Module – 4

Pumping Lemma for Context-Free Languages

Let L be the context free language and is infinite. Let z is sufficiently long string and $z \in L$ so that $|z| \geq n$ where n is some positive integer. If the string z can be decomposed into combinations of strings

$$z = uvwxy$$

such that $|vwx| \leq n$

$$|vx| \geq 1$$

then $uv^i wx^i y \in L$ for $i = 0, 1, 2, \ldots$

Proof: According to pumping lemma, it is assumed that string $z \in L$ is finite and is context free language. We know that z is string of terminals which is derived by applying series of productions. Proof of this theorem leads to the following two cases.

Case 1: To generate a sufficiently long string z one or more variables must be recursive. $A \overset{*}{\Rightarrow} \alpha A \beta$

and should be applied more than once.

Proof of case 1:

Assume that no variable is recursive.

Since no non-terminal (variable) is recursive, each variable must be defined only in terms of terminals and/or other variables. Since those variables are also non-recursive, they have to be defined in terms of terminals and other variables and so on. If we keep applying the productions, there are no variables at all in the final derivation and finally we get a string of terminals and the generated string is finite. From this we conclude that there is a limit on the length of the string that is generated from the start symbol S. This contradicts

our assumption.

It means that one or more variables are recursive and hence the proof.

Proof of case2:

Let $z \in L$ is sufficiently long string and so the derivation must have involved recursive use of some non-terminal A and the derivation must have the form

$$S \overset{*}{\Rightarrow} uAy$$

A is used recursively

$$S \overset{*}{\Rightarrow} uAy \overset{*}{\Rightarrow} uvAxy$$

and the final derivation should be of the form

$$S \overset{*}{\Rightarrow} uAy \overset{*}{\Rightarrow} uvAxy \overset{*}{\Rightarrow} uvwxy = z$$

It implies the following derivations.

$$A \overset{*}{\Rightarrow} vAx$$

and

$$A \overset{*}{\Rightarrow} w$$

We can conclude that the derivation $A \overset{*}{\Rightarrow} vwx$ must also be possible.

Next we have to prove that the longest string vwx is generated without recursion since it is assumed that $|vwx| \leq n$. This can be easily proved since CFG that generates CFL does not contain ϵ productions and unit productions.

The derivation

$$A \overset{*}{\Rightarrow} vAx$$

used clearly shows that

$$|vx| \geq 1$$

Note from the derivation

$$S \overset{*}{\Rightarrow} uAy \overset{*}{\Rightarrow} uvAxy$$

that $uvAxy$ occurs in the derivation

and $A \overset{*}{\Rightarrow} vAx$

and $A \overset{*}{\Rightarrow} w$ are also possible, it follows that

$$uv^i wx^i y \in L$$

and hence the proof.

Show that the following languages are not context free.

1) Let $L = \{a^n b^n c^n : n \geq 0\}$

2) Let $L = \{a^{n^2} : n \geq 0\}$

3) Let $L = \{a^n b^m a^n : n, m \geq 0 \text{ and } n \geq m\}$

4) Let $L = \{wcw : w \in \{a, b\}^*\}$

1) $n = 3$

$z = \underbrace{a|aa}_{u}\underbrace{a|b}_{v}\underbrace{|b}_{w}\underbrace{b|}_{x}\underbrace{bccc}_{y}$

$|z| > n$

$9 > 3$ True

$|vwx| \leq n$

$3 = 3$ True

$|vx| \geq 1$

$2 > 1$ True

For $i = 0$

$uv^i wx^i y = uwy$

$= aabbccc$

$\therefore$ Hence proved

2) $n = 3$

$z = a^{3^2}$

$z = \underbrace{a|a|a|a|a}_{u|v|w|x|}\underbrace{aaaa}_{y}$

$|z| > n$

$9 > 3$ True

$|vwx| \leq n$

$3 = 3$ True

$|vx| \geq 1$

$2 > 1$ True

For $i = 0$

$uv^iwx^iy = uwy$

$= aaaaaaa$

Hence proved.

3) $n = 3$

$m = 2$

$z = \underbrace{a|a|a|b|b}_{u|v|w|x}\underbrace{aaa}_{y}$ $|z| > n$ $9 > 3$ True.

$|vwx| \leq n$

$3 = 3$ True

$|vx| \geq 1$

$2 > 1$ True

For $i = 0$

$uv^iwx^iy = uwy$

$= aababaaa$

Hence proved.

4) $w = ababb$

$n = 5$

$z = \underbrace{abababb}_{uvwxy}$

$|z| > n$

$10 > 5$ True

$|vwx| \leq n$

$3 < 5$ True

$|vx| \geq 1$

$2 > 1$ True

For $i = 0$

$uv^iwx^iy = uwy$

$= aababbabb$

Hence proved.

CFLs are closed under union, concatenation and star.

Theorem: If L_1 and L_2 are CFLs then $L_1 \cup L_2$, $L_1 . L_2$ and L_1^* also denote the CFLs and so the Context Free Languages are closed under union, concatenation, star-closure.

Proof: Let L_1 and L_2 are two CFLs generated by the CFGs

$$G_1 = (V_1, T_1, P_1, S_1)$$

and

$$G_2 = (V_2, T_2, P_2, S_2)$$

respectively and assume that V_1 and V_2 are disjoint.

Case 1: Union of two CFLs is CFL.

L_3 generated by the grammar

$$G_3 = (V_1 \cup V_2 \cup S_2, T_1 \cup T_2, P_3, S_3)$$

where S_3 is start symbol for grammar G_3

$$P_3 = P_1 \cup P_2 \cup \{ S_3 \rightarrow S_1 / S_2 \}$$

It is easy to prove that

$L_3 = L_1 \cup L_2$

If we assume $w \in L_1$ then S_3 is

$S_3 \Rightarrow S_1 \overset{*}{\Rightarrow} w$

If we assume $w \in L_2$ then S_3 is

$S_3 \Rightarrow S_2 \overset{*}{\Rightarrow} w$

So if $w \in L_3$, one of the derivations

$S_3 \Rightarrow S_1$

or

$S_3 \Rightarrow S_2$ is possible.

