VIT Vidyalankar Institute of Technology

# End Semester Examination (MU)

**Evaluation of University Question Paper Solution Prepared by Faculty**

Subject (Write in full) : THEORY OF COMPUTER SCIENCE (Regular / KT)

Exam : May 20 ____ / Nov. 20 19 Exam Date : 25/11/2019 Q. Paper Code : 78070

Department : CMPN Year : FE/SE/TE/BE/ME/MMS Semester : V Scheme : CBSGS/CBCS

Handwritten Solution Prepared by : RAVINDRA S. SANGLE

Name of the Subject Cluster : SYSTEM SOFTWARE

Name of the Cluster Mentor / Assessor : PROF- PANKAJ VANWARI

1st Assessment :

| Question No. | Marks Obtained | | | | | | Total |
|---|---|---|---|---|---|---|---|
| 1 | (a) 05 | (b) 04 | (c) 05 | (d) 05 | (e) | (f) | 19 |
| 2 | (a) 10 | (b) 10 | (c) | (d) | (e) | (f) | 20 |
| 3 | (a) 10 | (b) 10 | (c) | (d) | (e) | (f) | 20 |
| 4 | (a) 10 | (b) 10 | (c) | (d) | (e) | (f) | 20 |
| 5 | (a) 10 | (b) 10 | (c) | (d) | (e) | (f) | 20 |
| 6 | (a) 10 | (b) 10 | (c) 09 | (d) 09 | (e) | (f) | 38 |
| Total (Out of 140 marks) | | | | | | | 137 |

Signature of Assessor / Cluster Mentor : Pankaj Vanwari - VAs.

2nd Assessment :

| Question No. | Marks Obtained | | | | | | Total |
|---|---|---|---|---|---|---|---|
| 1 | (a) | (b) | (c) | (d) | (e) | (f) | |
| 2 | (a) | (b) | (c) | (d) | (e) | (f) | |
| 3 | (a) | (b) | (c) | (d) | (e) | (f) | |
| 4 | (a) | (b) | (c) | (d) | (e) | (f) | |
| 5 | (a) | (b) | (c) | (d) | (e) | (f) | |
| 6 | (a) | (b) | (c) | (d) | (e) | (f) | |
| Total (Out of marks) | | | | | | | |

Signature of Assessor / Cluster Mentor : _____



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|
| | Q1) a) | Explain post correspondance problem. (5m) |
| | Soln | Let $A = w_1, w_2, w_3 \dots w_k$ and $B = x_1, x_2, x_3 \dots x_k$ be string over some alphabet $\Sigma$. |
| | | Post correspondance problem (PCP) is to find the correspondance sequence of integers, $i_1, i_2, i_3 \dots i_m$ for $m \geq 1$ such that $w_{i_1}, w_{i_2}, w_{i_3} \dots w_{i_m} = x_{i_1}, x_{i_2}, x_{i_3} \dots x_{i_m}$ |
| | | The sequence $i_1, i_2, i_3 \dots i_m$ is considered to be a solution for PCP instance. Each PCP instance is constituted by some set of values for A and B. |
| | | Note that the entire class of PCP instances is unsolvable. If we consider the PCP as a generic class of all such instances, then it is unsolvable. Furtheemore, theee exist no generic algo that can find a solution for any such PCP instance, hence it is also an undecidable problem. |
| | | However for a few value of A and B it might have a solution. |



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

Consider the following ex:

a) Post Correspondance problem with solution

Let $\Sigma = \{0,1\}$ and let A and B be defined as table below,

| i | A | B |
|---|---|---|
| | $w_i$ | $x_i$ |
| 1 | 0 | 000 |
| 2 | 01000 | 01 |
| 3 | 01 | 1 |

Find the correspondance sequence of int $i_1, i_2, i_3 \dots i_m$ for $m \geq 1$ such that $w_{i_1}, w_{i_2}, \dots w_{i_m} = x_{i_1}, x_{i_2} \dots x_{i_m}$.

soln

The given PCP instance has a solution for $m=4$

$i_1 = 2$

$i_2 = 1$

$i_3 = 1$

$i_4 = 3$

Observe that,

$w_2 w_1 w_1 w_3 = (01000)(0)(0)(01) = 010000001$

$x_2 x_1 x_1 x_3 = (01)(000)(000)(1) = 010000001$

Hence $w_2 w_1 w_1 w_3 = x_2 x_1 x_1 x_3$.



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

b) Post Correspondance problem without solution

Let $\Sigma = \{1, 0\}$ and Let A and B be defined as table below.

| i | A | B |
|---|---|---|
| | $w_i$ | $x_i$ |
| 1 | ba | bab |
| 2 | abb | bb |
| 3 | bab | Abb |

Find correspondance sequence of int $i_1, i_2, i_3 \cdots i_m$ for $m \geq 1$ such that,

$$w_{i_1}, w_{i_2}, w_{i_3} \cdots w_{i_m} = x_{i_1}, x_{i_2}, x_{i_3} \cdots x_{i_m}$$

soln

This PCP instance is unsolvable since we can not find any correspondance sequence of integers as sequired.

Therefore every instance of the given PCP might have a solution. In other words the given PCP is undecidable that means given PCP does not have any algorithmic soln.



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

Q1) b) Differentiate between FA and PDA. (5m)

Sol^n

| | Finite Automata (FA) | Push Down Automata (PDA) |
|---|---|---|
| i) | Finite Automata (FA) is a simple idealized machine used to recognize patterns within input taken from some character set. | Push Down Automata (PDA) is a type of automation that employs a stack. |
| ii) | It doesn't has the capability to store long sequence of input alphabets. | PDA has a stack to store input alphabets. |
| iii) | FA can be constructed for type-3 grammar | PDA can be constructed for type-2 grammar |
| iv) | Input alphabets are accepted by reaching "final states". | Input alphabets are accepted by reaching :- i) Empty stack ii) Final state |
| v) | NFA can be converted into equivalent DFA | NPDA has more capability than DPDA |
| vi) | FA consist of 5 tupples $M=\{Q, \Sigma, \delta, q_0, F\}$ | PDA has 7 tupples $M=\{Q, \Sigma, \delta, \Gamma, q_0, z_0, F\}$ |
| vii) | FA can be constructed for RL | PDA can be constructed for CFG |



