Thadomal Shahani Engineering College

Bandra (W.), Mumbai- 400 050.

CERTIFICATE

Certify that Mr./Miss RAGHAV . VIKRAM RATHI

of Computer Department, Semester V with

Roll No. 7113208 has completed a course of the necessary

experiments in the subject TCS under my

supervision in the **Thadomal Shahani Engineering College**

Laboratory in the year 2023 - 2024

Teacher In-Charge

Head of the Department

Date 27 Oct 2023

Principal

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CONTENTS

1.1 (1)

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

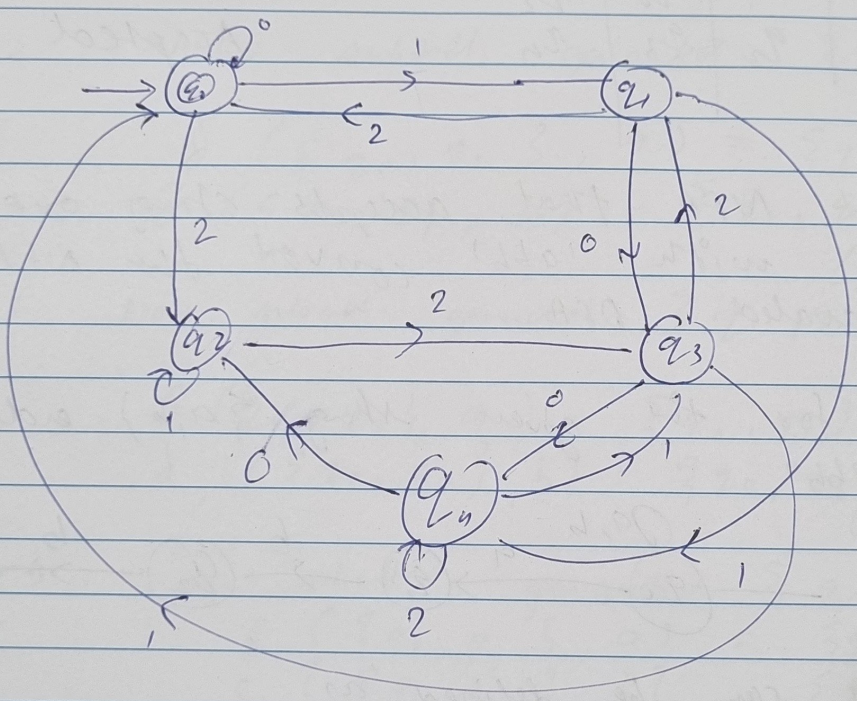
Kaylan 2113208

-: Assignment 1 :-

Q1). Design DFA to determine whether ternary number (base 3) is divisible by 5

→ Transition diagram

Ternary no $\Sigma = \{0, 1, 2\}$.



Above DFA can be represented as

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

Where $Q = \{q_0, q_1, q_2, q_3, q_4\}$

$\Sigma = \{0, 1, 2\}$

$q_0 = q_0$

$f = q_0$

Trantion table

Q \ Σ	0	1	2
q_0	q_0	q_1	q_2
q_1	q_3	q_4	q_0
q_2	q_1	q_2	q_3
q_3	q_4	q_0	q_2
q_4	q_2	q_3	q_4

Simulation $(202)_3$

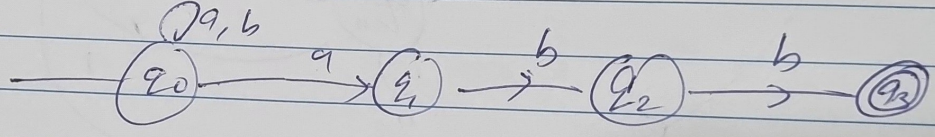
$\delta(q_0, 202) \vdash \delta(q_2,$
$\vdash \delta(q_1,$
$\vdash \delta[q_0)$
= final state

Accepted.

Q2) Construct NFA that accepts string over $\{a, b\}$ ending with 'abb' convert this NFA to equivalent DFA

Sol:

NFA for the given string $\{a, b\}$ ends with "abb"



NFA can be defined as

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

$Q = \{q_0, q_1, q_2, q_3\}$

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

$q_0 = q_0$

$F = \{q_3\}$

Trantion Table :

	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
q_2	ϕ	$\{q_3\}$
q_3	ϕ	ϕ

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

NFA to DFA

Step 1: Take $\{q_0\}$ as the initial state.

$\delta(q_0, a) = \{q_0, q_1\}$
$\delta(q_0, b) = \{q_0\}$.

Step 2: New subset generated $\{q_0, q_1\}$

$\delta(\{q_0, q_1\}, a) = \{q_0, q_1\}$
$\delta(\{q_0, q_1\}, b) = \{q_0, q_2\}$
↳ new state.

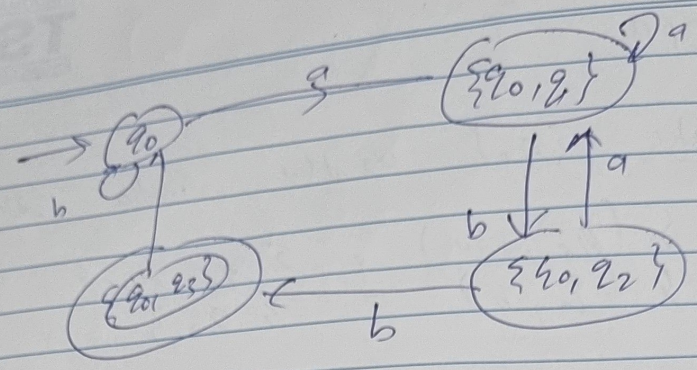
Step 3: New subset generated $\{q_0, q_2\}$.

$\delta(\{q_0, q_2\}, a) = \{q_0, q_1\}$
$\delta(\{q_0, q_2\}, b) = \{q_0, q_3\}$
↳ new state.

Step 4: new subset generated $\{q_0, q_3\}$
$\delta(\{q_0, q_3\}, a) = \{q_0, q_1\}$
$\delta(\{q_0, q_3\}, b) = \{q_0\}$.

No new state is generated
stop as new Partition table.

q \ Σ	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_3\}$



Above DFA can be represented as

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

$Q' = \{\{q_0\}, \{q_0, q_1\}, \{q_0, q_3\}, \{q_0, q_2\}\}$

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

$q_0 = q_0$

$F' = \{q_0, q_3\}$

Simulation: str : aabaabb.

$\delta'(q_0, aabaabb) \vdash \delta(\{q_0, q_1\}, abaabb)$

$\vdash \delta(\{q_0, q_1\}, baabb)$

$\vdash \delta(\{q_0, q_2\}, aabb)$

$\vdash \delta(\{q_0, q_3\}, abb)$

$\vdash \delta(\{q_0, q_3\}, bb)$

$\vdash \delta(\{q_0, q_3\}, b)$

= Final state.