$\therefore L_3 = L_1 \cup L_2$

Thus it is proved that context free languages are closed under union.

<u>case 2: Concatenation of two CFLs is CFL</u>

Consider the language L_4 generated by the grammar

$G_4 = (V_1 \cup V_2 \cup S_4, T_1 \cup T_2, P_4, S_4)$

where

S_4 is start symbol for G_4 and $S_4 \notin (V_1 \cup V_2)$

$P_4 = P_1 \cup P_2 \cup \{ S_4 \rightarrow S_1 S_2 \}$

It is clear from this the grammar G_4 is context free and the language generated by this grammar is context free and so

$$L_3 = L_1 L_2$$

Hence it is proved.

case 3: CFLs are closed under star-closure.

Consider the language L_5 generated by the grammar

$$G_5 = (V_1 \cup S_5, T_1, P_5, S_5)$$

S_5 is start symbol for G_5.

$$P_5 = P_1 \cup \{ S_5 \rightarrow S_1 S_5 / \varepsilon \}$$

It is clear from the grammar G_5 is context free and the language generated by this grammar is context free

$$L_5 = L_5^*$$

Hence it is proved.

CFLs are Not closed Under Interaction

Theorem: The CFL are not closed under intersection. If L_1 and L_2 are CFL, it is not always true that $L_1 \cap L_2$ is CFL

Proof: Let us prove this theorem by taking counter examples.

Consider two languages

$$L_1 = \{a^n b^n c^m / n \geq 0, m \geq 0\}$$

and

$$L_2 = \{a^n b^m c^m / n \geq 0, m \geq 0\}$$

The two languages are context free, the corresponding context free grammar is

$$S \rightarrow S_1 S_2$$
$$S_1 \rightarrow a S_1 b / \varepsilon$$
$$S_2 \rightarrow c S_2 / \varepsilon$$

and

$$S \rightarrow a S / S_1$$
$$S_1 \rightarrow b S_1 c / \varepsilon$$

Now let us take

$L_1 \cap L_2 = \{a^n b^n c^n / n \geq 0\}$

Using pumping lemma theorem we have already proved the language given is not CFL.

Thus we can prove that the family of CFL is not closed under intersection.

CFLs are Not closed Under Complementation

Theorem: The CFLs are not closed under complementation. If L is CFL, it is not true that complement of L is CFL

Proof: Let us prove this theorem by contradiction. If L_1 and L_2 are context free then $\overline{L_1}$ and $\overline{L_2}$ are also context free.

We have already proved that CFLs are closed under union.

So $\overline{L_1} \cup \overline{L_2}$

must be context free.

Since we have assumed that the CFLs are closed under complementation

$\overline{L_1} \cup \overline{L_2}$ must be context free.

But according to De-Morgan's Law

$\overline{L_1} \cup \overline{L_2} = L_1 \cap L_2$

So

$L_1 \cap L_2$ must be context free which is contradiction. Since CFL are not closed under intersection, our assumption that the CFLs are closed under complementation is not true.

So the family of context-free languages are not closed under complementation.

What is a Deterministic Context-Free Languages:

A language L is deterministic context free iff L can be accepted by some deterministic PDA.

Every deterministic context free language is context free.

Algorithms and Decision Procedures for CFL

A decision algorithm is an algorithm that computes the correct truth value for each of the i/p values of a decision problem. Alg has to terminate on all i/Ps. A decision problem is decidable if there exists a decision alg for it otherwise it is undecidable.

Decidable Questions:

Membership: Given a language L and a string w, is w in L?

To approaches to solves this problem:

* Use a grammar:

decideCFLusingGrammar(L:CFL, w:string) =

1) If L is specified as a PDA, use PDAtoCFG, to construct grammar G such that L(G) = L(M).
2) If L is specified as a grammar G, simply use G.
3) If w = ε then if S_G is nullable (remove ε production) then accept otherwise reject
4) If w ≠ ε

4.1) From G, construct G' such that $L(G') = L(G) - \{\varepsilon\}$ and G' is in CNF

4.2) If G derives w, it does in $2.|w|-1$ steps. Try all derivations in G of that number of steps. If one of them derives w, accept. Otherwise reject.

*) Use a PDA:

For every context free Language L, it is possible to build a PDA that accepts $L - \{\varepsilon\}$ and that has no ε transitions.

Then we show that every PDA with no ε transitions is guaranteed to halt.

Emptiness and Finiteness.

Theorem: Given a context free language L, there exists a decision procedure that answers each of the following questions.

1) Given a CFL L, is $L = \phi$?
2) Given a CFL L, is L infinite

Proof:

1) Let $G = (V, T, P, S)$ be a CFG that generates L. $L(G) = \phi$ iff S is unproductive (ie not able to generate any terminal strings). The algorithm is removeunproductive

(i.e remove useless symbols and productions from the grammar).

decideCFL empty(G : CFG) =

1) Let G' = removeunproductive(G)
2) If S is not present in G' then return True else return False.

2) Let G = (V, T, P, S) be a context free grammar that generates L. Let n be the number of non terminals in G. Let be be the branching factor of G. The longest string that G can generate without creating a parse tree with repeated non terminals along some path is of length b^n. If G generates no strings of length greater than b^n then L(G) is finite.

To test L is infinite invoke decideCFL(L, w) on all strings in Σ^* of length greater than b^n. If it returns True for any such string then L is infinite. If it returns false on all such strings the L is finite.

The undecidable Questions.

There exists no decision procedure for many other questions that we might like to be able to ask about CFL including.

*) Given a CFL, is $L = \Sigma^*$?

*) Given a CFL L, is the complement of L context free ?

*) Given a CFL L, is L regular ?

*) Given two context free languages L_1 and L_2 is $L_1 = L_2$?

*) Given two CFL's L_1 and L_2 is $L_1 \subseteq L_2$?