Scanned by CamScanner

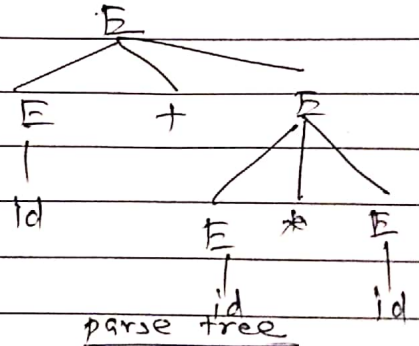
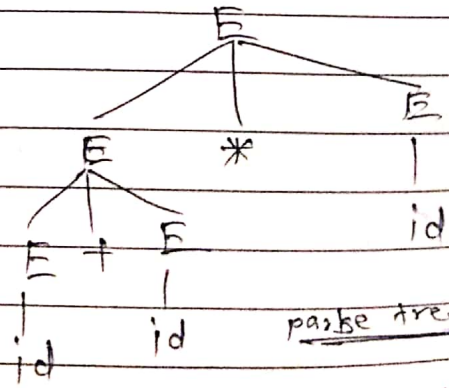
| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|
| | Q1) c) | Define Regular Expression and obtain a Regular expression such that $L\{R\} = \{w \mid w \in (0+1)^*\}$ with at the most three zeros. (5m) |
| | Soln | The languages accepted by finite automata are easily described by simple expression called regular expression. The class of languages accepted by FA is precisely the class of languages describable by regular expressions. |
| | | Let $\Sigma$ be a finite set of symbols and let $L, L_1$ and $L_2$ be sets of strings for $\Sigma^*$. |
| | | i) Concatenation. - $L_1 . L_2 = \{xy \mid x \in L_1 \text{ and } y \in L_2\}$ |
| | | ii) Kleen closure. $L^* = \bigcup_{i=0}^{\infty} L^i$ |
| | | iii) Positive closure - $L^+ = \bigcup_{i=1}^{\infty} L^i$ |
| | | ie $L^+ = L . L^*$ |
| | | $L\{R\} = \{w \mid w \in (0+1)^*\}$ with at the most three zeros. |
| | | $\Sigma = 1^* + 1^*01^* + 1^*01^*01^* + 1^*01^*01^*01^*$ |



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|
| | Q1) d) | What is ambiguous grammar? Check whether following grammar is ambiguous or not (5m) |
| | | $E \rightarrow E+E \mid E*E \mid (E) \mid id.$ |
| | Sol^n | A context free grammar (CFG) G is an ambiguous if there is at least one string in L(G) having two or more distinct derivation trees. (or equivalently two or more distinct leftmost or rightmost derivations). |
| | | Consider a CFG. $E \rightarrow E+E \mid E*E \mid (E) \mid id$ |
| | | string. "id + id * id" |
| | | LMD. $E \rightarrow E+E$ <br> $E \rightarrow id + E$ <br> $E \rightarrow id + E * E$ <br> $E \rightarrow id + id * E$ <br> $E \rightarrow id + id * id$ |
| | |   parse tree |
| | | LMD. <br> $E \rightarrow E*E$ <br> $E \rightarrow E+E*E$ <br> $E \rightarrow id + E*E$ <br> $E \rightarrow id + id * E$ <br> $E \rightarrow id + id * id$ |
| | |   parse tree |
| | | For same string we have multiple parse tree ∴ Given grammar is ambiguous. |



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

Space for Marks | Question No. | START WRITING HERE

Q2) a) Design a finite state machine to accept following language over the alphabet {0,1}
L = { w | w st with 0 & has odd length or st with 1 and has even length } (10m)

Soln

Finite state Machine (FSM) consist of finite set of states 'S' which accepts finite set of input alphabet 'I' to produce finite set of output 'O'.

FSM defines two functions.

① State function - $S \times I \rightarrow S$

② Machine function $S \times I \rightarrow O$

S = Finite set of states
= $\{q_s, q_0, q_1, q_2, q_3\}$

I = Finite set of i/p alphabet
= $\{0,1\}$

O = set of output = {Y, n}

STF = $S \times I \rightarrow S$

| S \ I | 0 | 1 |
|---|---|---|
| $\rightarrow q_s$ | $q_0$ | $q_1$ |
| $*q_0$ | $q_2$ | $q_2$ |
| $q_1$ | $q_3$ | $q_3$ |
| $q_2$ | $q_0$ | $q_0$ |
| $*q_3$ | $q_1$ | $q_1$ |

MAF = $S \times I \rightarrow O$

| S \ I | 0 | 1 |
|---|---|---|
| $\rightarrow q_s$ | Y | N |
| $*q_0$ | N | N |
| $q_1$ | Y | Y |
| $q_2$ | Y | Y |
| $*q_3$ | N | N |

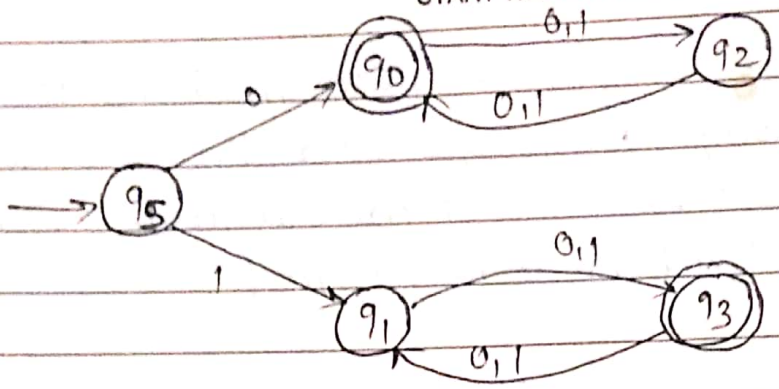


Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

Space for Marks | Question No.

START WRITING HERE




Example.

$\delta(q_5, 01011)$
$\delta(q_0, 1011)$
$\delta(q_2, 011)$
$\delta(q_0, 11)$
$\delta(q_2, 1)$
$q_0$ Accept

$\delta(q_5, 100)$
$\delta(q_1, 00)$
$\delta(q_3, 0)$
& $q_1$ Reject

(Q2) b) Give and explain formal definition of pumping lemma for Regular language and P.T follo language is not regular (10m)

$L = \{ 0^i \mid i \text{ is prime number} \}$

Soln

Pumping lemma for CFL States that for any context free language L, it is possible to find two substrings that can be "pumped" any numbers of times and still be in the same language. For any language L, we break its string into five parts and pump second and fourth substring.



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

Pumping lemma, here is also used as tool to prove that a language is not context free language, because if one story does not satisfy its conditions, then the language is not CFL.

Thus if L is in CFL, these exist an integer n, such that for all $x \in L$ with $|x| \geq n$, there exist $u, v, w, x, y \in \Sigma^*$ such that $x = uvwxy$ and

i) $|vwx| \leq n$

ii) $|vx| \geq 1$

iii) for all $i \geq 0$, $uv^i wx^i y \in L$.

