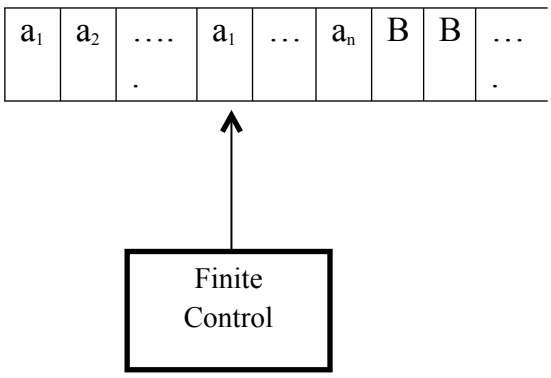
PART-B

❖ **Discuss the basic definitions of Turing Machine.**

The Turing Machine



★ The basic model of a TM has
1. a finite control
2. an infinite tape
3. a tape head

★ An input tape is divided into cells. The tape has a left most cell but it is infinite to the right.

★ Each cell of the tape may hold exactly one of a finite numbers of tape symbols.

★ Initially the n left most cells for n ≥ o hold the input which is string of symbols chosen from a subset of tape symbols called the input symbols.

★ The remaining infinity of cells each hold the blank, which is a special tape symbol that is not an input symbol.

★ In one move of the Turing Machine, depending upon the symbol scanned by the tape head and the state of the finite control
1. Changes state
2. Prints a symbol on the tape cell scanned replacing what was written there
3. Moves its tape head left or right one cell

Formal Definition:

★ Formally, a Turing Machine is donoted

$M = (Q, E, \angle, \delta, q_o, B, F)$

Where

$Q \rightarrow$ finite set of states

$\angle \rightarrow$ finite set of tape symbols

$B \rightarrow$ a symbol of $\angle$, is the blank

$E \rightarrow$ a subset of $\angle$ not including is the set of input symbols.

$\delta \rightarrow$ is the transition fuction mapping

$$Q \times \angle \rightarrow Q \times \angle \times \{L, R\}$$

$q_o \rightarrow$ a state in a Q is the initial state.

F $\rightarrow$ is the set of final states.

$F \leq Q$

Instantaneous Description : (ID)

★ ID of Turing Machine is denoted by

$\propto_1 q \propto_2$

Where

$q \rightarrow$ is the current state of M

$\propto_1 , \propto_2 \rightarrow$ is the string in $\angle^*$

Moves of the Turing Machine

★ A move of M is defined as follows Let $X_1 X_2 \ldots X_{1-1} q X_1 X_{1+1} \ldots X_n$ be an ID

suppose $\delta (q_1 X_i) = (P, \gamma, L)$

When we can write

$X_1 X_2 \ldots\ldots X_{1-1} q X_i \ldots X_n \perp X_1 X_2 \ldots X_{1-1} \gamma p X_{1+1} \ldots X_n$

Suppose $\delta (q_1 X_i) = (P, Y, R)$ then

$X_1 X_2 \ldots\ldots X_{1-1} q X_i \ldots X_n \perp X_1 X_2 \ldots X_{1-1} \gamma p X_{1+1} \ldots X_n$

The Language accepted by TM:

★ The language accepted by M, denoted L (M) is the set of words in Σ^x that cause M to enter a final state.

★ Formally the language accepted by TM.

M = (Q, E, $\angle$, δ, q_o, B, F) is

L (M) = {w | w in Σ^x and q o w $\perp \propto_1$ P $\propto_2$ for some P in F and $\propto_1$ and $\propto_2$

★ The TM halts, (e) has no next move, whenever the input is accepted.

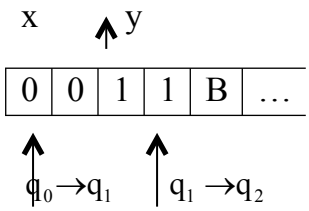
★ The TM will never halt for the words which are not accepted.

DESIGN OF TURING MACHINES

1. **Construct a Turing Machine for $L = \{ O^n 1^n \mid n \geq 1\}$**

Solution:

★ Initially the Turing Machine M contains $O^n 1^n$ followed by an infinity of Blanks. For eg.



M repeatedly replaces the leftmost O by and moves right to the leftmost L replacing it by Y. moves left to right to the left most X then moves one cell right to the leftmost O and repeats the cycle.