*) Given two CFLs L_1 and L_2 is, $L_1 \cap L_2 = \phi$?

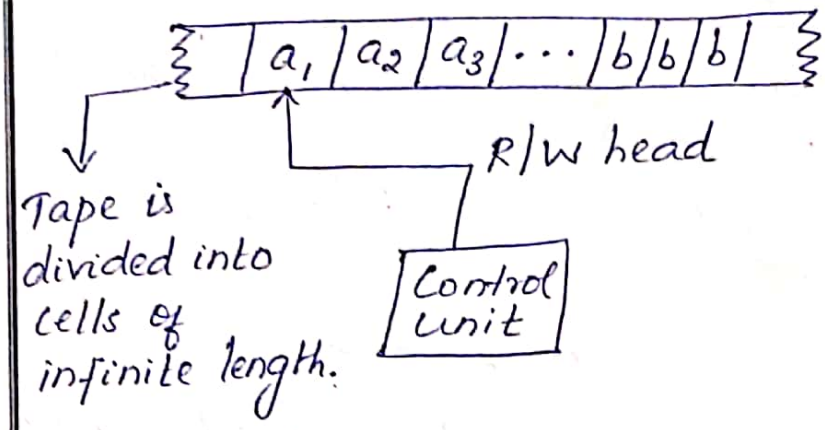
*) Given a context free language L, is L inherently ambiguous?

*) Given a CFG G, is G ambiguous ?

Turing Machine.

Turing Machine Model:

TM is a finite automaton connected to read write head with the following components.



Tape: Tape is used to store the information and is divided into cells. Each cell can store the information of only one symbol. The string to be scanned will be stored from the leftmost position on the tape. The string to be scanned should end with blanks. The tape is assumed to be infinite both on left and right side of the string.

Read-write head:

The read-write head can read a symbol from where it is pointing to and it can write into the tape to where it points to.

Control unit:

The reading from the tape or writing into the tape is determined by the control unit. The different moves

performed by the m/c depends on the current scanned symbol and the current state. The control unit consults action table & transition table and carry out the tasks.

The read_write head can move either towards left or right & movement can be on both the directions. The various actions performed by the m/c's are.

1) Change of state from one state to another state
2) The symbol pointing to by the read_write head can be replaced by another symbol.
3) The read_write head may move either towards left or right.

If there is no entry in the table for the current combination of symbol and state, then the m/c will halt.

The TM can be represented using various notations such that

*) Transition tables
*) Instantaneous descriptions.
*) Transition diagram

The TM $M = Q, \Sigma, \Gamma, \delta, q_0, B, F$ where

Q is set of finite states.

Σ is set of i/p alphabets

Γ is set of tape symbols

δ is transition function from

$Q \times \Gamma$ to $Q \times \Gamma \times D(L,R)$

q_0 is start state

B is Blank character

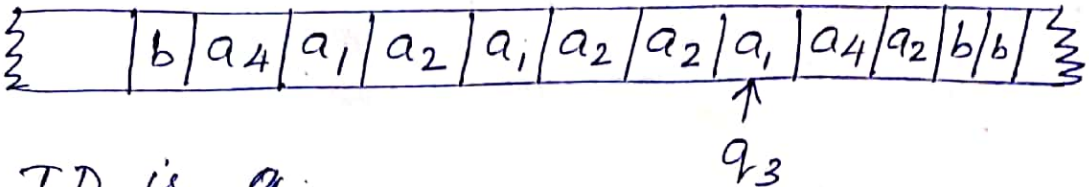
F is final state.

Representation of Turing Machine :

i) Representation by Instantaneous description.

An ID of a Turing Machine M is a string $\alpha\beta\gamma$, where β is the present state of M, the entire i/p string is split as $\alpha\gamma$, the first symbol of γ is the current symbol under the R/W head and γ has all the subsequent symbols of the i/p string, and the string α is the substring of the i/p string formed by all the symbols to the left of symbol pointed by R/W head.

(2) A snapshot of TM is shown in fig. Obtain the ID



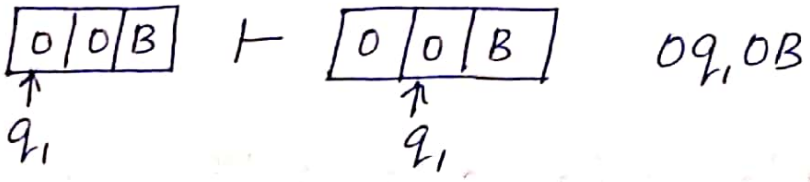
ID is

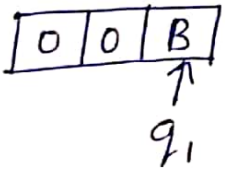
$a_4 a_1 a_2 a_1 a_2 a_2 q_3 a_1 a_4 a_2$

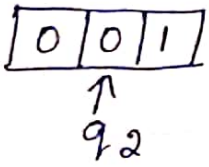
ii) Representation by Transition Table.

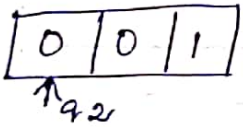
Consider the TM description given in Table 9.1. Draw the computation sequence of the i/p string 00.