Say $W = 0^p$ where p is prime & $p > n$

Then there are strings x, y and z so that $W = xyz$ such a way

$|y| > 0$

$|xy| \leq n$

$\therefore W = 0^q 0^s 0^{p-s-q}$ where $s > 0$ & $q + s \leq n$

Then by pumping lemma any string $xy^i z \in L$ & $i \geq 0$

∴ Check that

$xy^i z = 0^q (0^s)^i 0^{p-q-s}$

$= 0^{q+is+p-q-s}$

$= 0^{p+s(i-1)}$

put $i = 2p+1$



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|
| | | $= 0^{p+ps}$ |
| | | $= 0^{p(s+1)}$ |
| | | i.e length of $xy^iz$ is factor of two number means it is not prime number |
| | | Hence length of $xy^iz$ is not prime no. |
| | | $\therefore xy^iz \notin L$ |
| | | Hence, by contradiction L is not a regular language. |



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

Q3) a) Construct PDA accepting the language. (10m)

$L = \{a^{2n} b^n \mid n \geq 0\}$

Soln

Mathematical model of PDA is defined with 7-tuples.

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

$Q$ = finite set of states $= \{q_0, q_1, q_2, q_3, q_F\}$

$\Sigma$ = Finite set of i/p alphabet. $= \{a, b\}$

$\delta$ = Transition function

$$Q \times (\Sigma \cup \lambda) \times \Gamma \rightarrow Q \times \Gamma^*$$

$\Gamma$ = Stack alphabet

$q_0$ = Start state

$F$ = Final state ($q_F$)

$R$ = Initial stack top

Logic = For 'a' push 'X'
for 'b' pop 'X'

$\delta(q_0, a, R) = (q_0, XR)$ — NOP

$\delta(q_1, a, R) = (q_2, XR)$ — push

$\delta(q_2, a, X) = (q_1, XX)$ — NOP

$\delta(q_1, a, X) = (q_2, XX)$ — push

$\delta(q_2, b, X) = (q_3, \epsilon)$ — pop

~~$\delta(q_2, \epsilon, R) = q_F, R$ Accept~~

$\delta(q_3, b, X) = (q_3, \epsilon)$ - pop

$\delta(q_3, \epsilon, R) = q_F, R$ Accept

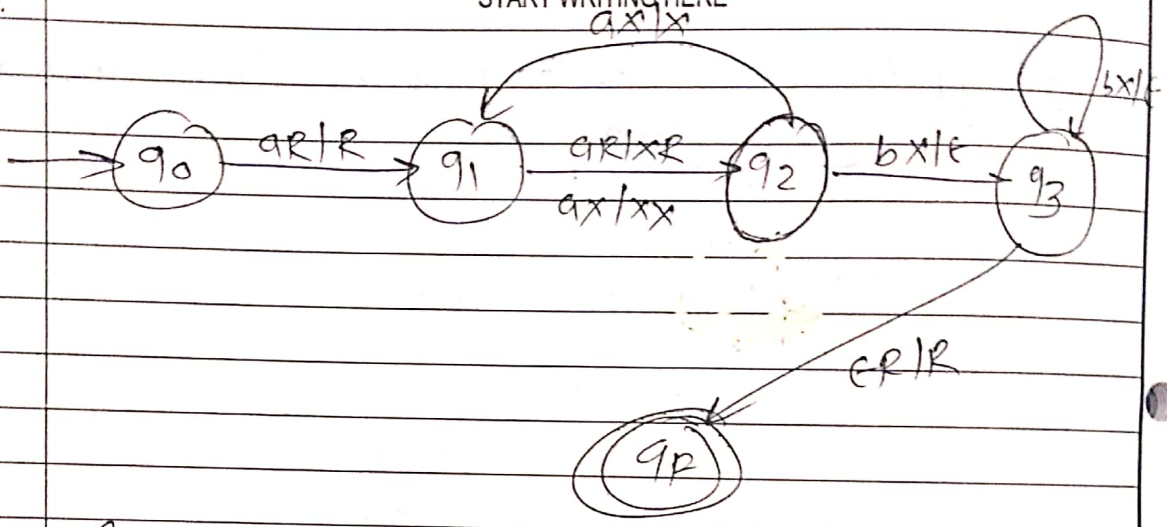


Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. |
|---|---|

START WRITING HERE




Example.

$\delta(q_0, aaaabb, R)$
$\delta(q_1, aaabb, R)$
$\delta(q_2, aabb, xR)$
$\delta(q_1, abb, xR)$
$\delta(q_2, bb, xxR)$
$\delta(q_3, b, xR)$
$\delta(q_3, \epsilon, R)$
$q_F, R$ Accept

$\delta(q_0, aaab, R)$
$\delta(q_1, aab, R)$
$\delta(q_2, ab, xR)$
$\delta(q_1, b, xR)$
~~Reject~~ Reject state

Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

Space for Marks | Question No. | START WRITING HERE

Q3) b) Consider the following grammar

$S \rightarrow iCtS \mid iCtSeS \mid a$

$C \rightarrow b$

For the string "ibtaeibta" find the following (10m)

i) Leftmost derivation
ii) Rightmost derivation
iii) Parse tree
iv) check if above grammar is ambiguous

Soln

Consider the given CFG

$S \rightarrow iCtS \mid iCtSeS \mid a$

$C \rightarrow b$

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

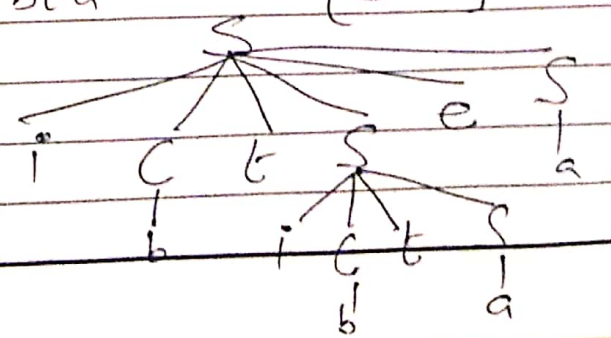
$V = \{S, C\}$, $T = \{a, e, i, b\}$

P = prodn rule, S = start variable

i) Leftmost derivation (LMD)

$S \rightarrow iCtSeS$ [$S \rightarrow iCtSeS$]
$\rightarrow ibtSeS$ [$C \rightarrow b$]
$\rightarrow ibtaeS$ [$S \rightarrow a$]
$\rightarrow ibtaeiCtS$ [$S \rightarrow iCtS$]
$\rightarrow ibtaeibtS$ [$C \rightarrow b$]
$\rightarrow ibtaeibta$ [$S \rightarrow a$]

parse tree for LMD






Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

Rightmost derivation. (RMD).