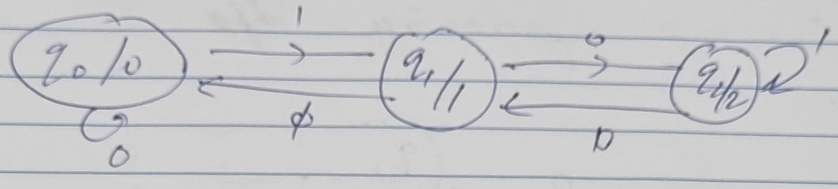
$\vdash \delta'(\{q_0, q_3\})$.

Accepted.

Q3) Construct more machine to find out module -3 for binary number.

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

o/p is 0, 1, 2, $\Delta = \{0, 1, 2\}$



Moore machine can be defined as :-

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

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

$q_0 = q_0$.

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

$\Delta = \{0, 1, 2\}$.

Transition table.

Current state	Next state		output
	0	1	
→ q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

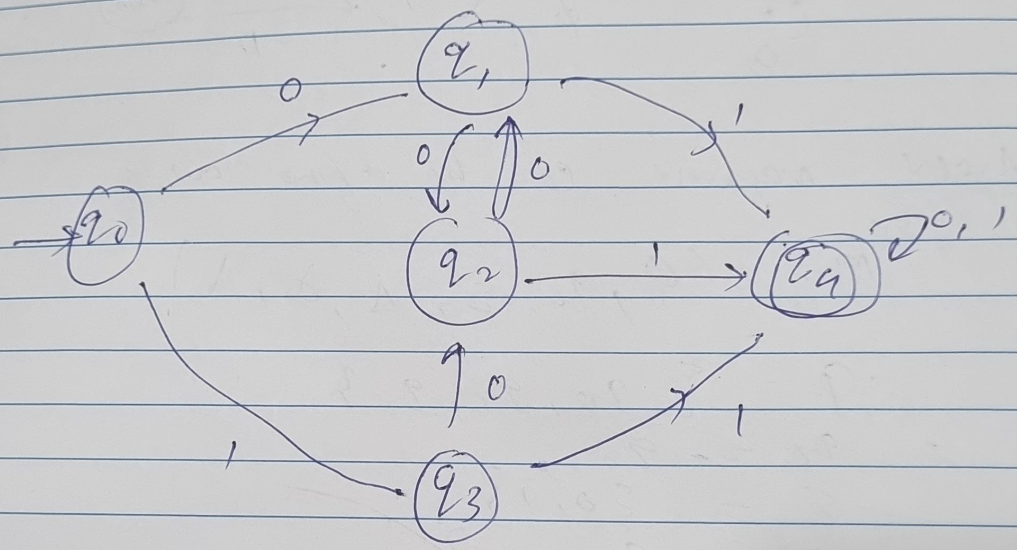
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

Simulation: 100 = (89)10

Inp	1	0	0	1	0
state	q_0	q_1	q_2	q_2	q_1
o/p	0	1	2	2	1

= 10%3 = 1

Q4.) Minimize following dfa

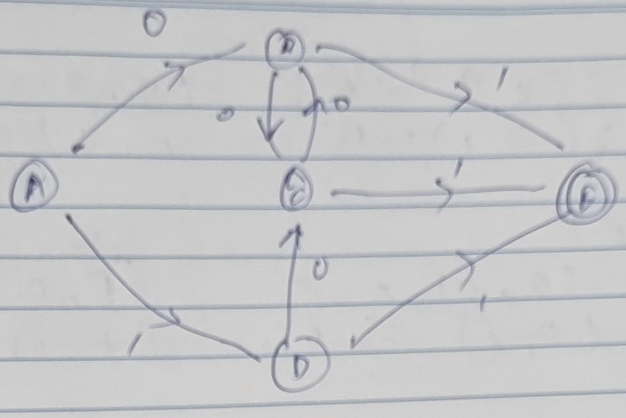


Step 1: Transition table

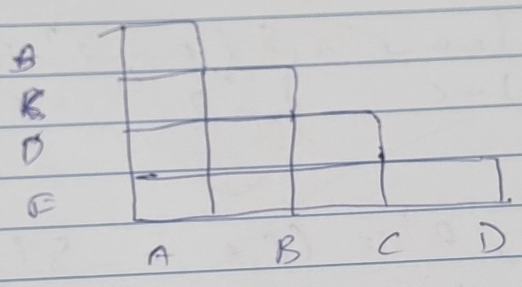
	0	1
→ q_0	q_1	q_3
q_1	q_2	q_4
q_2	q_1	q_4
q_3	q_2	q_4
q_4	q_4	q_4

Common $q_0 = A$, $q_1 = B$, $q_2 = C$, $q_3 = D$, $q_4 = E$

The diag. is:



Step 2:



More all final states are final
(B, E), (C, E), (D, E)

Step 3: Process all the states.

1) (A, E)

$\delta(A, 0) = B$ $\delta(E, 0) = D$

$\delta(A, 1) = E$ $\delta(E, 1) = E$

Not equivalent.

2) (B, E)

$\delta(B, 0) = C$ $\delta(E, 0) = E$

$\delta(B, 1) = E$ $\delta(E, 1) = E$ Not equivalent.

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

3) (C, E)
$\delta(C, 0) = B$ $\quad (B, 0) = E$
$\delta(C, 1) = E$ $\quad (E, 1) = E$

4) (D, E)
$(D, 0) = C$ $\quad (E, 0) = E$
$(D, 1) = E$ $\quad (E, 1) = E$

5) (A, D)
$\delta(A, 0) = B$ $\quad \delta(D, 0) = D$
$\delta(B, 1) = C$ $\quad \delta(D, 1) = E$

6) (B, D)
$\delta(B, 0) = (E, E)$ $\quad \delta(D, 0) = E$
$\delta(B, 1) = C$ $\quad \delta(D, 1) = E$
equivalent

7) (C, D)
$\delta(C, 0) = B$ $\quad \delta(D, 0) = F$
$\delta(C, 1) = C$ $\quad \delta(D, 1) = F$
equivalent.