	0	1	X	Y	B
q_0	(q_1, X, R)			$(q_3, Y, R$	-
q_1	(q_1, O, R)	(q_2, Y, L)		(q_1, Y, R)	-
q_2	(q_2, O, L)	-	(q_0, X, R)	(q_2, Y, L)	-
q_3					$q_3, Y, R)$ $(q_4, B R)$
q_4	-	-	-	-	-

M = ({ q_0, q_1, q_2, q_3, q_4 }, { 0 , 1}, {0, 1 , X, Y, B}, $\delta, q_0, B, \{q_4\}$

To verify the string w = 0011

q_0 0011 $\perp$x q_1 011 $\perp$x o q_1 11 $\perp$

xq_2 0y1 $\perp$ q_2 x0yl $\perp$ xq_0 0y1 $\perp$

xxq_1 y1 $\perp$ xxyq_1 l $\perp$ xxq_2 yy $\perp$

xq_2 xyy $\perp$ xxq_0 yy $\perp$ xxyq_3 y $\perp$

xxyy$q_3\perp$ xxyybq_4

- The Turing Machine accepts L = { $O^n 1^n$ | n ≥ 1}

2) **Construct a Turing Machine for L = {ww^R | w $\in$ {0,1}**

Solution

	0	1	X	Y	B
q_0	(q_1, X, R)			(q_3, Y, R	-
q_1	(q_1, O, R)	(q_2, Y, L)		(q_1, Y, R)	-
q_2	(q_2, O, L)	-	(q_0, X, R)	(q_2, Y, L)	-
q_3					q_3, Y, R) (q_4, B R)
q_4	-	-	-	-	-
q_5		(q_6, Y, L)	(q_8, X, R)	(q_8, Y, R)	
q_6	(q_6, O, L)	(q_6, 1, L)	(q_0, X, R)	(q_0, Y, R)	
q_7					
q_8					

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

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$

$\Sigma = \{0,1)$

$\Gamma = \{0,1, x, y, B\}$

Let consider w = 0011

q_0 0011 $\perp$ x q_1 110 $\perp$ x 1 q_1 10 $\perp$

x11$q_1$0 $\perp$ x110q_1B $\perp$ x11q_2 0 $\perp$

x1$q_3$1x $\perp$ x$q_3$11x $\perp$ q_3x11x $\perp$

x$q_0$11x $\perp$ xy$q_4$1x$\perp$ xy1q_4x$\perp$

xy$q_5$1x$\perp$ x$q_6$yyx

COMPUTABLE LANGUAGES AND FUNCTIONS

- **Explain Turing Machine as a computer of Integer functions with an example.**

COMPUTABLE LANGUAGES AND FUNCTIONS

- A Language that is accepted by a Turing Machine is said to be recursively enumerable
- The term enumerable derives from the fact that these languages whose strings can be enumerated by a turing machine.
- The class of recursively enumerable languages very broad and it includes the CFL's The class of recursively enumerable languages includes some languages for which we cannot determine membership. ie) It is not accepted by Turing machine.
- If L(M) is such a language, then any TM, recognizing L(M) must fail to half on some input not in L(M).
- M halts on input w.
- The subclass of the recursively enumerable called the recursive sts which ar those language accepted by any TM that halts on all inputs.

THE TURING MACHINE AS A COMPUTER OF INTEGER FUNCTION

- In addition to language acceptor, the TM may be viewed as a computer of function from integers to integers.
- The integer $1 \geq 0$ is represented by the string O^i
- If a function has K arguments $i_1, i_2, \ldots i_k$, then these integers are initially placed on the tape separated by 1's as $O^{i_1} | O^{i_2} | O^{i_k}$
- If the TM halts with a tape consisting of O^m, then we say that $f(i_1, i_2, \ldots i_k) = m$, where f is the function of k arguments. computed by this Turing machine.
- If Turing machine M computes function of k arguments then f need not have a value for all different k tuples.
- If f $(i_1, i_2, \ldots i_k)$ is defined for all $i_1, i_2, \ldots i_k$, then we say f is a total recursive function.
- A function f $(i_1, i_2, \ldots i_k)$ computed by a TM is called partial recursive function.