$S \rightarrow iCtSeS$ [S → iCtSeS]

$\rightarrow iCtSeiCtS$ [S → iCtS]

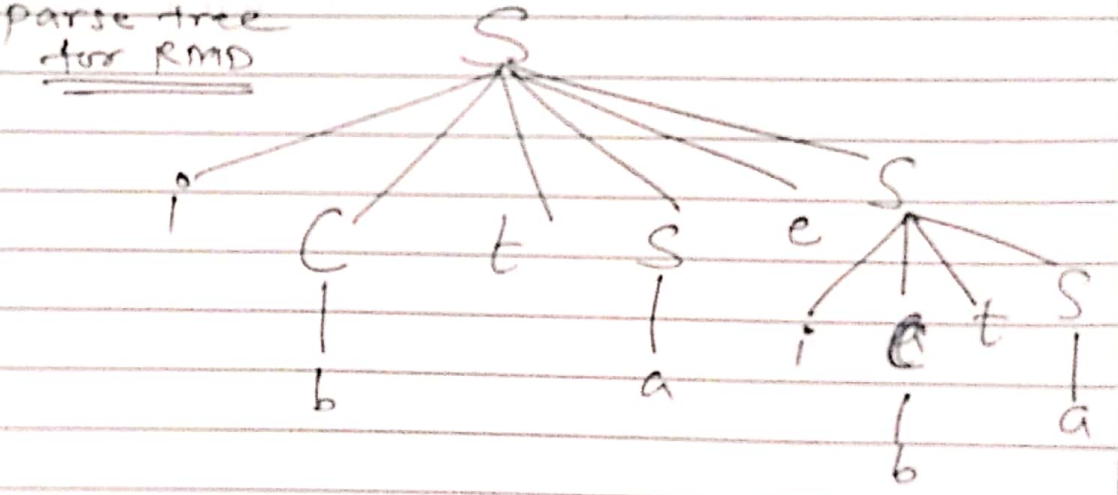
$\rightarrow iCtSeiCta$ [S → a]

$\rightarrow iCtSeibta$ [C → b]

$\rightarrow iCtaeibta$ [S → a]

$\rightarrow ibtaeibta$ [C → b]

Parse tree for RMD




The given grammar is ambiguous but with respect to the given string this grammar is NOT ambiguous because there exist only one parse tree to derive the sentence from given grammar.



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

Q4) a) Construct PDA to check $\{ wcw^R / w \in (a+b)^* \}$ where $w^R$ is reverse of w & c is constant. (10m)

Soln

Mathematical model of PDA is defined as

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

$Q$ = Finite set of states

$\Sigma$ = Input alphabet = $\{a, b, c\}$

$\delta$ = Transition function

$\Gamma$ = stack alphabet

$q_0$ = Start state

$F$ = final state = $q_F$

$R$ = Initial stack top

Logic –

For string 'w' ⇒ For each 'a' push X
⇒ For each 'b' pop X
Bypass the alphabet 'c'

for string $w^R$ ⇒ For each 'a' pop 'X'
⇒ for each 'b' pop 'Y'

$\delta(q_0, a, R) = (q_0, XR)$
$\delta(q_0, b, R) = (q_0, YR)$
$\delta(q_0, a, X) = (q_0, XX)$
$\delta(q_0, a, Y) = (q_0, XY)$
$\delta(q_0, b, X) = (q_0, YX)$
$\delta(q_0, b, Y) = (q_0, YY)$
$\delta(q_0, c, X) = (q_1, X)$
$\delta(q_0, c, Y) = (q_1, Y)$

$\delta(q_1, a, X) = (q_1, \epsilon)$
$\delta(q_1, b, Y) = (q_1, \epsilon)$
$\delta(q_1, \epsilon, R) = q_F, R$ Accept
$\delta(q_0, c, R) = q_F$ Accept

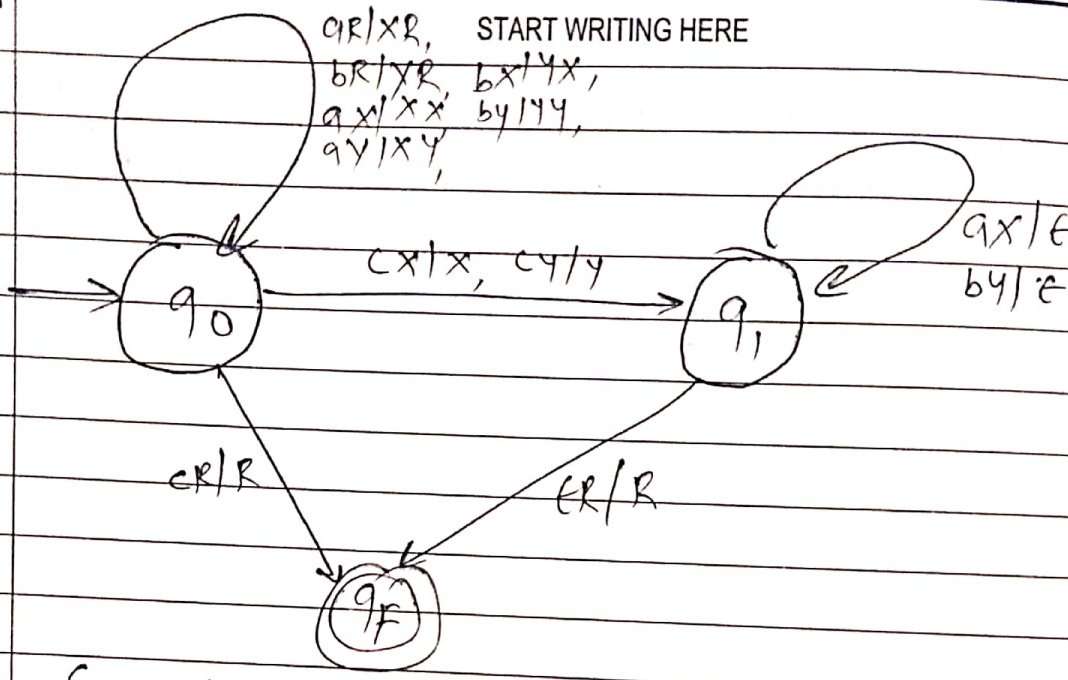


Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. |
|---|---|

START WRITING HERE




Example - ~~NPDA~~

~~δ(q0, abbcbba~~

δ(q0, abbcbba, R)

δ(q0, bbcbba, XR)

δ(q0, bcbba, YXR)

δ(q0, cbba, YYXR)

δ(q1, bba, YYXR)

δ(q1, ba, YXR)

δ(q1, a, XR)

δ(q1, ϵ, R)

qF, R Accept

δ(q0, abca, R)

δ(q0, bca, XR)

δ(q0, ca, YXR)

δ(q0, a, YXR)