8) (C, A)
$\delta(C, 0) = A$ $\quad \delta(A, 0) = B$
$\delta(C, 1) = B$ $\quad \delta(A, 1) = E$

9) (B, C)
$\delta(B, 0) = C$ $\quad \delta(C, 0) = E$
$\delta(B, 1) = B$ $\quad \delta(C, 1) = E$

(B)

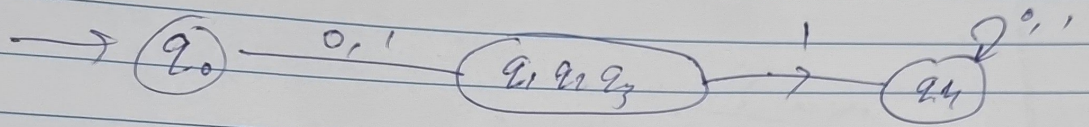
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

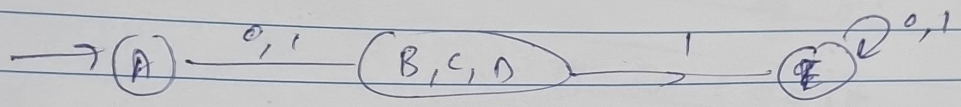
AB

$\delta(A, 0) = B$ $\quad \delta(B, 0) = E$

$\delta(A, 1) = C$ $\quad \delta(B, 1) = D$

Minimized DFA





(B)

2.1 (10)

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

21/3/2018

:: Assignment 2 :-

Q1) Give regular expression for

a) Set of all string over {0, 1} that end with '1' has no substring 00'

→ $\Sigma = \{0, 1\}$.
RE : $(0+\epsilon)(1+10)^* 1$

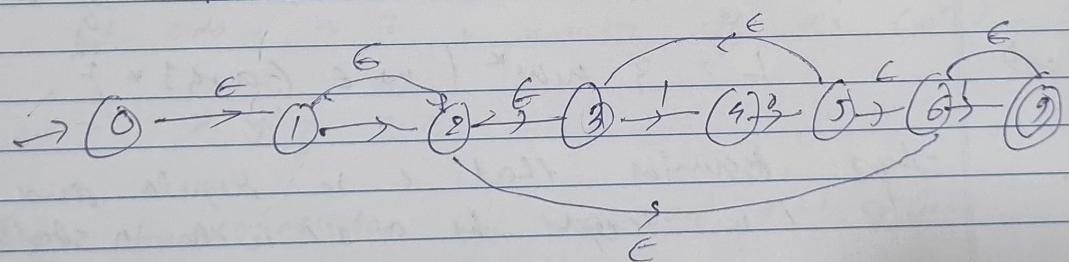
b. Set of all string over {0, 1} with even no of 1's followed by odd no. of 0's.

→ $\Sigma = \{0, 1\}$.
RE : $(11)^* 0 (00)^*$

Q2) Convert $(0+\epsilon)(10)^*(\epsilon+1)$ into NFA with ϵ-moves and hence obtain DFA.

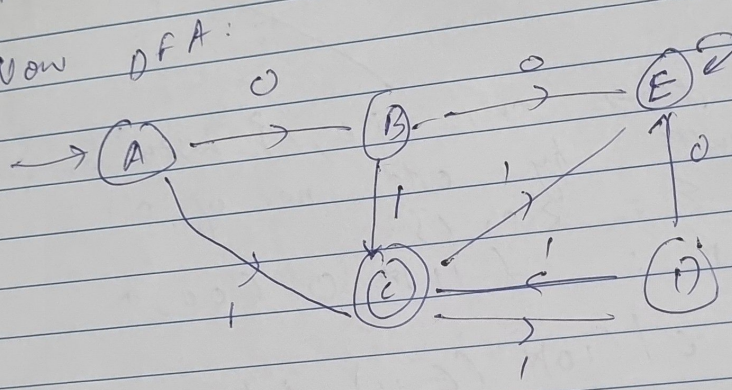
→ Soln,

RE to NFA with ϵ transition



State (x)		y ∈ closure (x)	δ(y,0)	δ(y,1)
→ {0}	A	0,1,2,3,6,7	{2}	{4,7}
{2}	B	2,3,6,7	{6}	{4,7}
* {4,9}	C	4,7	{5}	{ϕ}
{5}	D	5,3,6,7	{ϕ}	{9,9}
{ϕ}	E	ϕ	{0}	{ϕ}

Now DFA:



Q3) Prove the $\{wcw^R \mid w \in (a+b)^*\}$ is not regular where w^R is reverse of w.

→ $L = \{ww^R \mid w \in (a+b)^*\}$.

Step 1: Assuming that L is regular and L is accepted by a FA with n states.

Step 2: Choosing a string.

$w = a^n b \; ba^n$

$w \quad w^R$

$|w| = 2n + 2 \geq n$

$w = 2n + 2 \geq n$

w can be written as xyz with $|y| > 0$.

and $|xy| \leq n$

Since $|xy| \leq n$, x must be in the form a^s.

Since $|xy| \leq n$, y must be of the form $a^r | r > 0$

Therefore,

$w = a^n . bb . a^n = \underbrace{a^s}_{x} . \underbrace{a^r}_{y} . \underbrace{a^{n-s-r} bb a^n}_{z}$

Step 3: Checking whether xy^2z for $k=2$ belongs to L

$xy^2z = a^s a^{2r} a^{n-s-r} bb a^n = a^{n+r} bb a^n$

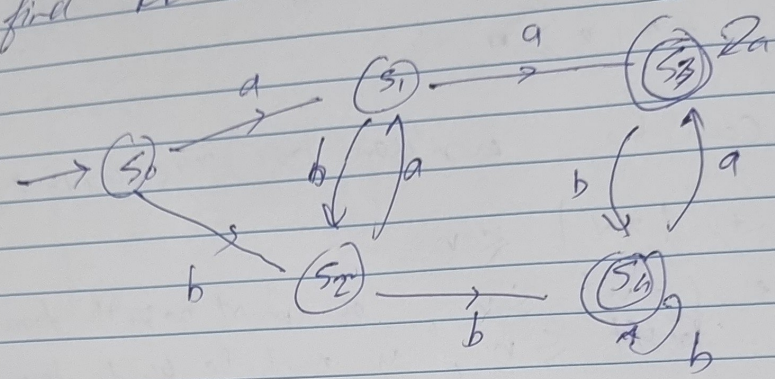
Since $r > 0$, $a^{n+r} bb a^n$ is not of the form ww^R because the string starts with a^{n+r} but ends with a number of a's (a^n).

$\therefore xy^2z \notin L$.

Therefore, by contradiction we can say that given language is not regular.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

q4) find RE for FA



FA can be described as.

$M = (Q, \Sigma, \delta_0, q_0, F)$.

where $Q = \{q_0, q_1, S_2, S_3, S_4\}$ (set of states)

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

$q_0 = S_0$.

$F = \{S_3, S_4\}$.

Transition fun. can be written as.

$\delta(S_0, a) = S_1$ $\quad$ $\delta(S_1, b) = S_2$

$\delta(S_1, a) = S_3$ $\quad$ $\delta(S_2, b) = S_2$

$\delta(S_2, a) = S_1$ $\quad$ $\delta(S_3, b) = S_4$

$\delta(S_3, a) = S_3$ $\quad$ $\delta(S_4, b) = S_4$

$\delta(S_4, a) = S_3$ $\quad$ $\delta(S_4, b) = S_4$.

Grammar G can be mapped as $G(V, T, S, P)$.

$V = \{S_0, S_1, S_2, S_3, S_4\}$.

$T = \{a, b\}$ $\quad$ $S = S_0$

2.5

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

and Production rules can be written as.