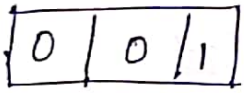
δ	Tape Symbols (Γ)		
States	B	0	1
→q_1	1Lq_2	0Rq_1	
q_2	BRq_3	0Lq_2	1Lq_2
q_3		bRq_4	bRq_5
q_4	0Rq_5	0Rq_4	1Rq_4
(q_5)	0Lq_2		



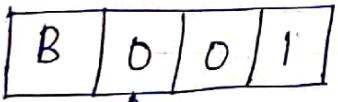


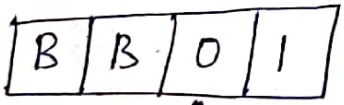


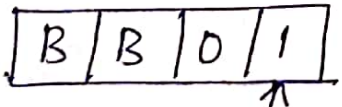


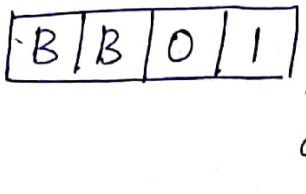


q_2

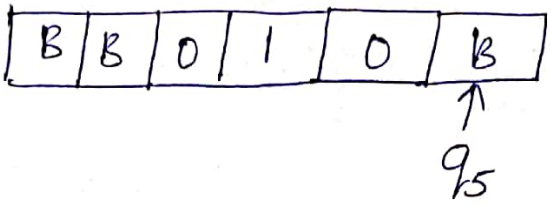




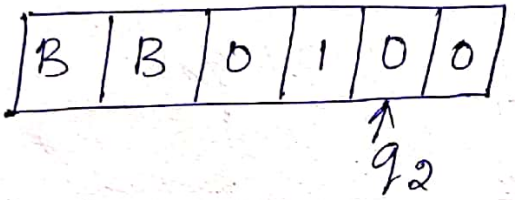




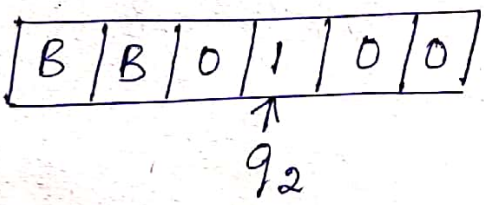
$BB01q_4B$



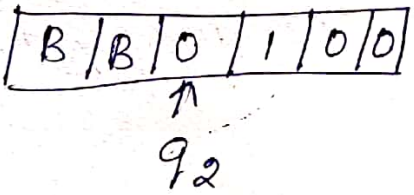
$BB010q_5$



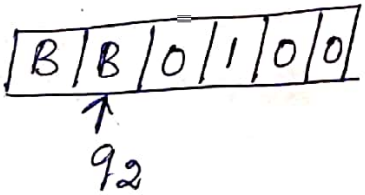
$BB01q_200$



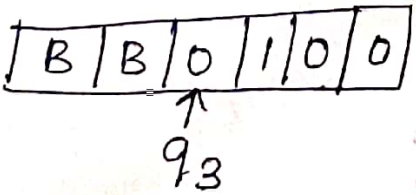
$BB0q_2100$



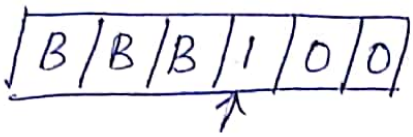
BBq_20100



Bq_2B0100

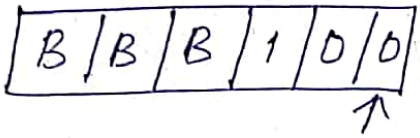


BBq_30100

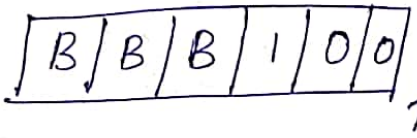


$BBBq_4100$

$BBB1q_400$

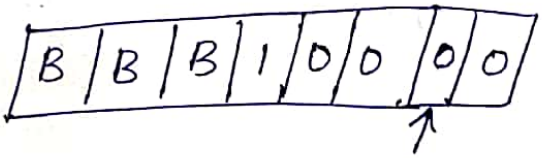


$BBB10q_40$

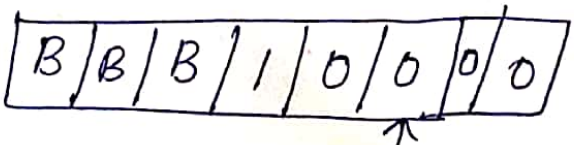


$BBB100q_4B$

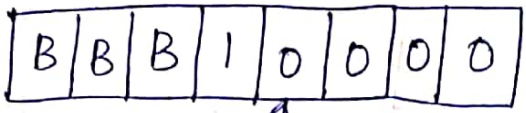
$BBB1000q_5B$



$BBB100q_200$



$BBB10q_2000$

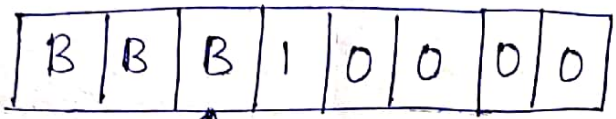


$BBB1q_20000$

q_2

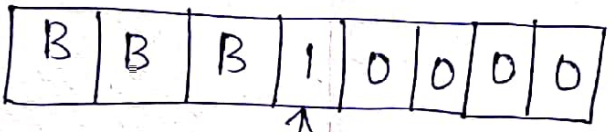
$BBBq_210000$

q_2



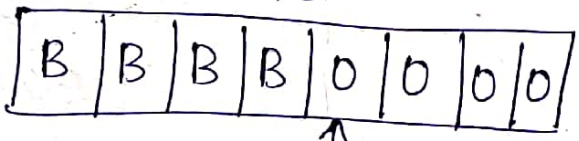
$BBq_2B10000$