Reject state



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|
| | Q4/b) | Convert following CFG to CNF (10m) |
| | | $S \rightarrow 0A0 \mid 1B1 \mid BB$ |
| | | $A \rightarrow C$ |
| | | $B \rightarrow S \mid A$ |
| | | $C \rightarrow S \mid \epsilon$ |
| | Soln | Consider a CFG as $G = (V, T, P, S)$ |
| | | $V$ = set of variables = $\{S, A, B, C\}$ |
| | | $T$ = Set of terminals = $\{0, 1, \epsilon\}$ |
| | | $P$ = production rule |
| | | $S$ = start variable |
| | | In the above CFG $C \rightarrow \epsilon$ is a null production and 'C' is nullable variable |
| | | As $A \rightarrow C \rightarrow \epsilon$<br>$B \rightarrow A \rightarrow C \rightarrow \epsilon$<br>$S \rightarrow BB \rightarrow \epsilon$<br>$\therefore$ A, B, S & C all are nullable variables |
| | | After elimination of nullable variables the resultant CFG becomes |
| | | $S \rightarrow 0A0 \mid 00 \mid 1B1 \mid 11 \mid BB \mid B$ |
| | | $A \rightarrow C$ |
| | | $B \rightarrow S \mid A$ |
| | | $C \rightarrow S$ |
| | | Here, $A \rightarrow C$, $C \rightarrow S$, $B \rightarrow S$, $S \rightarrow B$ are unit productions |
| | | $\therefore$ After elimination of unit production the resultant grammar becomes, |



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

$S \rightarrow 0A0 \mid 00 \mid 1B1 \mid 11 \mid BB$

$\because A \rightarrow C \rightarrow S$ & $B \rightarrow S$

$\therefore S \rightarrow 0S0 \mid 00 \mid 1S1 \mid 11 \mid SS$

This is simplified CFG.

Now convert the resultant simplified CFG to CNF. ($A \rightarrow a$ or $A \rightarrow BC$)

$\therefore$ Chomsky Normal form (CNF) is.

$S \rightarrow XC_1 \mid XX \mid YC_2 \mid YY \mid SS$

$X \rightarrow 0$

$Y \rightarrow 1$

$C_1 \rightarrow SX$

$C_2 \rightarrow SY$



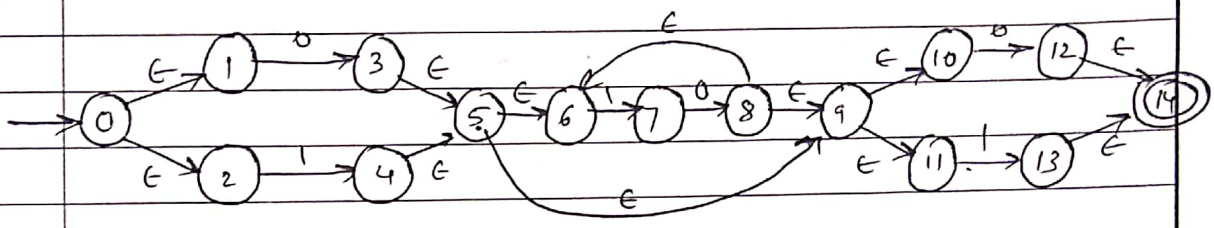
Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

Space for Marks | Question No. | START WRITING HERE

Q5) a) Convert $(0+1)(10)^*(0+1)$ in to NFA with ∈-moves and obtain DFA. (10m)

Soln R.E $\epsilon = (0+1)(10)^*(0+1)$




NFA with ∈-transition

NFA to DFA conversion -

| States (x) | | Y = ∈-closure(x) | δ(Y,0) | δ(Y,1) |
|---|---|---|---|---|
| → | {0} A | 0, 1, 2 | {3} | {4} |
| | {3} B | 3, 5, 6, 9, 10, 11 | {12} | {7, 13} |
| | {4} C | 4, 5, 6, 9, 10, 11 | {12} | {7, 13} |
| * | {12} D | 12, 14 | { } | {B} |
| * | {7, 13} E | 7, 13, 14 | {8} | { } |
| | ~~{13}~~ I | ~~13, 14~~ | ~~7~~ | ~~7~~ |
| | {8} F | 8, 9, 10, 11, 6 | {12} | {13, 7} |
| | { } G | { } | { } | { } |

Imp states → {1, 2, 6, 7, 10, 11}

DFA transition table / function -

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



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

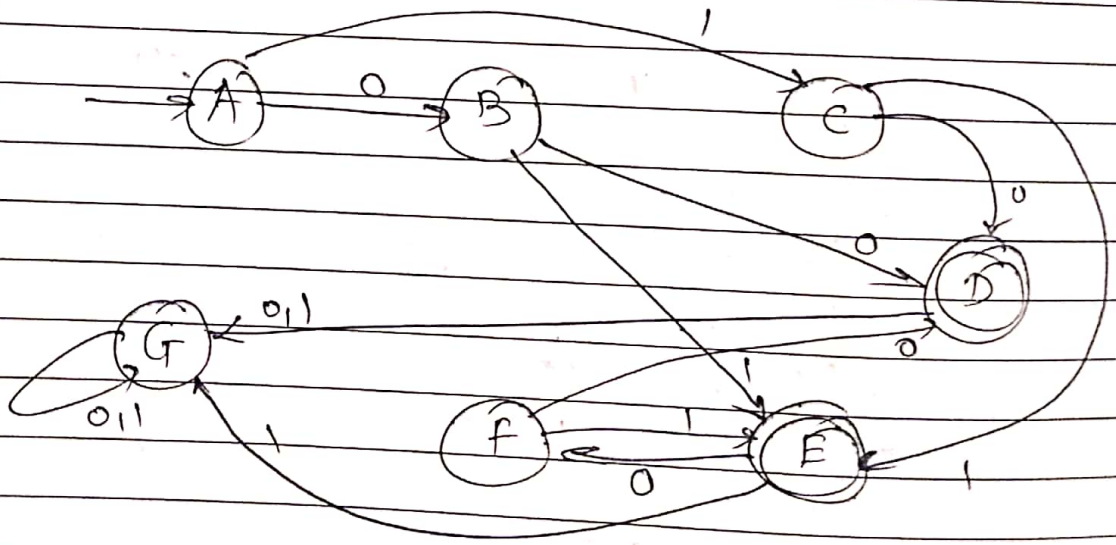
| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

$\delta : Q \times \Sigma \to Q$

| Q \ Σ | 0 | 1 |
|---|---|---|
| → A | B | C |
| B | D | E |
| C | D | E |
| * D | G | G |
| * G | F | G |
| F | D | E |
| G | G | G |