$S_0 \rightarrow aS_1$
$S_0 \rightarrow bS_2$
$S_1 \rightarrow a$
$S_1 \rightarrow bS_2$
$S_2 \rightarrow aS_1$
$S_2 \rightarrow b$
$S_3 \rightarrow a$
$S_3 \rightarrow b$
$S_4 \rightarrow a$
$S_4 \rightarrow b$

$S_0 \rightarrow aS_1 / bS_2$
$S_1 \rightarrow a / bS_2$
$S_2 \rightarrow aS_1 / b$
$S_3 \rightarrow a / b$
$S_4 \rightarrow a / b.$

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

27/10

:- Experiment 3 :-

Q1) Explain chomskey Hierarchy in detail.

→ - A grammer can be classified on the basis of productions rules
- chomskey classified grammers into the following types:

1. Type 3 : Regular grammer
2. Type 2 : Context free grammar
3. Type 1 : Context Sensitive grammer.
4. Type 0 : Unrestricted grammer.

Type 3 OR Regular grammer.

⇒ A grammer is called type 3 or regular grammer if all its production are of the following forms:

$A \rightarrow \epsilon$
$A \rightarrow a$
$A \rightarrow aB$
$A \rightarrow Ba$

where $a \in \Sigma$ and $A, B \in V$

A lang generated by Type 3 grammer is known as regular language.

Type 2 OR Context free Grammar

⇒ A grammar is called Type 2 or context free grammar if all its production are of follows form.

$A \rightarrow \alpha$

where, $A \in V$ and $\alpha \in (V \cup T)^*$

V is set of var, and T is set of terminals

→ The lang. generated by type 2 grammar is called as a context free lang. a regular lang. but not the reverse.

Type 1 OR context sensitive Grammar.

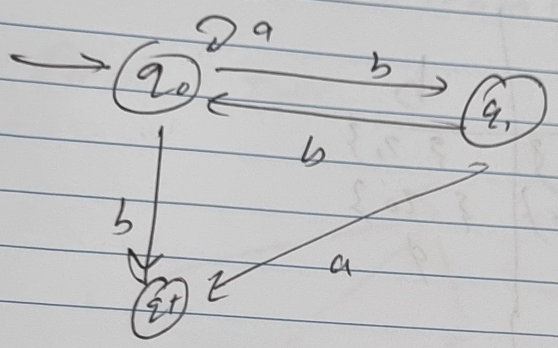
⇒ A grammar is called a Type 1 or context sensitive Grammar is called a Type 1 or context free of the following form.

$$\alpha \rightarrow \beta$$

where, β is atleast as long as α.

Q2) Construct finite automatic recognize $L(G)$ where G is grammer given by.

$S \rightarrow aS \mid bA \mid b$
$A \rightarrow aA \mid bS \mid a$



Grammer $(u) = (V, T, S, P)$.

$V \rightarrow \{A, \varepsilon\}$
$T \rightarrow \{a, b\}$
$S \rightarrow S$
$P \rightarrow S \rightarrow aS \mid bA \mid b$
$A \rightarrow aA \mid bS \mid a$

The above grammer is compared to finite automata defined as:

$M \rightarrow (Q, \Sigma, \delta, q_0, F)$
$Q \rightarrow \{q_0, q_1, q_f\}$
$\Sigma \rightarrow \{a, b\}$
$q_0 \rightarrow q_0$
$F \rightarrow \{q_f\}$.

where q_0 corresponds to 's' in above grammar
q_4 corresponds to 'A'

Transition Table :- (δ)

q	a	b
q_0	$\{q_0, q_4\}$	$\{q_1\}$
q_1	$\{q_1, q_4\}$	$\{q_0\}$
q_4	ϕ	ϕ

Q3) Consider the following grammar.
$S \rightarrow iCtS \mid iCtSeS \mid a$
$C \rightarrow b$.

→ For string 'ibttbtaea' find following

i) LMD
ii) RMD
iii) Parse Tree.
iv) Check if above grammar is ambiguous.

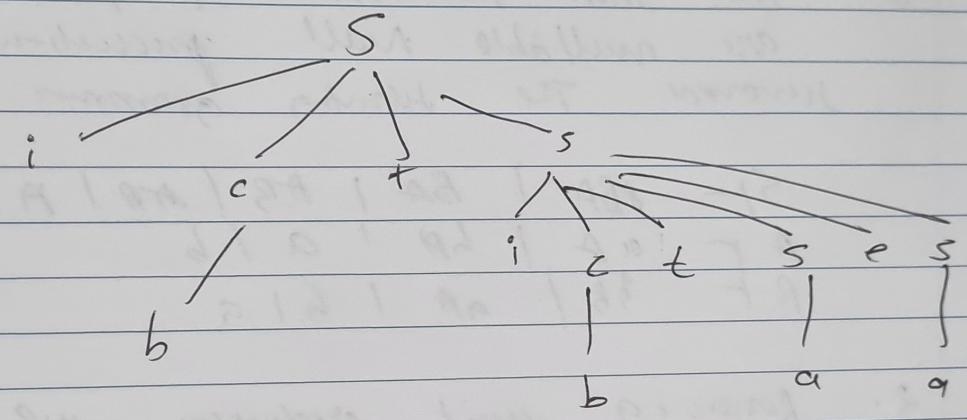
i) LMD.

$S \Rightarrow iCtS$
$S \Rightarrow ibtS$
$S \Rightarrow ibtiCtSeS$
$S \Rightarrow ibtibtaeS$
$S \Rightarrow ibtibtaea$.

ii) LMD.

⇒ S → ic tS
S → ic t ictses
S → ictictaeea.
S → ictebsaea
S → ibtebtaea.

iii) Parse Tree.



iv) The above grammer is ambigous grammer due to dangling if problem

The set of productions after above changes is.

S → ABA | BA | AB | AA | CaA | CbA
| a | b | CbB | CaA.
A → CaA | CbA | a | b
B → CbA | CaA | b | a
Ca → a
Cb → b

automata.
is PDA.
we then

Q1) Convert the following Grammar to CNF form

S → ABA | bA | ∈
A → aA | bA | ∈
B → bB | aA | ∈

⇒ Sol

1. The non-terminals {S, A, B} are nullable Null productions are removed The resulting grammar is.

S → ABA | BA | AB | AA | A | B.
A → aA | bA | a | b
B → bB | aA | b | a

2. Removing unit production we get

S → ABA | BA | AB | AA | aA | bA | a | b | bB | aA
A → aA | bA | a | b
B → bB | aA | b | a

3. Every symbol in α, in product of the form $A \to \alpha$ where $|\alpha| \geq 2$ should be a variable. This can be done by adding two productions

$C_a \to a$

$C_b \to b$.

DPDA.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

27/10

:- Experiment 4 :-

Q1) Explain PDA & NPDA with help of example

Ans :) **Push Down Automata :-** PDA in a way to implement a context free grammer (CFG) in a similar way to DFA (Deterministic finite Automata) for a regular grammer.

A DFA can remember a finite amount of information but a PDA can remember an infinite. Basically, a PDA is finite state machine + a stack.

