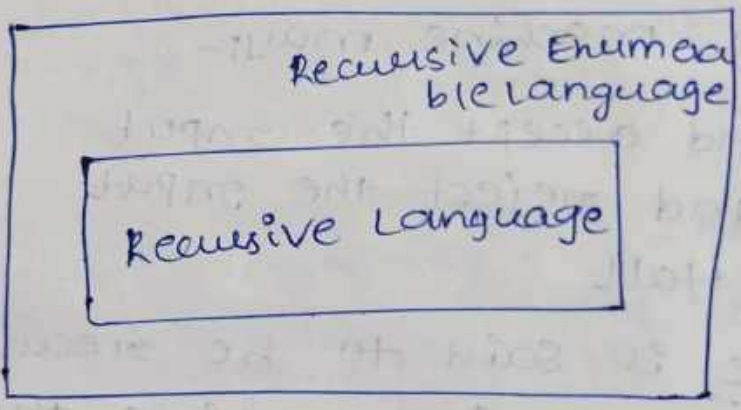
⇒ A language is said to Recursively enumerable if there exist a turing machine that accepts every string of the language, and does not accept strings that are not on the language.

⇒ For the strings that are not on the language the turing machine may (or) may not halt.

Note:-

Every recursive language is also recursive enumerable language. It is not obvious whether every recursively enumerable language is also recursive



(imp)

Types of turing Machine:-

⇒ There are no. of other types of turing machines in addition to one we have seen, such as turing machine with multiple tapes, one tape but with multiple heads, two dimensional, non-deterministic turing machine etc---

⇒ It turns out that computationally all these turing machines are equally powerfull. that

is what one type can compute. however the type can also compute. however the efficiency of competition that:show fast they can compute may vary.

ii) Non-deterministic turing machine:-

⇒ A non-deterministic turing machine is a machine for which like non-deterministic finite Automata, at any current state and for the tape symbol it is reading, there may be different possible actions to be performed.

⇒ Here, An action means a combination of writing a symbol on the tape, moving the tape head and going to next state

example:-

$L = \{ww \mid w \in (a+b)^*\}$

⇒ Given a string x, a non-deterministic turing machine that accepts the language L would first guess the midpoint of x, which is ~~where~~ the place where the second half of x starts. It must find the mid point by ~~playing~~ pairing of x two heaendy symbols from of x

⇒ Formally a non-deterministic turing machine transaction function takes

$Q \times \Gamma \times \{L, R\}$

$Q \times \Gamma \longrightarrow 2$

(ii) Turing machine with 2-dimensional tape:-

⇒ It is a kind of turing machine that is one finite control, one read-writehead, one 2D tape.

⇒ these cells on the tape is 2D, that is the tape has the top end and the left end but extends indefinitely to the right and down.

⇒ It is divided into rows of small squares. For any TM of this space type there is an equivalent TM with one D tape that is equally powerfull.

⇒ To simulate a new 2D tape with 1D tape first we map the squares of 2D tape to those of 1D tape diagonally as shown in the following table

2-D Tape:- v & h are end points

v	v	v	v	v	v
h	1	2	6	7	5
h	3	5	8	4	
h	4	9	13		
h	10	12			
h	14				

1-D Tape:-

v	1	v	2	3	h	4	5	6	v	7	8	9	10	h	...

Transition function:-

$Q \times T \longrightarrow Q \times T \times \{L, R, T, B\}$

(Left, Right, TOP, Bottom)

⇒ some turing machine with one-D tape can simulate every move of turing machine with a 2-D tape. Hence they are as powerfull as turing machine with atleast 2-D tape. since, TM with 2D tape obviously can simulate TM with 1-D tape, it can be said that they are equally powerfull

(iii) TM with multiple tapes:-

⇒ This kind of TM has one finite control and more than one tape each with its own read/write head

⇒ It is denoted by a 7-tuple

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

* It's transition function is a partial function

$Q \times T^n \longrightarrow (Q \times \{h\}) \times T^n \times \{L, R\}^n$

⇒ A configuration for this kind of TM must show the current state the machine is in and the state of each tape. It can be proved that any language accepted by a n-tape TM can be accepted by a one-tape TM and that any function computed by n-tape TM can be computed by a one-tape TM

Since the converses are obviously by a one-tape TM. Since the converses are obviously true, one can say that one-tape TM. Since the converses are obviously true, one can say that one-tape TMs are as powerful as n-tape TM's.