Merging states

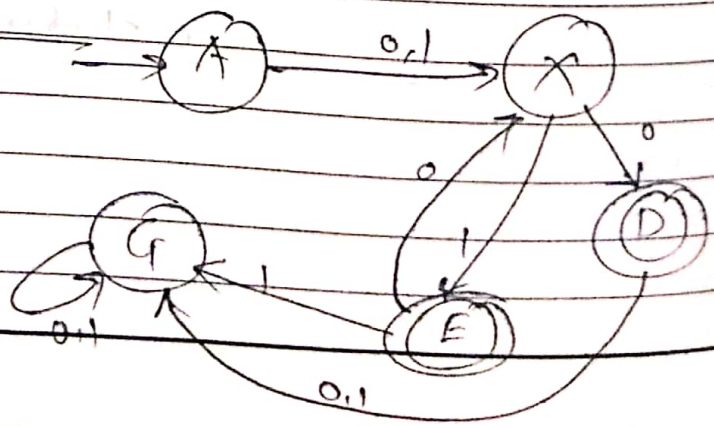
Merging state




DFA Min. States B, E & f can merge because they has same imp states

$\delta : Q \times \Sigma \to Q$

| Q \ Σ | 0 | 1 |
|---|---|---|
| → A | X | X |
| [B,E,F] X | D | E |
| * D | G | G |
| * E | X | G |
| G | G | G |






Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

Space for Marks | Question No. | START WRITING HERE

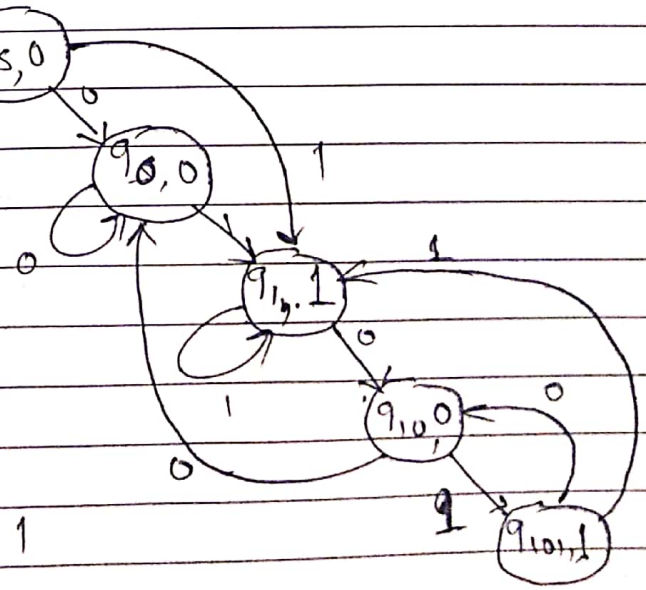
Q5) b) Construct Moore and Mealey Machine to convert each occurrence of 101 by 111 (10m)

Soln

NOTE :-
The given question is WRONG and No modifications received by Univeesity of Mumbai.

Moore Machine -
Mathematical Model of moose m/c is defined as,
$M = (Q, \Sigma, \delta, \Delta, \lambda, q_s)$
$Q$ = Finite set of states = $\{q_s, q_0, q_1, q_{10}, q_{101}\}$
$\Sigma$ = Finite set of i/p alphabet = $\{0,1\}$
$\delta$ = Transition function - $Q \times \Sigma \to Q$
$\Delta$ = Output alphabet = $\{0,1\}$
$\lambda$ = output mapping $\lambda(Q) = \Delta$, $q_s$ ~~$q_{10}$~~ start state
Transition table -
$\delta: Q \times \Sigma \to Q$

| $Q$ \ $\Sigma$ | 0 | 1 |
|---|---|---|
| $\to q_s$ | $q_0$ | $q_1$ |
| $q_0$ | $q_0$ | $q_1$ |
| $q_1$ | $q_{10}$ | $q_1$ |
| $q_{10}$ | $q_0$ | $q_{101}$ |
| $q_{101}$ | $q_{10}$ | $q_1$ |




$\lambda$ = o/p mapping
$\lambda(q_s) = 0$, $\lambda(q_{10}) = 0$
$\lambda(q_0) = 0$ $\lambda(q_{101}) = 1$
$\lambda(q_1) = 1$



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

Space for Marks | Question No. | START WRITING HERE

## Mealey Machine

Mathematical model of mealey m/c is defined as

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

$\delta$ = Transition function - $Q \times \Sigma \to Q$

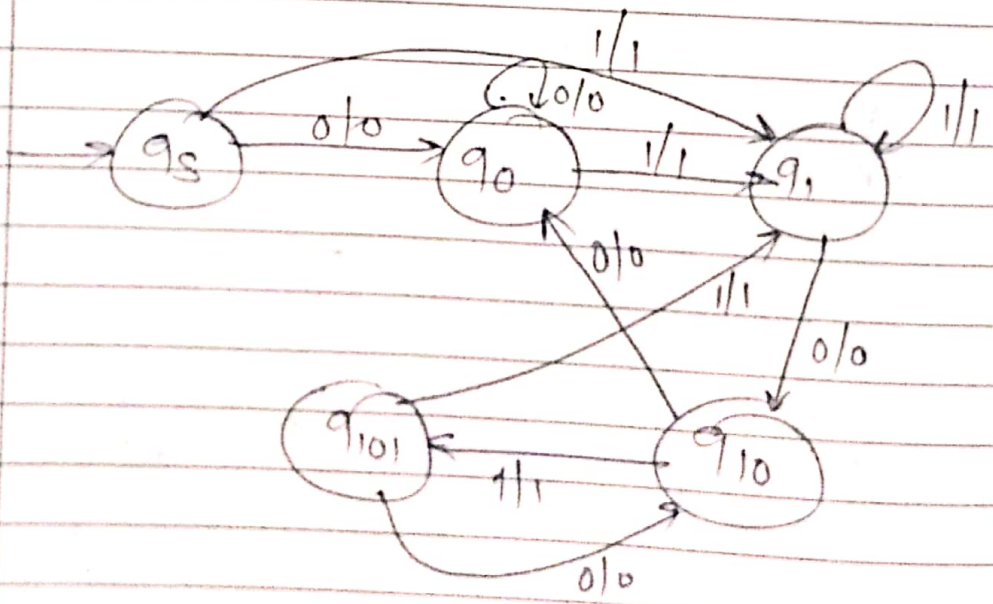
$\Delta$ = o/p symbol

$\lambda$ = o/p mapping = $Q \times \Sigma \to \Delta$

$\lambda = Q \times \Sigma \to \Delta$

| $Q$ \ $\Sigma$ | 0 | 1 |
|---|---|---|
| $\to q_s$ | 0 | 1 |
| $q_0$ | 0 | 1 |
| $q_1$ | 0 | 1 |
| $q_{10}$ | 0 | 1 |
| $q_{101}$ | 0 | 1 |

In moore m/c o/p symbol is associated with state. whereas in Mealey m/c o/p sym with every transition






Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

Space for Marks | Question No. | START WRITING HERE

Q6) Write a short note on following (Any 2) (20m)

a] Chomsky Hierarchy -

Chomsky has suggested four different classes of phrase structure grammar.