PDA has three components.

A i/p tape.
A control unit.
A stack with infinite size.
A PDA is further divided into DPDA & NPDA.

i) DPDA (Deterministic Pushdown Automata) :-

In a DPDA, for each combination of current state & input symbol, there is at most one possible action that the automata can take. This means the DPDA.

ii) NPDA (Non-Deterministic Pushdown Automata)
In an NPDA, there can be multiple possible actions for the same input symbol & stack symbol. This means that the automata can choose one of several transitions to make based on the same i/p symbol & stack symbol. This flexibility in the recognition of lang. but has required exploration of multi paths during computation

Definition of DPDA & NPDA with example.
Let $M = (Q, \Sigma, N, \Sigma_0, Z, F, \delta)$ be a PDA The PDA is deterministic if & only if
i) $\delta(q, a, Z_0)$ has one element.
ii) If $\delta(q, \epsilon, Z_0)$ is non-empty $\epsilon \in \Sigma$ both conditions should be satisfied for the PDA the PDA is non deterministic.

Eg:
Is the PDA corresponding to lang. $L = \{a^n b^n / n >= 1\}$ by the finite state is deterministic?

Soln:
$\delta(q_0, q, z_0) = \delta(q_0, qzq)$
$\delta(q_0, q, q) = \delta(q_0, qq)$
$\delta(q_0, b, q) = \delta(q_1, \epsilon)$
$\delta(q_1, b, q) = \delta(q_1, \epsilon)$
$\delta(q_1, \epsilon, z_0) = \delta(q_f, \epsilon)$

The PDA should satisfy the two condition.

i) $\delta(q_0, q, z_0)$ has only one elmat, in this case for each $q \in Q$, $a \in \Sigma$, & $z \in N$, there exists only one definition. so the first condition is satisfied.

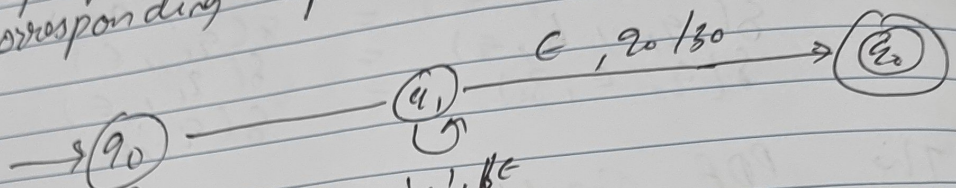
ii) To satisfy the second condition is satisfied the transition $\delta(q_1, \epsilon, z_0) = (q_f, \epsilon)$ since the transition is defined the Trasition $\delta(q, q, z_0)$ where $a \in \Sigma$ should not be defined which is the PDA is deterministic.

Q2) Construct PDA accepting language of palidrome $L = \{ ww^r / wcw^r, w = \{0,1\}^* \}$ where w^r is reverse of w. this is the lang. of all palindrome both odd & even over alphabet 0, 1.

Logic:

We keep pushing the first part of the string to the stak & then as we move to the next part, we then

keep poping as we find out the corresponding pairs



1, 0/10
0, 0/00
1, 1/11
0, 1/01
1, z_0/1β_0
0, β_0/0β_0

1, 1, 1/ϵ
0, 0/ϵ

PDA can be defined as.

$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, \beta_0\}, q_0, \beta_0, \{q_2\}, \delta)$

Transition function.

$(q_0, 0, 0) = (q_0, 00)$
$(q_0, 1, 0) = (q_0, 10)$
$(q_0, 1, 1) = (q_0, 11)$
$(q_0, 0, 1) = (q_0, 01)$
$(q_0, 1, \beta_0) = (q_0, 1\beta_0)$
$(q_0, 0, \beta_0) = (q_0, 0\beta_0)$
$(q_0, G, \epsilon) = (q_1, \epsilon)$

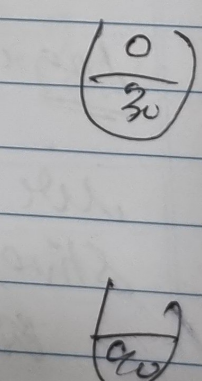
$(q_0, c, \epsilon) = (q_1, \epsilon)$
$(q_1, 1, 1) = (q_1, \epsilon)$
$(q_1, 0, 0) = (q_1, \epsilon)$
$(q_1, s, \beta_0) = (q_2, \beta_0)$

Simulation:

$q_0(0101\Delta) \vdash 0101\Delta$

$\vdash 0101\Delta$

$\vdash 0101\Delta$



$\vdash$ 0 1 0 Δ

q_0

Accepted.

$q_0(011)$

$\vdash$ 0 1 1 Δ

q_0

$\vdash$ 0 1 1 Δ

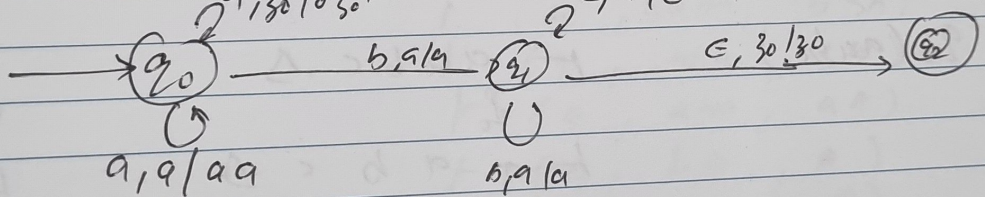
q_0

$\vdash$ 0 1 1 Δ

$\therefore$ not accepted

Q3) Construct PDA for accepting the following language.

$$L = \{ a^m b^n c^m \mid m, n >= 1 \}.$$



Transition function.

$(q_0, z_0) = (q_0, a z_0)$

$(q_0, a, a) = (q_0, aa)$

$(q_0, b, a) = (q_1, a)$

$(q_1, c, a) = (q_1, \epsilon)$

$(q_1, \epsilon, z_0) = (q_2, z_0)$.

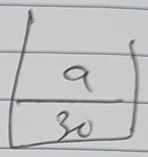
THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

PDA can be defined as:

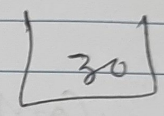
$M = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, \in, z_0\}, q_0, z_0, \{q_2\}, \delta)$

Simulation:-

$q_0(a, bc)$ ⊢ a b c Δ (q0 under a)

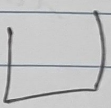


⊢ a b c Δ (q1 under b)



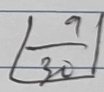
⊢ a b c Δ (q1 under c)

⊢ a b c Δ (q2 under Δ)

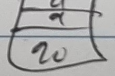


∴ accepted.

$q_0(aabc)$ ⊢ a a b c Δ (q0 under a)



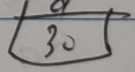
⊢ a a b c Δ (q0 under a)



⊢ a a b c Δ (q1 under b)

⊢ a a b c Δ (q1 under c)

⊢ a a b c Δ (q1 under Δ)



∴ Not accepted.

