UNIT

①

Recursive Enumerable Langua ges:-

nguage is recursively enumerable if some Turing Machine accepts it.

For string ω

if ω ∈ L then M halts in a final state.

if ω ∉ L then M halts in a nonfinal state. (or) loop forever.

Recursive Languages:-

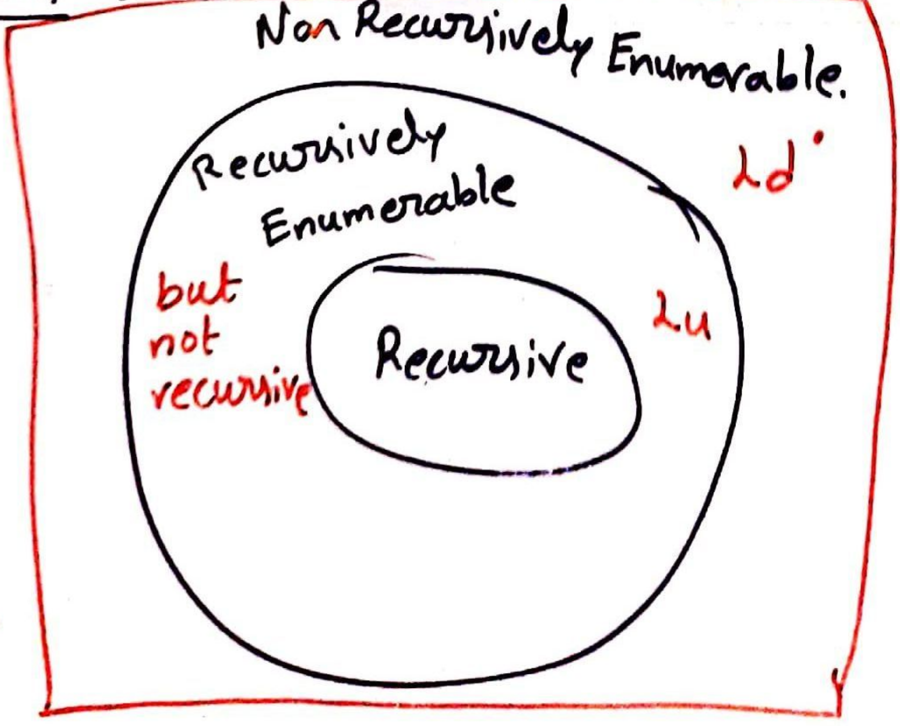
A language is recursive if some Turing Machine accepts it and halts on any input string. Let L be a recursive language and M the Turing Machine that accepts it.

For string ω:

if ω ∈ L then M halts in a final state.

if ω ∉ L then M halts in a non-final state.

Relationship among three classes:-



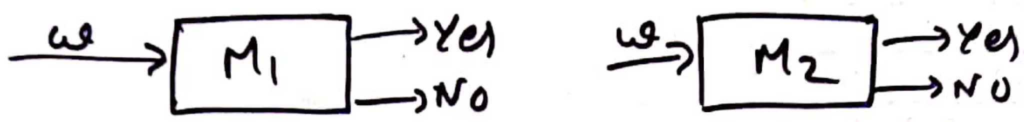
Properties of recursive languages:-

Theorem :- 1

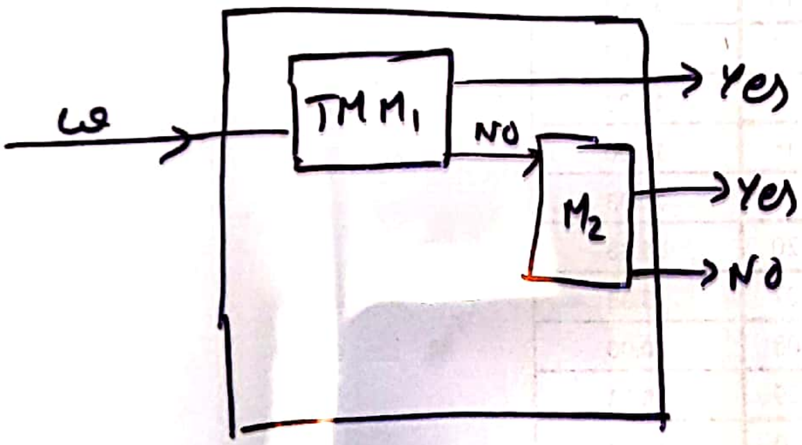
If L_1 & L_2 are two recursive languages then $L_1 \cup L_2$ is also recursive language.