1) Type 0 (Unrestricted Grammar) -

The grammar which has no restrictions on the production is called as "Unrestricted Grammar".

It does not have any distinction as non-terminal symbols. The productions are of the type - $\alpha \rightarrow \beta$ where $\alpha$ & $\beta$ are string

ex $aaa \rightarrow bbba$, $aa \rightarrow ba$, $b \rightarrow aba$

- Such languages can not be tested
- It is the superset of all languages.

2) Type 1. (Context Sensitive Grammar)

As the name implies, the productions of this grammar are context sensitive ie productions can be applied if and only if their corresponding context conditions are satisfied.

∴ The productions for this grammar are of the form $\alpha P \beta \rightarrow \alpha Q \beta$

ie P will be replaced by Q under that condition it is preceeded by string $\alpha$ and followed by string $\beta$: ex. $bPa \rightarrow bQa$.



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

- Such languages can be tested or identified using Linear Bounded Automata
- It is the subset of type 0.

3] Type 2 (Context-free Grammar)

The grammar with the restriction that, on the left hand side of each production there should be only one symbol which is a "non-terminal" is called as CFG.

It is also called because there are no context conditions for applying production rules as that in context sensitive grammar

The productions are of this form.

$$P \longrightarrow \alpha$$

where P - variable, $\alpha$ - sentential form.

ex- $A \Rightarrow aAB$, $S \rightarrow aAbB$.

4] Type 3 (Regular or Linear Grammar)

Grammar is said to be linear or regular if

i) LHS of every prodn rule contains only one variable and

ii) RHS of every prodn rule contains at the most one variable which can appear at the leftmost or rightmost position.

- It is subset of type 2.

These regular languages can be expressed by expression called as regular expression



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

i) Left linear grammar - Non-terminal appears as the leftmost symbol on the RHS of the production.
The prodn are of the type $P \rightarrow Bb$

ii) Right linear grammar: Non-terminal appears as the rightmost symbol on the RHS of the productions.
The prodn are of the type $P \rightarrow bB$



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|
| | Q6) b) | **Halting problem :-** |
| | | For a given initial configuration of a TM, two cases arise, |
| | | i) The machine starting at this configuration will halt after a finite number of steps. |
| | | ii) The machine starting at this configuration never halts no matter how long it runs. |
| | | Given any TM, the problem of determining whether or not it ever halts is called the Halting problem. |
| | | To solve halting problem, we need some mechanism that consumes any functional matrix, input data type, and the initial configuration of TM and determines whether or not the TM will ever halt. Any such mechanism essentially should be an algo (TM) that needs to simulate any given TM for checking whether or not it halts. |
| | | In reality one can not solve the halting problem - the halting problem is unsolvable. This means that there exist no TM, which |



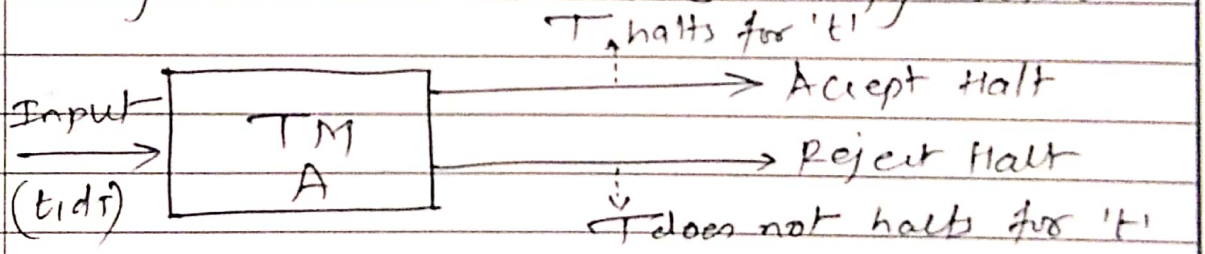
Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

can determine whether or not a given TM will ever halts.
Consider that the halting problem is unsolvable by contradiction.

i) Assume that there exist a TM A, which decides whether or not any computations by any TM T will ever halts, given the descriptions dT(SFm) of T, and the input-tape t of T. Then for every input (t, dT) to A; if T halts, then A reaches an "accept halt" state; else A reaches to a "reject halt" state as shown fig below



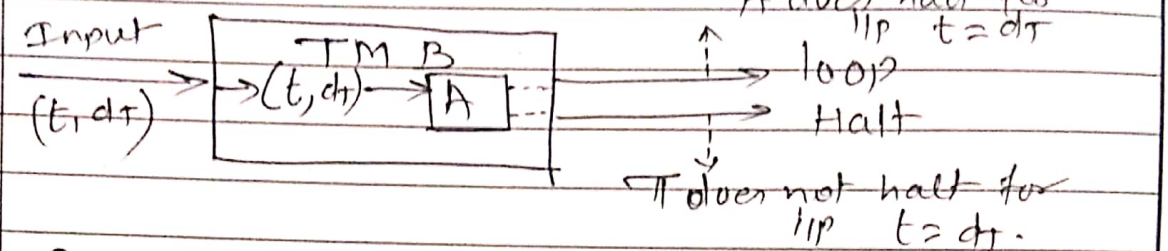

We now attempt to construct another TM B, which takes (t, dT) as input, it functions as follows—
First it copies the input and duplicates the same onto its tape. Then it takes this duplicated info tape as the input to A. Whenever A reaches the "accept Halt" state; B loops forever, and whenever A reaches the "Reject Halt" state B halts.

Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

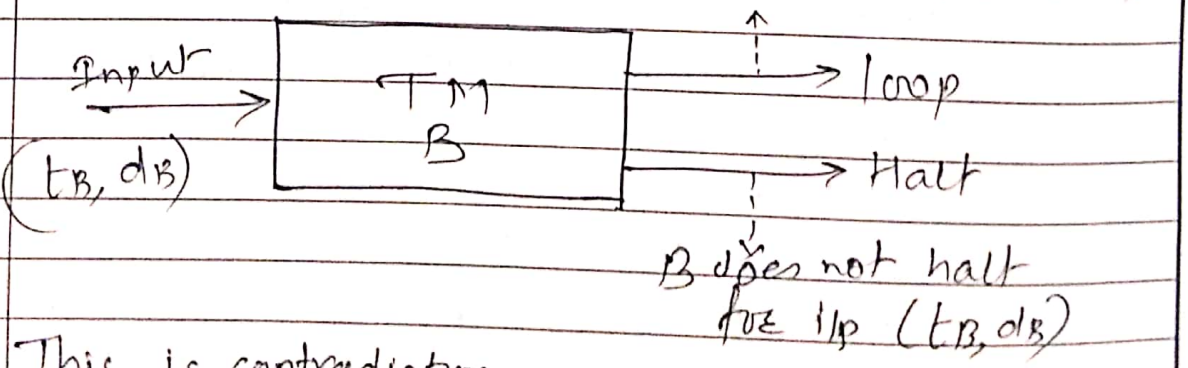
Space for Marks | Question No.