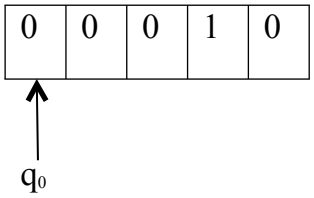
3.Construct a Turing machine to perform addition

Solution

Suppose the input is $O^m | O^n$

Finally the TM halts on tape containing O^{m+n}

Suppose w = 00010



0 0 0 0 B

q_1 q_1

0 0 0 B B

q_3

	0	1	B
q_0	(q_o, O, R)	(q_1, O, R)	
q_1	(q_1, O, R)		(q_2, B, L)
q_2	(q_3, B, R)	-	(q_3, B, R)
q_3	-	-	-

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, \{q_3\}$

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

$\Sigma = \{0,1)$

$\Gamma = \{0,1, B\}$

$F = \{q_3\}$

Simulating for w = 00010

$q_0\,00010 \perp 0q_1\,0010 \perp 00q_0\,010 \perp$

$000q_0 10 \perp 0000q_1 0 \perp 00000q_1\,B \perp$

$0000q_2 BB \perp 0000Bq_3$

Hence Turing machine performs addition

4) Construct a Turing Machine to perform proper subtraction

Solution:

Proper subtraction M-n is defined as

$$M - n = \begin{cases} M - n & \text{for } M \geq n \\ O & \text{for } M < n \end{cases}$$

- ✶ The TM started with O^m, O^n on its tape and halts on O^{m-n} on its tape.
- ✶ M repeatedly replaces its leading O by blank then searches light for a 1 followed by a O and changes the O to 1.
- ✶ Next M moves left until it encounters a blank and then repeats the cycle. The repetition ends if
    1. Searching right for a O, M encounters a blank. Then the n O's in $O^m \mid O^n$ have all been changed to 1's and n + 1 of the m O's have been changed to B. M replaces the n + 1 1's by a O and n B's leaving m-n O's on its tape.
    2. Beginning the cycle M cannot find a O to change to a blank, because the first m O's already have been changed. Then n ≥ m, so m – n = O. m replaces all remaining 1's and O's by B.

	0	1	B
q_0	(q_o, B, R)	(q_5, B, R)	
q_1	(q_1, O, R)	$(q_2, 1, R)$	
q_2	$(q_3, 1, L)$	$(q_2, 1, R)$	(q_4, B, L)
q_3	(q_3, O, L)	$(q_3, 1, L)$	(q_o, B, R)
q_4	(q_4, O, L)	(q_4, B, L)	(q_6, O, R)
q_5	(q_5, B, R)	(q_5, B, R)	(q_6, B, R)
q_6	--	--	--

M is defined as

$M = \{Q, \Sigma, \angle, \delta, q_o, B, F\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$

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

$\angle = \{0, 1, B\}$

$F = \{q_6\}$

A sample computation of M on input 0010 is

q0 0010 |-- Bq1 o10 |-- $BOq_1$10|-- $BO1q_2O$

$BOq_3$11 |-- Bq_3o11 |-- q_3BO11 |-- Bq_0 011

$BBq_1$11 |--$BB1q_2$1 |-- $BB11q_2$

$BB1q_4$1B|-- $BBq_4$1BB |-- Bq_4BBBB

BOq_6BB..

PROGRAMMING TECHNIQUES FOR TURING MACHINE

❖ **Explain the programming techniques for Turing Machine Construction.**

To describe the complicated TM constructions, some techniques are used,

- The techniques are
    1. Storage in the finite control
    2. Multiple tracks
    3. Checking of symbols
    4. Shifting over
    5. Subroutines

1. Storage in the finite control

- The finite control can be used to hold a finite amount of information.
- The state is written as pair of elements
    a. Exercising control
    b. Storing a symbol

Example:

Consider a TM 'M' that looks at the first symbol. Records it in its finite control and checks that the symbol does not appear elsewhere on its input.

Solution

M = (Q,

Where Q is

Ie., Q consists of the pairs

[q0, 0], [q0, 1], [q0, B]

[q0, 0], [q1, 1], [q1, B]

The set F= {[q1,B]}

- The first component of the state controls the action, while the second component 'remembers' a symbol.
    1) =([q1,0],0,R),
    2) =([q1,1],1,R)

  - Initially q0 is the control component of the state and M moves right. The first component of state becomes q1 and the first symbol seen is stored in the second component.
    1) =([q1,0],1,R),
    2) =([q1,1],0,R)
  - If M has a O stored and sees a 1, or vice versa, then M continues to move to the right.