Proof :- L_1 is accepted by TM M_1

L_2 is accepted by TM M_2

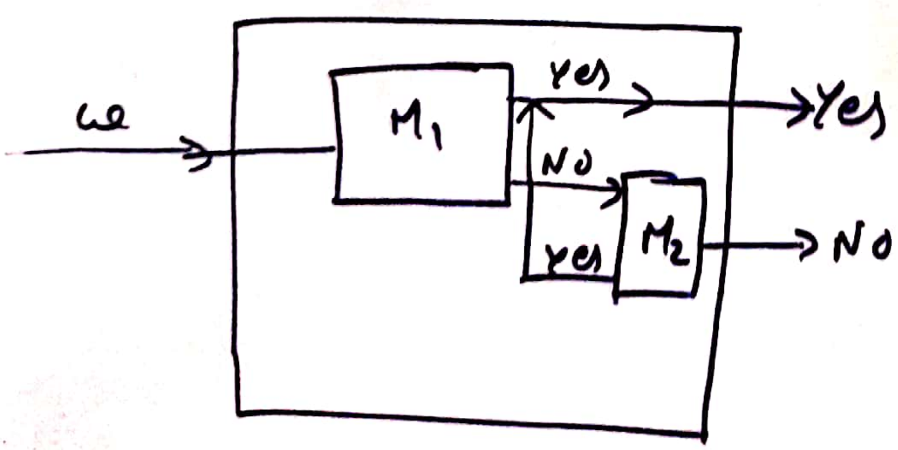


The compound TM M that accepts $L_1 \cup L_2$ is given below.



Theorem :- 2 :-

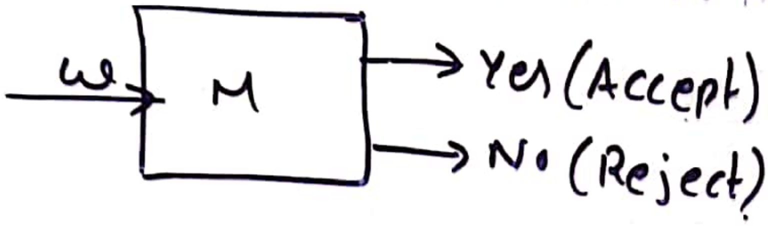
If L_1 & L_2 are two recusive languages then $L_1 \cap L_2$ is also recursive languages.



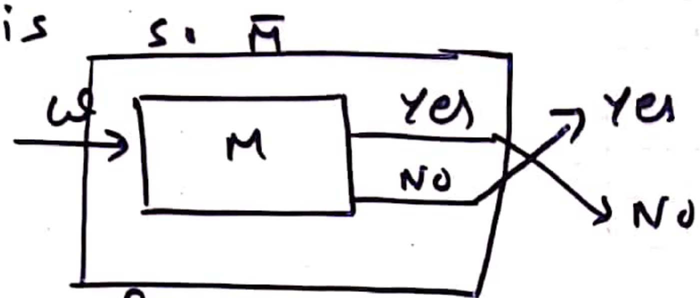
Complement

If L is a recursive language, so is $\bar{L}$

Proof: $L = L(M)$ for some TM M that always halts



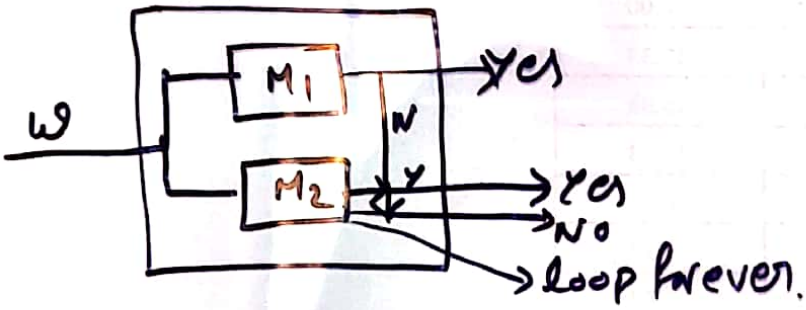
$\bar{L}$ is so $\bar{M}$



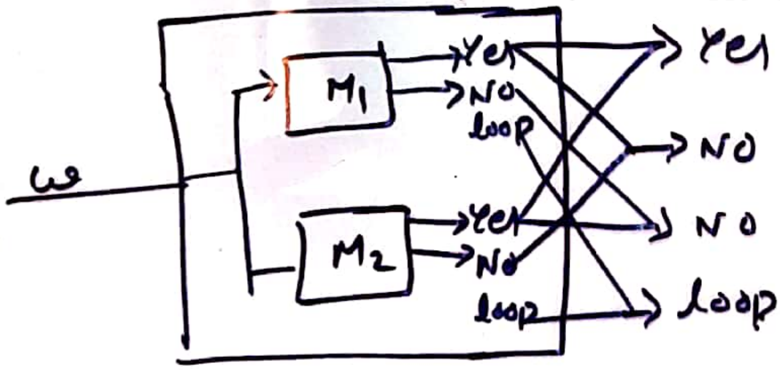
Properties of Recursive Enumerable languages:

If L_1 & L_2 are two recursively enumerable languages then $L_1 \cup L_2$ is also recursively enumerable.