1) Design PDA for following CFG show acceptance of one valid string.

S → aAA | a.

A → bS | aS.

Let PDA be P.

P = ({q}, {a, b}, {S, A, a, b}, δ, q, S, ϕ)

R1 : δ (q, ε, S) → {(q, aAA) ; (q, a)}

R2 : δ (q, ε, A) → {(q, bS), (q, aS)}

R3 : δ (q, a, a) → (q, ε)

R4 : δ (q, b, b) → (q, ε)

Simulation :

δ (q, ababa, ε) ⊢ (q, ababa, aAA) (R1)

⊢ (q, baba, AA) (R2)

⊢ (q, baba, bSA) (R2)

⊢ (q, aba, SA) (R3)

⊢ (q, aba, aA) (R1)

⊢ (q, ba, A) (R3)

⊢ (q, ba, bS) (R2)

⊢ (q, a, S) (R4)

⊢ (q, a, a) (R1)

⊢ (q, ε)

∴ Accepted.

—: Assignment 5 :—

5.1 (B)

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

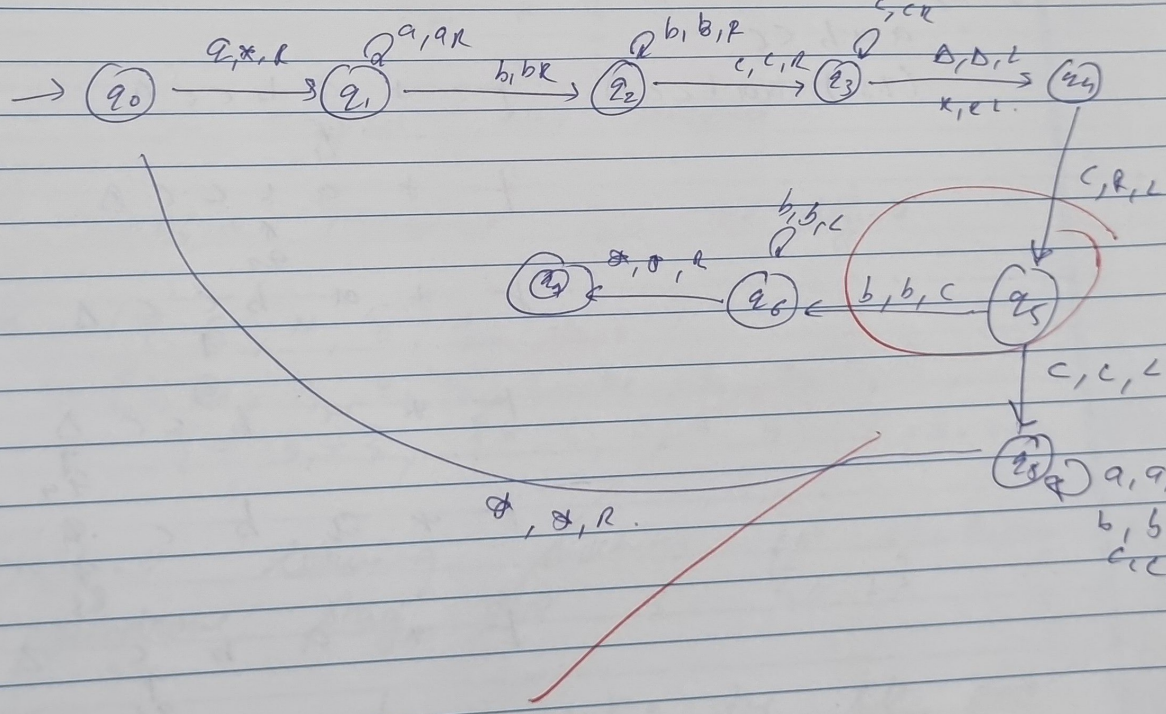
Q1) Design Turing Machine for recognizing the following language.

$L = \{ a^n b^m c^n \mid m, n >= 1 \}$.

logic:

We mark a with * and its corresponding c (starting from end of the string) with X if 'b' come following by '*' on the left after changing 'c' to 'X' we accept the string.

Language $L = \{ aabcc, abc, aaabbccc, abbc \ldots \}$.



Transition table

	a	b	c	⊔	&	Δ
q_0	$q_1, *, R$					
q_1	q_1, a, R	q_2, b, R	q_3, c, R		$q_4, 4, L$	q_n, Δ, L
q_2		q_2, b, R	q_3, c, R			
q_3			$q_5, 8, L$			
q_4		q_6, b, L	q_6, c, L			
q_5		q_6, b, L		$q_7, * R$.		
q_6						
(q_7)				$q_0 *, R$		
q_8	q_8, a, L	q_8, b, L	q_8, c, L			

Step

aabcc

(q_0) aabccΔ $\vdash$ * a b c c Δ

q_1

$\vdash$ * a b c c Δ

q_2

$\vdash$ * a b c c Δ

q_3

$\vdash$ * a b c c Δ

q_4

$\vdash$ * a b c & Δ

q_5

$\vdash$ * a b c & Δ

q_4

$\vdash$ * * b c & Δ

q_0

$\vdash$ * * b c & Δ.

q_2

⊢ * * b & & Δ
↑ q_4

⊢ * * b & & Δ
↑ q_0

⊢ * * b & & Δ
↑ q_0

⊢ * * b & & Δ
↑ q_7

Accepted.

ii) ab
(q_0) abΔ ⊢ * b Δ
↑ q_1
⊢ * b Δ
↑ q_2

∴ trapped Not accepted.

