{false, true\}$ for some alphabet Σ. That is the input of P is a language and the output is a truth value.

iii) The value $P(L) = true$ means L has property P.

iv) The value $P(L) = False$ means L does not have property P.

v) Example properties are : is finite, is infinite, is r.e. etc.

vi) A non-trivial property of r.e. languages is a property of languages push that $P(L_D)$ for some r.e. language L_0 and $P(L_1)$ for some r.e. language L_1.

Proof :

We assume that ϕ does not have a property $P : P(\phi) = False$ we show that $A_{TM} \leq_{MLP}$. For this we must exhibit a Turing computable function for which $m' = f(M, \omega)$ is a machine accepting a language with property P if m accepts ω.

Let the behaviour of m' on input x to be :-

i) Run m on ω.

ii) If Rejects, reject

iii) Run m_1 on x, where M_1 is fixed machine for which $P((m_1)) = 1$. We know that such a M_i exists because P is a non-trivial solution of r.e. languages.

iv) If m accepts, accept; if m_1 rejects, reject.

Name : Sanket Chandrashekhar Harvande (19)

Sign : [signature]

Page no.: 04

Date: __/__/____

d) Post Correspondence Problem :-

⇒ i) The post correspondence problem (PCP) consists of two lists of strings over some alphabet Σ, the two lists must be of equal length.

ii) If there exists a solution to PCP, there exists infinitely many solutions

	LIST A	LIST B
i	w_i	x_i
1	110	110110
2	0011	00
3	0110	110

- A & B are defined above & let $\Sigma = \{0, 1\}$

- for instance let $m=3$, $i_1=2$, $i_2=3$ & $i_3=1$
- If solution exists to the above PCP, it should verify the following conditions i.e. $w_2, w_3, w_1 = x_2 x_3 x_1$
i.e. 00110110110 = 00110110110. since the above condition verified, therefore the solution is the list 2, 3, 1.
- It is not necessary this solution is unique, i.e. there can exists more than one solution.