q_2



$BBBq_310000$

q_3



$BBBBq_50000$

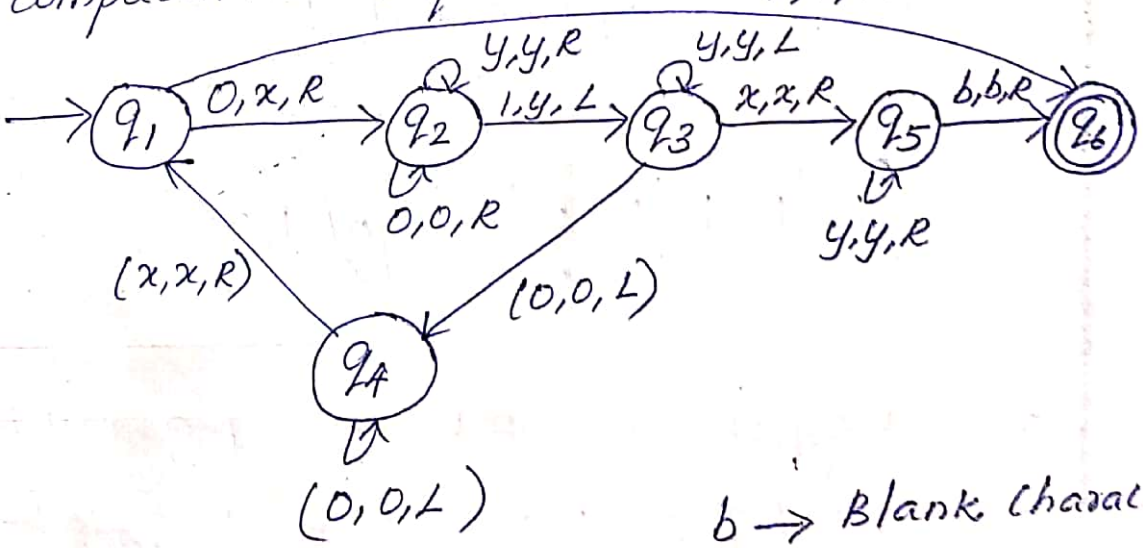
q_5

$q_100B \vdash 0q_10B \vdash 00q_1B \vdash 0q_201 \vdash q_2001 \vdash$
$q_2B001 \vdash Bq_3001 \vdash BBq_401 \vdash BB0q_41 \vdash$
$BB01q_4B \vdash BB010q_5 \vdash BB01q_200 \vdash$
$BBq_20100 \vdash BBq_20100 \vdash Bq_2B0100 \vdash$

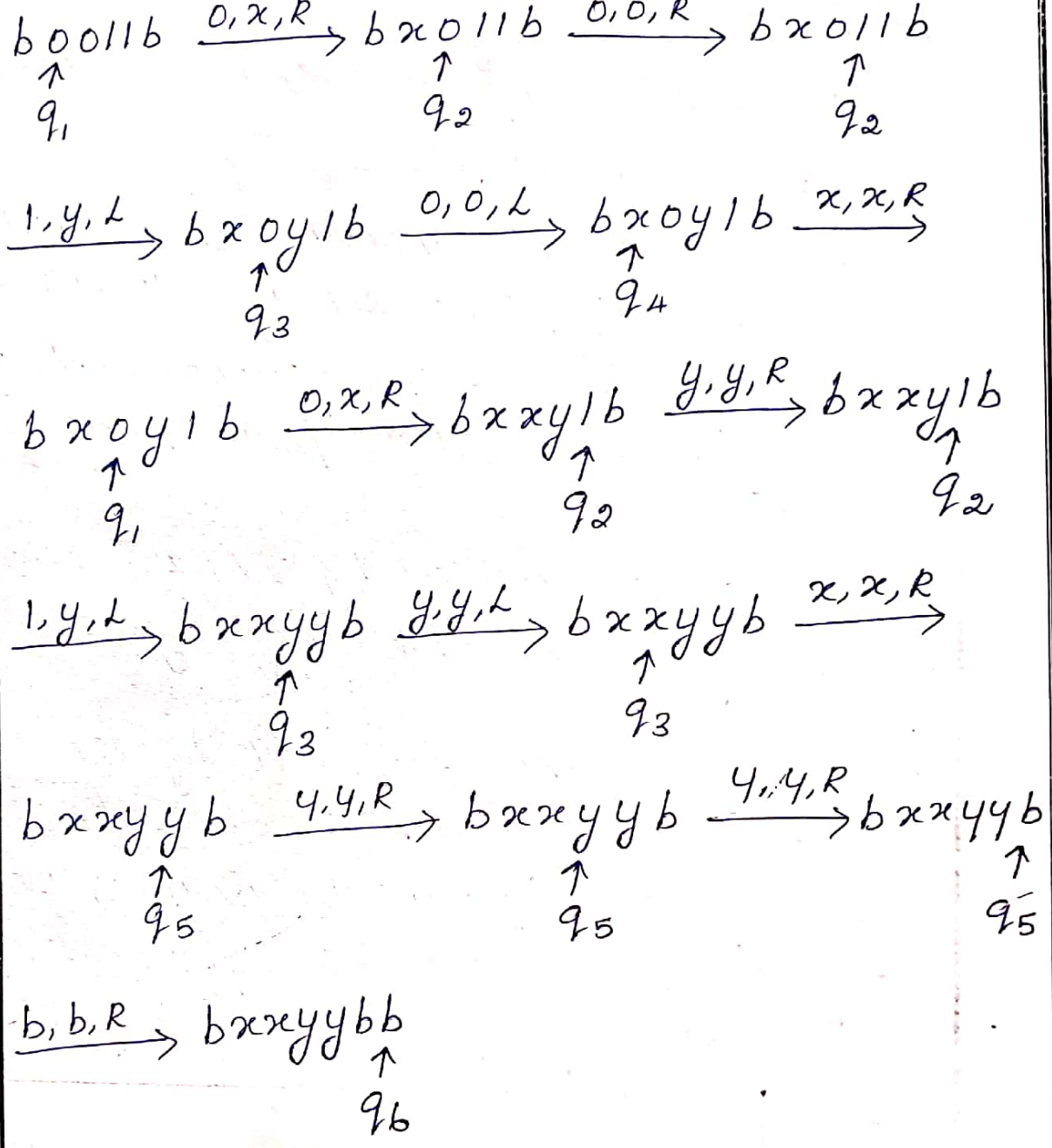
$BBq_30100 \vdash BBBq_4100 \vdash BBB1q_400 \vdash$
$BBB10q_40 \vdash BBB100q_4B \vdash BBB1000q_5B$
$\vdash BBB100q_200 \vdash BBB10q_2000 \vdash$
$BBB1q_20000 \vdash BBBq_210000 \vdash$
$BBq_2B10000 \vdash BBBq_310000 \vdash$
$BBBBq_50000$

iii) Representation by Transition Diagram

M is a turing machine represented by the transition system. Obtain the computation sequence.



b → Blank character



Language Acceptability by TM:

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ be a TM. The language $L(M)$ accepted

by M is defined as

$L(M) = \{ w / q_0 w \vdash^* \alpha_1 p \alpha_2$ where $w \in \Sigma^*$, $p \in F$ and $\alpha_1, \alpha_2 \in \Gamma^* \}$

where $q_0 w$ is the initial ID and $\alpha_1 p \alpha_2$ is the final ID. The set of all those words w in Σ^* which causes M to move from start state q_0 to final state p.