    1) =([q1,B],0,L),
    2) =([q1,B],0,L)

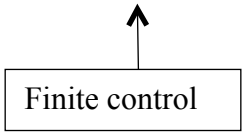
- M enters the final state [q1,B] if it reaches a blank symbol without having first encountered a second copy of the left most symbol.

✶ If M reaches a blank in state (q_1, 0) or (q_1, 1) it accepts.
✶ For state (9,10) and symbol 0 or for state (q_1 , 1) and symbol 1, δ is not defined.
✶ Thus if M encounters the tape symbol stored in its state. M halts without accepting.

2. **Multiple tracks:**

✶ The tape of the Turing machine is divided into K tracks for any finite K.
✶ The symbols on the tape are considered k tuples one component for each track

Ȼ	1	0	1	1	1	1	$	B	...
B	B	B	B	1	0	1	B	B	
B	1	0	0	1	0	1	B	B	



Example:

- ✶ The tape of a TM takes a binary input greatest than 2, written on the first track and determined whether it is a prime.

Solution:

- ✶ The input is surrounded by C and $ on the first tracks.
- ✶ Thus the allowable input symbols are

  [C, B, B] → is identified with C

  [O, B, B] → is identified with O

  [1, B, B] → is identified with 1

  [$, B, B] → is identified with $

  [B, B, B] → is identified with B

✶ To test it its input is prime, the Turing Machine

→ Write the number two in binary on the second track.

→ Copies the first tracks on the third

→ The second track is subtracted as many times as possible from the third track, and

leaving the remainders.

→ If the remainder is zero, the number on the first track is not a prime.

3. Checking off symbols.

- ✶ Checking off symbols is useful for visualizing how a Turing Machine recognizes languages defined by repeated strings such as

  $\{ w w \mid w \text{ in } \Sigma^* \}$

  $\{w c y \mid w \text{ and } y \text{ in } \Sigma^*, w \neq y\}$

  $\{w w^R \mid w \text{ in } \Sigma^*\}$

✶ It is also useful when lengths of substrings must be compared, such as in the languages

$\{a^i b^i \mid \geq 1\}$

$\{ a^i b^j c^k 1 \; i \neq j, \text{ or } j \neq k\}$

- ✶ We introduce an extra track on the tape that holds a blank or ✓
- ✶ The ✓ appears when the symbol below it has been considered by the Turing machine in one of its comparisons.

Example:

Consider a Turing Machine M ($Q, \Sigma, \angle, q_0, B, F$)

which recognizes the langues L = {w c w | w in (a + b)

Solution:

Let Q = {[q, d] | q = q_1, q_2 …. q_q

d = a, b or B}

- ✶ The second component of the state is used store an input symbol
  Σ= { [B, d] | d = a, b or c
- ✶ The input symbol [B, d] is identified with d

Γ = [x d] | x = B or and d = 0

q_0 = [q_1, B] and F = {[q_9, B[}

δ is defined as

1) δ ([q_1, B], [B, d] = ([q_2, d], [, d], R)
2) δ ([q_2, d], [B, e]) = ([q_2, d], [B, e], R)
3) δ ([q_2, d], [B, c]) = ([q_3, d], [B, c], R)
4) δ ([q_3, d], [, e]) = ([q_3, d], [, e], R)
5) δ ([q_3, d], [B, d]) = ([q_4, B], [, d], L)
6) δ ([q_4, B], [, d]) = ([q_4, B], [, d], L)
7) δ ([q_4, B], [B, c]) = ([q_5, B], [B, c], L)
8) δ ([q_4, B], [B, c]) = ([q_5, B], [B, c], L)
9) δ ([q_6, B], [B, d]) = ([q_6, B], [B, d], L)
10) δ ([q_6, B], [, d]) = ([q_1, B], [, d], R)
11) δ ([q_5, B], [, d]) = ([q_7, B], [, d], R)
12) δ ([q_7, B], [B, c]) = ([q_8, B], [B, c], R)
13) δ ([q_8, B], [, d]) = ([q_8, B], [, d], R)
14) δ ([q_8, B], [B, B]) = ([q_4, B], [, B], L)

Shifting Over:

- ✶ A Turing machine can make space on its tape by shifting all non – blank symbols a finite number of cells to the right.
- ✶ The tape head moves to the right, repeatedly store the symbols read in the finite control
- ✶ The symbols are replaced by the symbols read from cells to the left.
- ✶ The TM can return to the vacated cells and print symbols of its choosing.