START WRITING HERE




Considering the original behaviours of A, we find that B, acts as follows:
It loops if T halts for i/p t, and halts if T does not halt for i/p t, Thus the working of TM B is exactly opposite of that TM T.

Now since B itself is TM, set us set T = B. In this case, B halts for the i/p if and only if B does not halts for same i/p, and loops forever if B halts for the input. Fig follo shows working of TM B, which takes itself as input.




This is contradiction,
Hence we conclude that no A which can decide whether or not any other TM, will ever halt cannot exist.
Therefore, we conclude that the halting problem is unsolvable.



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

Q6) c) **Rice's Theorem :-**

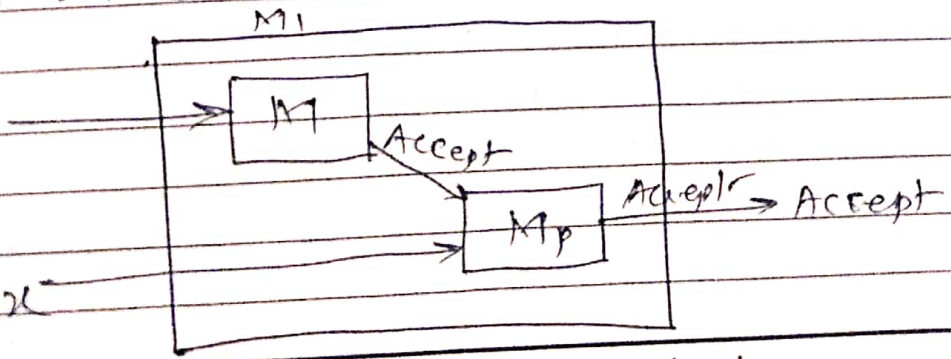
A property of RE languages form a set of RE languages. Thus if being a regular language is the property of the RE language, then it forms set of all regular languages. A property of RE language is said to be trivial if it is empty (ie not satisfied by any RE languages) or if it is satisfied by all RE languages. Otherwise it is said to be non-trivial.

**Statement of Rice Theorem.**

Every non-trivial property of an RE language is undecidable.

**proof.**

To prove Rice theorem, we need to show that $L_P$ is undecidable for a non-trivial property P. We shall proceed by reducing $L_U$ to $L_P$. As we know, if $L_U$ is undecidable, so is $L_P$. To demonstrate $L_U$ to $L_P$. this theorem we construct TM $M_1$ as shown below




TM $M_1$ that proves $L_P$ is undecided.



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

The construction of $M_1$ is such that if M does not accept w, then $L(M_1) = \emptyset$; if M accepts w, $L(M_1) = L_P$.

On input $(M, w)$ we create a TM $M_1$ as follows,

i) On input x, let the TM run on the string w until it accepts the string

ii) Next, run $M_P$ on x. Accept it iff $M_P$ does.

Note that $M_1$ accept the same language as $M_P$ if M accepts; w; $M_1$ accepts the empty language if M does not accept w.

Thus if M accepts w, the TM $M_1$ has the property P; else it does not have it. This turns $M_1$ into $M_P$ iff $(M, w)$ is in $L_U$. Hence $L_U$ is seduced to $L_P$. This proves that the property P is undecidable just likely.

The essence of Rice thm is that any problem, which requires determining the property of the language recognized by a given TM, is undecidable. The only exceptions are the trivial properties that are always either true or false.



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

Q6) d) **Universal Turing Machine** –

i) A Universal Turing Machine is a Turing machine which is all powerful or universal. It is universal in the sense that "it is capable of doing anything that any other TM can do".

2) The UTM should have the capability of imitating any TM 'T' given the following information on tape.

a) The description of 'T' in terms of its FM (Function Matrix) ie operations (program area on tape).

b) The initial configuration of 'T' ie starting state and the starting symbol. (state area on tape)

c) The processing data (input string) to be fed to 'T' (data area of tape)

2) This means, UTM can have both transition tables and data of any turing machine on its tape ie all Turing machine can be stored on its tape.



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

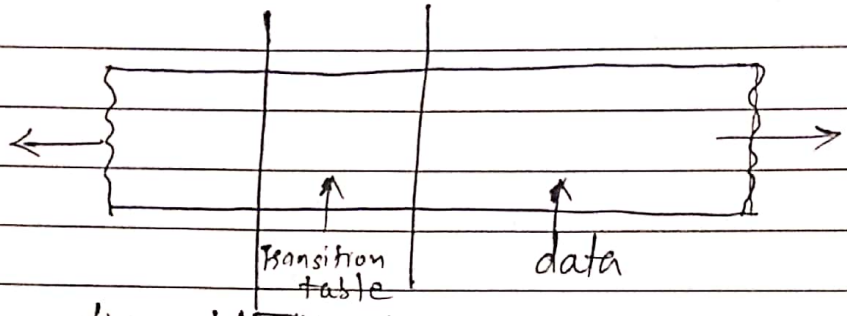



fig. - UTM tape

UTM should have an imitation algo to interpret correctly the rules of operation given in the FM of 'T'

UTM should have a table look-up facility and should perform the following steps -

Imitation Algorithm (Operations) -

i) Scan the square on the tape area of the tape and read the symbol that T reads to start and also the initial state of T.

ii) a) Move the tape to the program area containing FM of 'T' and find the row in FM headed by the state symbol read.

b) Along that row, find the column headed by the input symbol read and read the triplet : (new symbol, new state, direction to move)

iii) Move the tape to reach the appropriate



Scanned by CamScanner

| Total Marks of Question no. | | Examiner | |
|---|---|---|---|
| | | Moderator | |
| | | Re-Assessor | |

| Space for Marks | Question No. | START WRITING HERE |
|---|---|---|

square in the data area. Replace the symbol by new symbol, move the head in the required direction, and finally reach the state area and replace the state by new state and read the next symbol. Then go to step 2.

The UTM laid the foundation for
a) Stored program computers and
b) Interpretative implementation of programming languages

As UTM is also a TM; we should convert this imitation algorithm into FM for UTM.

Scanned by CamScanner