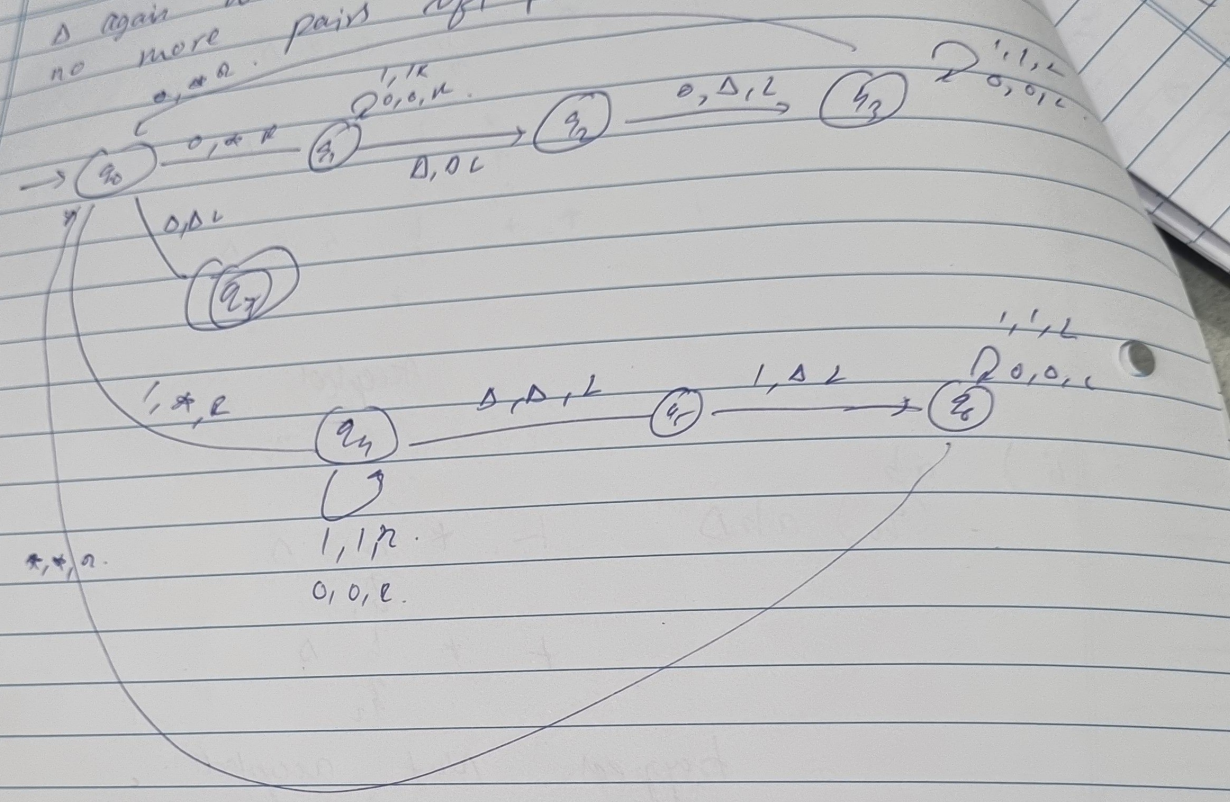
∴ Turning Machine can be defined

$M = (Q, \Sigma, \Gamma, \delta, q_0, \Delta, F)$
$= \{\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\},$
$\{a, b, c\}, \{a, b, c, *, \&\}, \delta, q_0, \Delta, \{q_7\}\}$.

Q2. Construct turning machine for checking even palindrome string over $\Sigma = \{0, 1\}$.

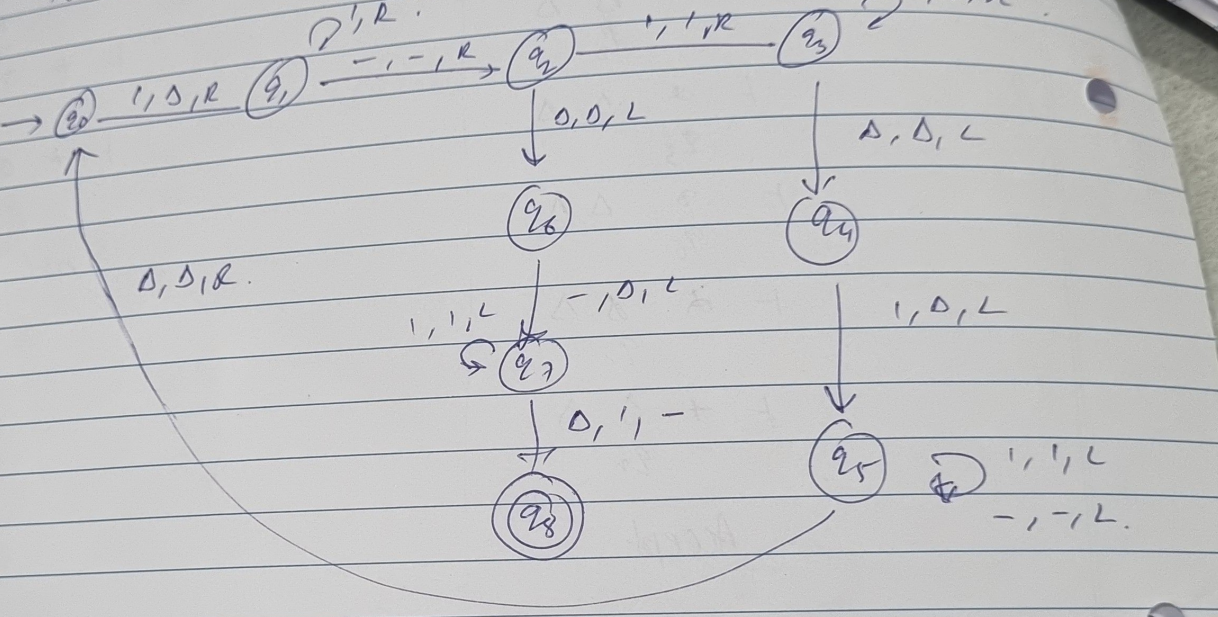
Logic. we keep marking the left part of palindrome with *, & search for its corresponding match on the other end of the.

Δ again we keep doing this until we have no more pairs left, then we accept.



	0	1	*	Δ
q_0	$q_1, *, R$	$q_4, *, R$		q_7, Δ, L
q_1	$q_1, 0, R$	$q_1, 1, R$		q_2, Δ, L
q_2	$q_3, 1, L$			
q_3	$q_3, 0, R$	$q_3, 1, R$	$q_0, *, R$	
q_4	$q_4, 0, R$	$q_4, 1, R$		q_5, Δ, L
q_5		q_6, Δ, L		
q_6	$q_6, 0, L$	$q_6, 1, L$	$q_0, *, R$	

& the 1's on the right side OR the
they as Δ two's as we work the
on the left side we will find that
the character after immediately after '-' is
then we move to left once change
the last target Δ to 1, and stop.



Simulation 1111 - 11 = 11

(q_1) 1111 - 11Δ ⊢ Δ1111 - 11Δ

⊢ Δ1111 - 11Δ

⊢ Δ1111 - 11Δ

⊢ Δ1111 - 11Δ

$\vdash \Delta 111 - 11 \Delta$

q_4

$\vdash \Delta 111 - 1 \Delta\Delta$

q_5

$\vdash \Delta 111 - 1 \Delta\Delta$

q_0

$\vdash \Delta\Delta 11 - 1\Delta\Delta$

q_1

$\vdash \Delta\Delta 11 - 1\Delta\Delta$

q_3

$\vdash \Delta\Delta 11 - \Delta\Delta\Delta$

q_5

$\vdash \Delta\Delta 11 - \Delta\Delta\Delta$

q_6

$\vdash \Delta\Delta\Delta 1 - \Delta\Delta\Delta$

q_4

$\vdash \Delta\Delta\Delta 1 - \Delta\Delta\Delta$

q_2

$\vdash \Delta\Delta\Delta 1\Delta\Delta\Delta\Delta$

q_8

$\vdash \Delta\Delta 11\Delta\Delta\Delta$

q_8

$\therefore$ final ans = 11

Transition Table

	1	-	Δ
q_0	q_1, Δ, R	$q_2, -, R$	q_6, Δ, L
q_1	$q_1, 1, R$		q_4, Δ, L
q_2	$q_3, 1, R$		
q_3	$q_3, 1, R$		q_0, Δ, R
q_4	q_5, Δ, L	$q_5, -, L$	
q_5	$q_5, 1, L$	q_7, Δ, L	
q_6			$q_8, 1, -$
q_7	$q_7, 1, L$		
q_8			

TM can be defined as

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

$= (\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}, \{1, -\}, \{1, -, \Delta\}$

$\delta, q_0, \Delta, \{q_8\})$

Q4) Define TM & Explain different variants of TM

→ Turing Machine.