Example :

Construct a Turing machine M = {Q, Σ, ∠, δ, q_o, B, F} which shift non blank symbols two cells to the right.

Solution :

- Let the tape of M does not contain blanks between non – blanks.
- Let Q contains the states of the form $[q,A_a,A_2]$ for $q = q_1$ of q_2A_1,A_2 in Γ
- Let x be a special symbol not used buy M except in shifting process
- M starts the shifting process in state $[q_1,B,B]$
- δ are defined as
  1) $\delta([q_1, B, B], A1 = ([q_1, B, A_1], x, R)$ for A_1 in $\Gamma - \{B, x\}$
  2) $\delta([q_1, B, A_1], A_2 = ([q_1, A_1, A_2], x, R)$ for A_1 and A_2 in $\Gamma - \{B, x\}$
  3) $\delta([q_1, B, A_1], A_3 = ([q_1, A_1, A_3], A_1, R)$ for A_1, A_2, A_3 in $\Gamma - \{B, x\}$
  4) $\delta([q_1, A_1, A_2], B = ([q_1, A_2, B], A_1, R)$ for A_1, and A_2 in $\Gamma - \{B, x\}$
  5) $\delta([q_1, A_1, B], B = ([q_2, B, B], A_1, L)$
  6) $\delta([q_1, A_1, A_2], B = ([q_1, A_2, B], A_1, R)$ for A_1, and A_2 in $\Gamma - \{B, x\}$

Subroutines :

- The Turing machine can simulate any type of subroutine, include recursive procedure and any of the parameter passing mechanisms.
- We describe only the use of parameter less, non-recursive sub routines.
- The general idea is
  - To write part of a TM program to serve as a subroutine.
  - It will have a designated initial state and return state which has no move and which will be used to effect a return to the calling routine.
  - To design a Tm that calls the subroutine a new set for states for the subroutines is mad, and a move from the return state is specified.
  - The call is effected by entering the initial state for the subroutine and the return is effected by the move from the return state.

Example :

The design of a TM to implement the total recursive function

Solution :

- M starts with $O^m \mid O^n$ on its tape and ends with O^{mn} surrounded by blanks.
- The general idea is to place a 1 after $O^m \mid O^n$ and then copy the block of n O's onto the right end m time each time erasing one of the O's.
- The δ is defined as for copy sub routine as

	0	1	2	B
q_1	$(q_2, 2, R)$	$(q_4, 1, L)$		
q_2	(q_2, O, R)	$(q_2, 1, R)$		(q_3, O, L)
q_3	(q_3, O, L)	$(q_3, 1, L)$	$(q_1, 2, R)$	

q_4		$(q_5, 1, R)$	$(q_4, 1, L)$

- ✶ To complete the program for multiplication we add states to convert initial ID $q^0 O^m \mid O^n$ to B O^{m-1},q,O^n 1.

i.e

$\delta\ (q_0, O), = (q_6, B, R)$

$\delta\ (q_6, O), = (q_6, O, R)$

$\delta\ (q_6, 1), = (q_1, 1, R)$

- ✶ The δ is defined Multiplication as

	0	1	2	B
q_5	(q_7, O, L)			
q_7		$(q_8, 1, L)$		
q_8	(q_9, O, L)			(q_{10}, B, R)
q_9	(q_9, O, L)			(q_{10}, B, R)
q_{10}		(q_{11}, B, R)		
q_{11}	(q_{11}, B, R)	(q_{12}, B, R)		

Simulating for w = 00100

q_0 00100 $\perp B q_6 0100 \perp$ $B0q_6100 \perp B01q_100$

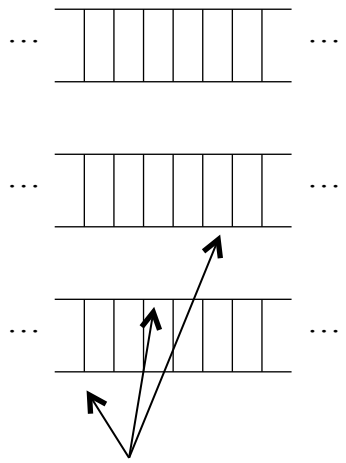
$B012q_20 \perp$ $B012q_2 B \perp B012q300$ ….