The language accepted by TM is called recursively enumerable language.

Consider the TM described by the transition table given. Describe the processing of a) 011 b) 0011 c) 001 using IDs. Which of the above strings are accepted by TM.

δ	Tape Symbols				
States	0	1	x	y	b (Blank)
→q_1	xRq_2				bRq_5
q_2	$0Rq_2$	$1Lq_3$		yRq_2	
q_3	$0Lq_4$		xRq_5	yLq_3	
q_4	$0Lq_4$		xRq_1		
q_5				$yxRq_5$	bRq_6

a) 0 1 1 q_1011

↑
q_1

$x\ 1\ 1$ xq_211

↑
q_2

$x\ y\ 1$ q_3xy1

↑
q_3

$x\ y\ 1$ xq_5y1

↑
q_5

$x\ y\ 1$ xyq_51

↑
q_5

$q_1011 \vdash xq_211 \vdash q_3xy1 \vdash xq_5y1 \vdash xyq_51$

b) 0 0 1 1 q_10011

↑
q_1

$x\ 0\ 1\ 1$ xq_2011

↑
q_2

$x\ 0\ 1\ 1$ $x0q_211$

↑
q_2

x 0 y 1 $x0yq_3 1$
↑ ↑
q_3 xq_30y1

x 0 y 1 q_4x0y1
↑
q_4

x 0 y 1 xq_10y1
↑
q_1

x x y 1 xxq_2y1
↑
q_2

x x y 1 $xxyq_21$
↑
q_2

x x y y xxq_3yy
↑
q_3

x x y y xq_3xyy
↑
q_3

$x\ x\ y\ y$ $\quad xxq_5yy$

$\uparrow$

q_5

$x\ x\ y\ y$ $\quad xxyq_5y$

$\uparrow$

q_5

$x\ x\ y\ y$ $\quad xxyyq_5b$

$\uparrow$

q_5

$x\ x\ y\ y\ b\ b$ $\quad xxyybq_6b$

$\uparrow$

q_6

i/p string 0011 is accepted.

c) $0\ 0\ 1$ $\quad q_1001$

$\uparrow$

q_1

$x\ 0\ 1$ $\quad xq_201$

$\uparrow$

q_2

$x\ 0\ 1$ $\quad x0q_21$

$\uparrow$

q_2

$x\ 0\ y$ (↑ q_3) $\quad x\ q_3\ 0\ y$

$x\ 0\ y$ (↑ q_4) $\quad q_4\ x\ 0\ y$

$x\ 0\ y$ (↑ q_1) $\quad x\ q_1\ 0\ y$

$x\ x\ y$ (↑ q_2) $\quad x\ x\ q_2\ y$

$x\ x\ y$ (↑ q_2) $\quad x\ x\ y\ q_2\ b$

M halts. 001 is not accepted by M.

Design of TM

Q) Design a TM that accepts $0^n 1^n / n \geq 1$

Consider this situation

X X 0 0 0 Y Y Y 1 1 1 (↑ q_0)

$\delta(q_0, 0) = q_1, X, R$

$\delta(q_1, 0) = q_1, 0, R$

$\delta(q_1, Y) = q_1, Y, R$

$\delta(q_1, 1) = q_2, Y, L$

$\delta(q_2, Y) = q_2, Y, L$

$\delta(q_2, 0) = q_2, 0, L$

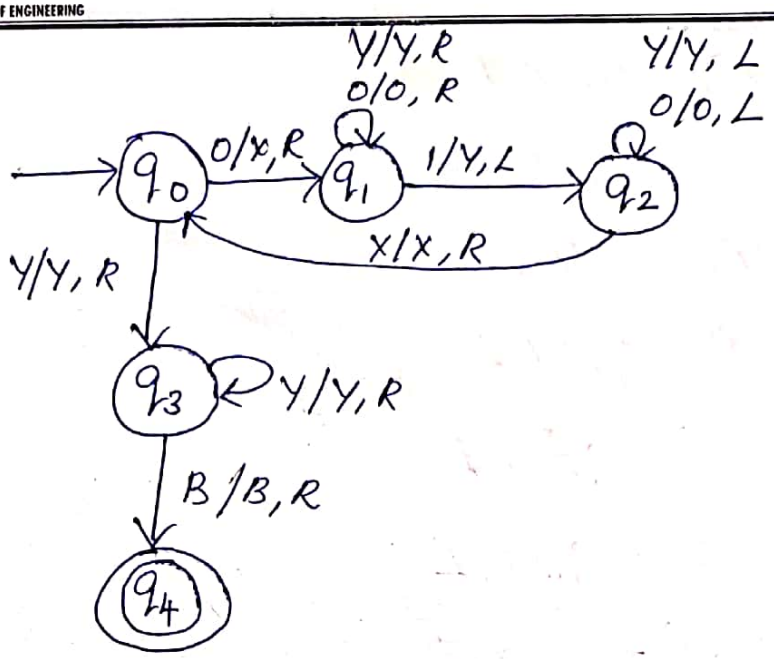
$\delta(q_2, X) = q_0, X, R$

$\delta(q_0, Y) = q_3, Y, R$

$\delta(q_3, Y) = q_3, Y, R$

$\delta(q_3, B) = q_4, B, R$

δ	Tape symbols				
States	0	1	X	Y	B
q_0	(q_1, X, R)			q_3, Y, R	
q_1	$q_1, 0, R$	q_2, Y, L		q_1, Y, R	
q_2	$q_2, 0, L$		q_0, X, R	q_2, Y, L	
q_3				q_3, Y, R	$q_4 B, R$
q_4					