* Turing Machines with multiple heads:-

→ This kind of TMs has one finite control and one tape, but more than one read/write heads. In each state only one of the heads is allowed to read and write

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

→ the transition function is a partial function

$\delta: Q \times \{H_1, H_2, \ldots, H_n\} \times \Gamma \rightarrow \{Q \times \{h\}) \times \Gamma \times \{L, R\}$

where $H_1, H_2, \ldots, H_n$ denote the tape heads.

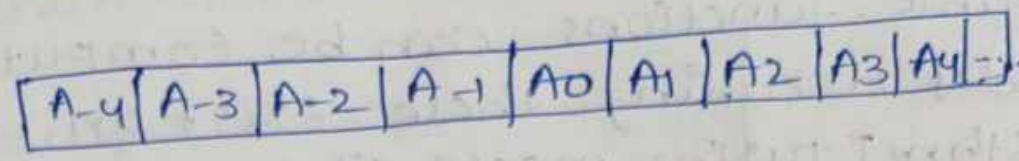
→ It can be easily seen that this type of TMs are as powerful as one-tape TMs.

→ Turing machines with Infinite tapes:-

→ This is a kind of TM that have one finite control and one tape which extends infinitely on Both directions

→ It turns out that this type of TMs are also as powerfull as one-tape TMs whose tape has left end.

* Two-way infinite tapes:-



Computable functions:-

In the theory of computation (TOC) A computable function is a mathematical function that can be computed using a turing machine (or) any other Equivalent model of computation.

Definition

A function $f: \Sigma^* \rightarrow \Sigma^*$ where Σ is finite Alphabet is computable if there exist a turing machine M such that

(i) for every input $x \in \Sigma^*$, M halts that stops computing on x

(ii) if M halts on x the output of M on x is f(x)

Properties:-

(i) Partiality:-

Computable functions can be partial there may not be defined for all inputs

(ii) Determinism:-

Computable functions are deterministic, meaning their output depends only on input.

Effectiveness:-

computable functions can be computed Effectively, meaning there exist an algorithm [turing machine] to compute them

Examples of computing functions:-

→ Arithmetic operations [+, -, *, /]
→ logical operations [AND, OR, NOT]
→ String manipulation [concatenation, substring extraction]
→ Sorting algorithms [Bubble sort, Merge sort]
→ Searching algorithms [linear search, Binary search]

Non-computable functions:-

→ Halting problem [determining whether turing machine halts on a Given input]

→ decision problem for first halt logic [Determining whether Given sentence is true on halt models]

→ Word problem for finitely presented groups [Determining whether 2 words represent the same group element]

closure properties of recursive & recursively Enumerable languages:-

(i) Recursive grammar closed under union
(ii) " " " intersection
(iii) set difference
(iv) complement
(v) intersection with a regular language
(vi) union with a regular language
(vii) concatenation
(viii) kleen closure (A^*)
(ix) kleen plus (A^+)
(x) Reversal
(xi) ∈free homomorphism
(xii) inverse homomorphism
(xiii) ∈free substitution
(xiv) left difference with regular language
(xv) Right difference with regular language
(xvi) left Quotient with regular language
(xvii) Right Quotient with regular language.

Not closed under:-

(i) Homomorphism
(ii) Substitution
(iii) Subset.

Recursive Enumerable grammars:-

Closed under:-

(i) Union
(ii) Intersection
(iii) Intersection with regular languages.
(iv) Union with regular languages
(v) Concatenation
(vi) Kleen* (A^*)
(vii) Kleen+ (A^+)
(viii) Reversal
(ix) Homorphism, ϵ-free Homomorphism, Inverse Homomorphism
(x) ϵ-substitution
(xi) Substitution
(xii) left difference with regular language
(xiii) left and right position with regular languages

Not closed under:-

(i) Set difference
(ii) compliment
(iii) Subset
(iv) Right difference with regular languages.