MULTIHEAD AND MULTITAPE TURING MACHINE

- ❖ **Discuss about the Multihead and Multi Tape Turing Machine.**

Multihead and Multi tape turning machine :

- A multitape turning machine consists of a finite control with K tape heads and k taps.
- Each tape is infinite in both directions.
- On a single move depending on the state of the finite control and the symbol scanned by each of the tape heads, the machine can
  1) Change state
  2) Print a new symbol scanned by each tape heads
  3) Move each of tape heads independent one cell to the left or right or keep it state.
- Initially the input appears on the blank.



Finite Control

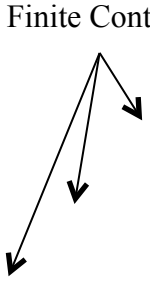
Theorem :

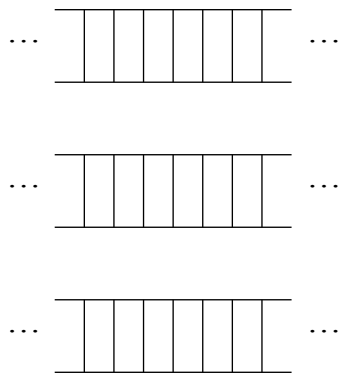
It a language L is accepted by a multitape turning machine. It is accepted by a single tape turning machine.

Proof :

- Let L be accepted by M, a TM with k tapes
- We can construct M_2 a single tape TM with 2k tracks
- One track records the contents of the corresponding tape of M.
- The other tape is Blank, except for a Marker that holds the symbol scanned by the corresponding head of M.

A three tape turning machine





- The finite control of M_2 stores the state of M1 along with a count of number of head markers to the right of M_2's tape head.

Head 1		x				
Tape 1	A_1	A_2	…		..	A_m
Head 2				x		
Tape 2	B_1	B_2	…		…	B_m
Head 3	x					
Tape 3	C_1	C_2	…		…	C_m

Simulation of three tapes by one

- Each move of M_1 is simulated by a sweep from left to right and then from right to left by the tape head of M_2
- Initally M2's head is at the left most cell containing a head marker.
- To simulate a move of M_1
  $\rightarrow M_2$ sweeps right visiting each of the cells with head markers and recording the symbol scanned by each head of M_1
  $\rightarrow$When M_2 crosses a head marker it must update the count of head markers to its right
  $\rightarrow$When no more head markers are to the right M_2 has seen the symbols scanned by each of M_1's heads.

- ✶ Now M_2 makes a left until it reaches the left most pass it updates the tape symbol of M_1
- ✶ Finally M_2 changes the state of M_1 recorded in M_2's control to complete the simulation of one move of M_1
- ✶ If new state of M_1 is accepting then M_2 accepts move of M_1
- ✶ It takes $2k^2$ moves of M_2 to simulate k moves of M_1

CHOMSKIAN HIERARCHY OF LANGUAGES

❖ **Describe the chomskian hierarchy of Languages.**

The Chomskian hierarcy of languages :

- ✶ The four classes of languages
  1. Regular sets
  2. Context free languages
  3. Context sensitive languages
  4. Recursive and recursively enumerable languages
- ✶ The four classes of languages are grammatically charecterised.
- ✶ These four classes of languages are called as chomshy hierarchy.

1. Regular Grammar :

- ✶ It all productions of a CFG are of the form

  $A \to wB$ where $A, B \to$ are variables

  $A \to w$ where $w \to$ are string of terminals

  then we say the grammar is light lineave.
- ✶ If all productions are of the form

  $A \to Bw$

  $A \to B$

  then we say the grammar is left lineave.
- ✶ A right or left linear grammar is called as regular grammar.
- ✶ Regular grammar derives regular languages.
- ✶ Regular languages are accepted by finite Automata.

2. Unrestricted Grammar :

- ✶ A grammar is used with 4 tube notation.

  G = (V, T, P, S)
- ✶ The productions are of the form

  $\alpha \to \beta$

where, $\alpha, \beta \rightarrow$ are arbitray string of grammar symbols with $\alpha \neq \in$

- These grammar also known as type 0 grammar or phase structure grammar.
- These are accepted by turning machines.