A TM is a finite automata that can read, write & erase symbols on infinity long tape. The tape is divided into squar & each sequence contains a sybal

Matrhical definition.

$TM = (Q, \Sigma, N, \delta, q_0, \Delta, F)$.

$Q \Rightarrow$ set of states.

$\Sigma \Rightarrow$ set of i/p.

$N \rightarrow$ set of symbols on tape.

δ → Transition function
q_0 → initial state.
Δ → End of string.
F → final states.

Variants of TM:-

i) Multiple-tape TM: A k tape Turing Machine (for some k≥0) has k tapes & one tape head that reads & writes all of them one by one

ii) Two way infinite tape TM: inf. tape of two way TM is unbounded in both direction left & right.

iii) Multi tape TM: It has multi tape & is controlled by a single head TM multi tape TM is diff. than K-tape TM but expressive power is the same.

iv) multi-tape multi-head TM: It has multiple tapes & head per tape has its ocean head

v) Multi-dimensional tape TM: The head in can move in up, down, left right direction in the if.

vi) multi-head TM: A multi head TM contains multiple heads all of them move read & write independtly.

vii) Non-Deterministic TM: It has a single one way infinite tape for a given state & i/p symbol has at least one choice to move finite no of choices for.

not move, each choice has several choices of the path that it might follow. a given I/P string. It is equivalent to deterministic TM.

All variants can be simulated using different less complex variants.

THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

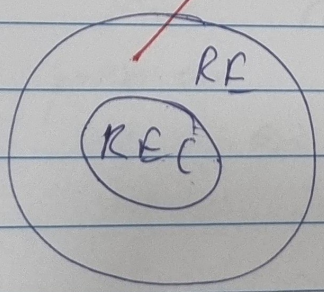
:- Assignment 6 :-

Q1). Write short note on recursive & Recursively enumerable language.

Ans: Recursively enumerable (RE) or type 0 language are from type 0 grammars. An RE language can be accepted by Turing machine which means it will enter into rejecting state for the strings which are not part of the language. Turing recognizable lang. are also called as Turing recognizable lang.

Recursive language (REC)

A recursive language (subset of RE) can be decided by TM which means it will enter into final state for the strings of language & rejecting state for the strings which are not part of the lang. eg: $L = \{ a^n b^n c^n \mid n >= 1 \}$ is recursive because we can construct a TM which will move to final state if the string is of the form $a^n b^n c^n$ else move to move to non-final state do the TM will always halt in this. The Relation between RE & REC lang follows fig.



Closure Properties of Recursive language:

i) Union If L_1 & If L_2 are two recursive lang. Then union $L_1 \cup L_2$ will also be recursive because if TM halts for L_1 & halts for L_2 if will also halt for $L_1 \cup L_2$

ii) Concatenation if L_1 & If L_2 are two recursive language then concatenation $L_1 . L_2$ will also be recursive.

iii) Kleene closure: If L_1 is recursive its kleene closure L_1^* will also be recursive.

iv) Intersection & complement: If L_1 & L_2 are two recursive lang. Then inter. $L_1 \cap L_2$ will also be recursive.

What is Halting Problem? Explain in detail

The halting problem is determining whether a comp. prog. will eventually stop or run for. ever Creating a general algo. that can. accurately predict this for all program is impossible Alan TM prof showed no way to solve this problem for all cases. An undecidable problem is a sort of computational problem requiring a yes/NO answer but where no comp. prog. can give the proper ans. all of the time, that is, any possible algorithm or program would sometimes give the wrong ans. Or run forever without providing any answer.

Proof:

i) Assume we can create a machine called (P, I) where HM is halting machine P is the program, I is the input. After given both i/p the machine HM will o/p when or not the program P terminates.

HM (P, I)

Yes (if the prog HALTS)

No (if the prog does not HALT)

ii) Now, create an inverted machine TM that takes a prog. P as a i/p

(M(P)

if HM(P,P) == Yes → loop forever.

if HM(P,P) == No → HALT

iii) Now, take a situation where the program TM is passed to the TM function as an input

IM (IM)

HM (IM, IM) == Yes (HALT)

↓

loop forever.

HM (IM, IM) == No (NOT HALT)

It will now halt because of the above not halting condition.

It is impossible for the outer fun to halt if its inner fun is in a loop (LHS) it is likewise. inputs for outer fun. to halt even if its inner func. is halting (LHS).

Q3) Write detailed explation on Rice theorem.

→ It states that any non-trivial semantic properties of a lang. which is recgined by a turing machine is undecidable machines that satisfy that property.

formal Definition: If P is a lang of all Turing lang. and the property L_P is recognized by TM M $L_P = \{ \langle M \rangle \mid L(M) \in P \}$ is undecidable.

Proof: Suppose, a property P is non-trivial & let since P is nontrivial at least one lang. satisfies P, i.e $(M_f) \in P$ ∃ Turing Machine M_0.

A function that maps an instance $M \times w = \{ \langle M, w \rangle \mid M$ accepts input $w \}$ to a TM such that

a) if M accept w & N accepts the same long as M_0, then $L(M) = L(M_0) \in P$.

b) If M does not accept w & N accept then $L(N) = \phi \in P$

Since ATM is undecidable & it can be reduced to LP is also undecidable.

qn) Define post correspondence problem. Prove that PCP with two lists $x = \{b, bab^3, ba\}$ & $y = \{b^3, ba, a\}$ have a solution.

=> The post corresponding problem (PCP) is a problem in the field of Theoritical computer sci. & math. logic. It is decision problem that can be used to illustrate the concept of undecidability & is closely related to the theory of formal lang. & automata. The problem is:

Given a finit set of domans : each with 2 strs can upper of a "lower" strs. written on it & a target strs can you find a sequence of domans choices from the set. such that when you concate the upper strs. it matches the concate noted lower strs.

$x = \{b, bab^3, ba\}$

$y = \{b^3, ba, a\}$.

1) b
2) b a bbb
3) b a

bbb
ba
a

i) find a domino where starting symb. are same.
∴ 2nd domino.
up: babbb down: ba.

ii) lets match the no. of bs.
∴ pick first domino.
b a bbbb babbb.

iii) To even out b's we pick first one again
b a bbbbbb babbbbbb

iv) No, upper part has one b letter so we can pick the best domino.

b a bb bbbb a ba bbbbbb a.

final sequence 2113.