Q2) Design a TM that accepts $L = \{0^n 1^n 2^n / n \geq 1\}$

Consider the situation XX00YY11ZZ22

$\delta(q_0, 0) = q_1, X, R$

$\delta(q_1, 0) = q_1, 0, R$

$\delta(q_1, Y) = q_1, Y, R$

$\delta(q_1, 1) = q_2, Y, R$

$\delta(q_2, 1) = q_2, Y, R$

$\delta(q_2, Z) = q_2, Z, R$

$\delta(q_2, 2) = q_3, Z, L$

$\delta(q_3, Z) = q_3, Z, L$

$\delta(q_3, 1) = q_3, 1, L$

$\delta(q_3, Y) = q_3, Y, L$

$\delta(q_3, 0) = q_3, 0, L$

$\delta(q_3, X) = q_0, X, R$

$\delta(q_0, Y) = q_4, Y, R$

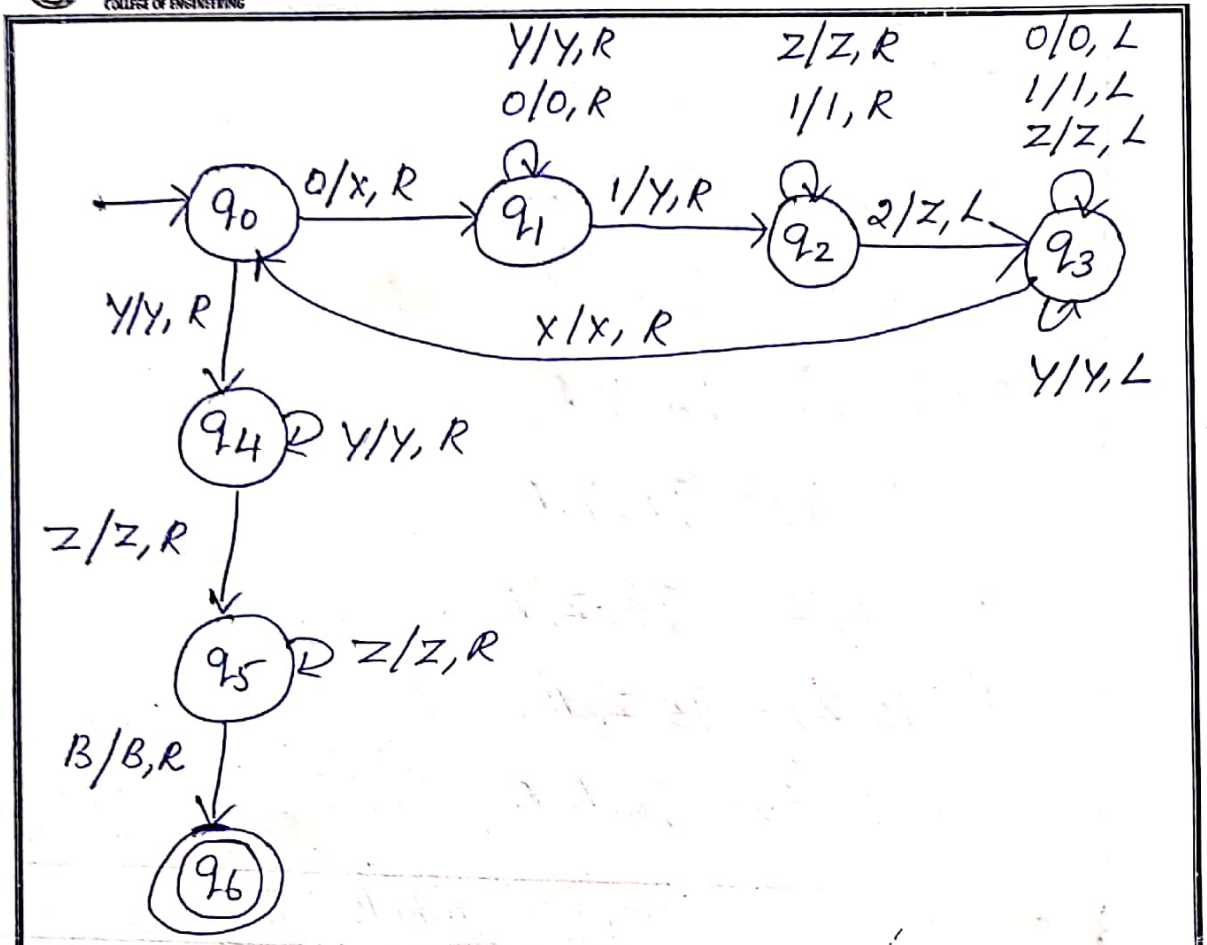
$\delta(q_4, Y) = q_4, Y, R$

$\delta(q_4, Z) = q_5, Z, R$

$\delta(q_5, Z) = q_5, Z, R$

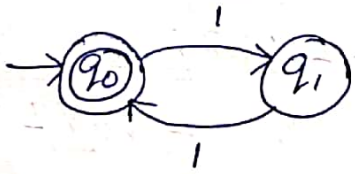
$\delta(q_5, B) = q_6, B, R$

δ	Tape Symbols Γ						
States	0	1	2	Z	Y	X	B
q_0	q_1, X, R				q_4, Y, R		-
q_1	$q_1, 0, R$	q_2, Y, R			q_1, Y, R		
q_2		$q_2, 1, R$	q_3, Z, L	q_2, Z, R			
q_3	$q_3, 0, L$	$q_3, 1, L$		q_3, Z, L	q_3, Y, L	q_0, X, R	
q_4				q_5, Z, R	q_4, Y, R		
q_5				q_5, Z, R			q_6, B, R
q_6							



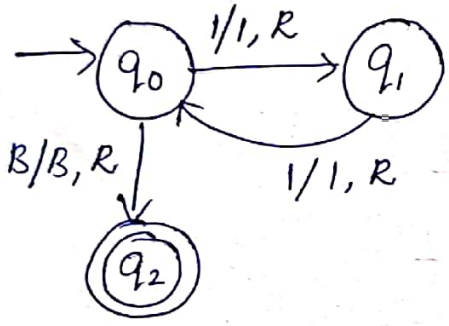
Q3) Design a TM to recognize all strings consisting of an even number of 1's.