3. Central sensitive languages :

- Suppose we place the restriction on the production $\alpha \rightarrow \beta$ of a phase structure grammer that β be atleast as long as α.
- The resulting grammar is context sensitive grammar.
- These grammars are called as type 1 grammers
- These grammar derives context sensitive language
- Context sensitive languages are accepted by linear bounded Automata.

Linear Bounded Automata (LBA)

- A linear bounded automata is a non-deterministic turning machine satisfying the following two conditions.
  1. Its input alphabet includes two special symbols α and $, the left and right end marker respectively.
  2. The LBA has no moves left from or right from $, nor may it print another symbol over or $.
- The LBA is denoted as
  $M = (Q, \Sigma, \Gamma, \delta, q0, \varsigma, \$, F)$
  Where $Q, \Sigma, \Gamma, \delta, q_0,$ and F are as for a Non-deterministic TM
  $\varsigma, \$, \rightarrow$ are symbols in Σ the left and right end markers.
- The language accepted by M is
  $L(M) = \{w \mid w \text{ is in } (\Sigma - \{\varsigma, \$\})^* \text{ and } q_0 \varsigma w \$ \alpha q \beta \text{ for some } q \text{ in } F$

4. Context free Grammar :

- The CFG is defined as 4 tuples
  G = (V, T, P, S)
  Where productions are of the form
  $A \rightarrow \alpha$
  Where $A \rightarrow$ is not-terminal
  $\alpha \rightarrow$ is a string of terminals
- These grammars are type 2 grammars.
- These grammars derives context free language (CFLS)
- The context free languages are accepted push down automata.
-

HALTING PROBLEM

❖ **Explain Halting Problem. Is it solvable or unsolvable problem? Discuss.**

The Halting Problem :

✶ If the problem of whether a TM M accept the string w is unsolvable.

→It cannot be solved for recursively enumerable languages.

→This approach will procedure an answer only if T halts not if it loops forever.

Accepts : Given a TM, M and a string w, Does, TM, M accepts w?

✶ An instance of the problem accepts consists of a pair <M, w> with the universal turning machine.

Lu = {<m', w'> | w $\in$ L (M) }

✶ The halting problem is related to the membership problem for recursively enumerable languages.

Halting Problem :

✶ For a given turning machine M and a given string w, instead of asuling whether turning machine accepts w. it asks whether turning machine halts on input w. This problem is referred as halting problem.

Halts : Given a turning machine M, and a string w does M halts on input w?

✶ We can consider the corresponding language

$L(M) = \{< M^1, W^1 > | M$ halts on input w }

Reducing One Decision Problem to another :

✶ If P1 and P2 are decision problems. P1 is reducible P2 ie) written as

$P1 \leq P2$

If there is an algorithmic procedure that allows us

"Given an arbitrary instance I of P_1 to find an instance F (I) of P_2. So that for every I the answer for the two instances I and F (I) are the same.

Reducing One language to another :

✶ If L1 and L_2 are languages over alphabet Σ_1 and Σ_2 respectively, we say that L1 reducible to L_2 ie) written as

✶ This type of reducibility is called as many one reduciability

Theorem :

The halting problem is undecidable.

Proof :

✶ We show that Accept can be reduced to 'Halts'

✶ For both problems, an instance is a pair consisting of turning machine, M and a string w.

✶ Given a pari <m, w> an instance of Accepts we must construct another pair <m_1, w_1> an instance of Halts.

✶ This means that M_1 and w_1 must be constructed so that M_1 halts on input w_1 iff M accepts w.

i.e) If M accepts w, m_1 must halt on w_1 and if M either rejects w or runs forever on w, m, must not halt on w.

- Let us choose w = w_1
- To construct M1, the moves of M are simulated by changing the move of the form

$\delta(p,a)=(h_r,b,D)$ to

$\delta(p,a)=(p,a,s)$

- i.e) If M ever arrives in state p with a on the current tape, M1 is stuch in this state and the symbol for ever.
- M1 will begin by inserting a new symbol # in the left most cell, moving everything else over one cell.
- Then q_0 is moved with the tape head to next cell ie.) second cell
- M_1 has the same moves as M, as well the additional move

$\delta(q,\#)=(q,\#,s)$ for all possible.

States q.

- If M ever moves its tape head off, M, will enter an infinite loop, with its tape head on the left most cell. Ie) It never halts.
- Thus the hulting problem is undecidable.