The DFA to accept the given L is



δ	1
→ q0	q1
q1	q0

δ	Tape Symbol Γ	
States	1	B
$\overset{*}{\rightarrow} q_0$	$q_1, 1, R$	q_2, B, R
q_1	$q_0, 1, R$	–
q_2	–	–



Q4) Design a TM over {1, b} which can compute a concatenation function over Σ={1}. If a pair of words (w1, w2) is the input, the output has to be $w_1 w_2$

Steps:

a) Replace the separating symbol by 1

b) The rightmost 1 is found and replaced by a blank b.

c) The R/W head returns to the starting position.

$w_1 = 11$

$w_2 = 111$

ip: b 1 1 b 1 1 1 b

↑

q_0

$\delta(q_0, 1) = q_0, 1, R$

$\delta(q_0, b) = q_1, 1, R$

$\delta(q_1, 1) = q_1, 1, R$

$\delta(q_1, B) = q_2, B, L$

$\delta(q_2, 1) = q_3, B, L$

$\delta(q_3, 1) = q_3, 1, L$

$\delta(q_3, B) = q_4, B, R$

δ	Tape Symbols Γ	
states	1	b
$\rightarrow q_0$	$q_0, 1, R$	$q_1, 1, R$
q_1	$q_1, 1, R$	q_2, B, L
q_2	q_3, B, L	–
q_3	$q_3, 1, L$	q_4, B, R
(q_4)		

Techniques for TM Construction.

i) Turing machine with Stationary head.

We can include the option that the head can continue to be in the same cell for some input symbols. Then we can define

$$\delta(q, a) = q', y, S$$

means, that the TM on reading the i/p symbol a changes the state to q' and writes y in the current cell in place of a and continues to remain in the same cell. In terms of IDs.

$$wax \rightarrow wyx \quad (ie)\ wqax \vdash wq'yx$$

(q under a; q' under y)

This move can be simulated by the std. TM with two moves, namely

$$wqax \vdash wyq''x \vdash wq'yx$$

ie $\delta(q, a) = q', y, S$ is replaced by

$\delta(q, a) = q'', y, R$ and

$\delta(q'', x) = q', y, L$

ii) Storage in the state:

We can use a state to store a symbol as well. So the state becomes a pair (q,a) where q is the state and a is the tape symbol stored in (q,a). So the new set of states becomes $Q \times \Gamma$

eg: Construct a TM that accepts the language $01^* + 10^*$

solution: We have to construct a TM that remembers the first symbol and checks that it does not appear afterwards in the input string. So we require two states q_0, q_1. The tape symbols are 0, 1 and b. So the TM having 'storage facility in state' is

$M = (\{q_0, q_1\} \times \{0,1,b\}, \{0,1\}, \{0,1,b\}, \delta, [q_0, b], \{[q_0, b]\})$

Implementation description:

1) M is now in (q_0, a)
   i) If its next symbol is b, M enters

$[q_0, b]$ an accepting state.

ii) If next symbol is a, M halts without reaching the final state.

iii) If the next symbol is $\bar{a}$ ($\bar{a}=0$ if $a=1$) and ($\bar{a}=1$ if $a=0$) M moves right without changing state.

3) Step 2 is repeated until M reaches $[q_0, b]$ or halts.

iii) Multiple Track Turing Machine

In a multiple track turing m/c, a single tape is assumed to be divided into several tracks. Now the tape alphabet is required to consists of k-tuples of tape symbols, k being the number of tracks. The only difference b/w the Std. TM and TM with multiple tapes is tape symbols. In the former it is Γ and in the latter it is Γ^k.

iv) Subroutines.

We know Subroutines are used in computer languages, when some task has to be done repeatedly. We can

implement this facility for TM as well.

First, a TM program for the subroutine is written. This will have a initial state and return state. After reaching the return state, there is a temporary halt.

For using subroutine, new states are introduced. When there is a need for calling the subroutine, moves are effected to enter the initial state for the subroutine and return to main program of TM.

We can use this concept to design a TM for performing multiplication of two positive integers.

Design a TM which can multiply two positive integers.

The input (m,n), m,n being given, the positive integers are represented by 0^m10^n

At the end of the computation, 0^{mn} surrounded by b's is obtained.

Steps :

i) 0^m10^n1 is placed on the tape. Output will be written after the rightmost 1

ii) The leftmost 0 is erased.

iii) A block of n 0's is copied onto the right.

iv) Steps (ii) and (iii) are repeated m times and 10^m10^{mn} is obtained on the tape.

v) The prefix 10^m1 of 10^m10^{mn} is erased, leaving the product mn as O/P.

For every 0 in 0^m, 0^n is added on to the right end. This requires repetition of step 3. We define a subroutine called, COPY for step 3.

Transition table for Subroutine COPY.

δ	Tape Symbols Γ			
States	0	1	2	b
q_1	$q_2, 2, R$	$q_4, 1, L$		
q_2	$q_2, 0, R$	$q_2, 1, R$		$q_3, 0, L$
q_3	$q_3, 0, L$	$q_3, 1, L$	$q_1, 2, R$	
q_4		$q_5, 1, R$	$q_4, 0, L$	
q_5				

Transition Table of M

δ	Tape Symbols Γ			
states	0	1	2	b
q_0	q_6, b, R			
q_6	$q_6, 0, R$	$q_1, 1, R$		
q_5	$q_7, 0, L$			
q_7	–	$q_8, 1, L$		
q_8	$q_9, 0, L$			q_{10}, b, R
q_9	$q_9, 0, L$			q_0, b, R
q_{10}	–	q_{11}, b, R		
q_{11}	q_{11}, b, R	q_{12}, b, R		