Proof:



$L_1 \cap L_2$

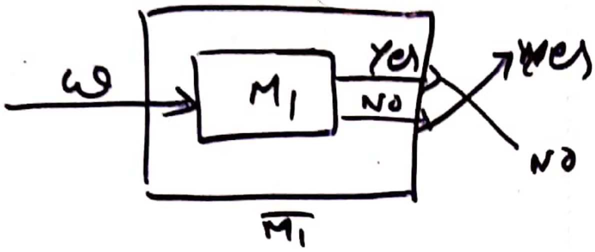


Theorem :- 3

If L is RE $\overline{L}$ may not be RE

$L = \{m \text{ halts on } x\}$

$\overline{L} = \{m$ ~~halts~~ ~~does~~ not halts on $x\}$



There is no point of loop so it may not RE

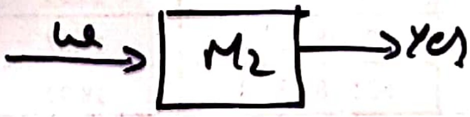
Theorem :- 4

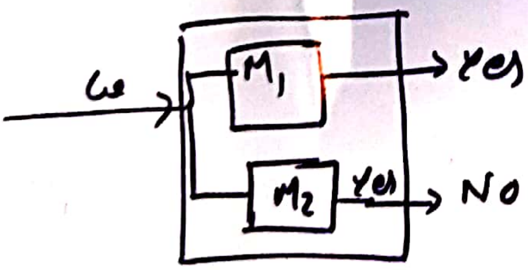
If L and $\overline{L}$ are recursively enumerable, then L is recursive.

Proof :-

L is RE accepted by M_1

$\overline{L}$ is RE accepted by M_2





If input w to M is in L then M_1 will accept. If so M accepts & halts.

If w is not is in L. then it is in $\overline{L}$, So M_2 will accept. when M_2 accepts, M halts without accepting. That means on all inputs M halts. Since M halts and $L(M) = L$ we conclude that L is recursive.

A Language that is not Recursively Enumerable:

Inorder to prove that a language that is not RE, follow the steps

1) Codes for Turing Machine.
2) Diagonalization Language.
3) L_d is not Recursively Enumerable.

1) Codes for TM:

If w is the binary string, treat $1w$ as a binary integer i.
Then we shall call w the ith string.

ϵ - first string
0 - second
1 - Third
00 - Fourth
01 - fifth.

To represent a TM $M = \{Q, \{0,1\}, \Gamma, \delta, q_1, B, F\}$ as a binary string, we must first assign integers to the states, tape sym, & directions L & R.

- We shall assume the states are $q_1, q_2 \cdots q_k$ for some k. The start state will be q_1 & q_2 will be the only accepting state.
- We shall assume tape symbols $X_1 X_2 \cdots X_m$ for some m.

$X_1 \rightarrow 0$
$X_2 \rightarrow 1$
$X_3 \rightarrow B$

Other tape symbols can be assigned to the remaining integers.

- We shall refer to direction L as D_1 & direction R as D_2

$L \rightarrow 0$ $\quad$ $R \rightarrow 00$

$$\delta(q_i, X_j) = (q_k, X_l, D_m)$$

We shall code as $0^i 1 0^j 1 0^k 1 0^l 1 0^m$.

A code for the entire TM M Consists of all the codes for the transitions in some order, seperated by pairs of 1's

$$C_1 11 C_2 11 \text{---} C_{n-1} 11 C_n$$

where each of the c's is the code for one transition of M.

Ex:- $M = (\{q_1, q_2, q_3, q_4, q_5, q_6\}, \{0,1\}, \{0,1,B\}, \delta, q_0, B, \{q_2\})$

where δ Consists of the rules.

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

$\delta(q_6, 1) \Longrightarrow (q_6, 1, R)$

$\delta(q_6, B) = (q_3, B, L)$

$\delta(q_3, 1) = (q_4, B, L)$

$\delta(q_4, 1) = (q_3, B, L)$

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

$\delta(q_5, B) = (q_2, 1, L)$

Tape Symbols $\begin{cases} 0 - 0 \\ 1 - 00 \\ B = X_3 \ (000) \end{cases}$

$R = D_2 \ (00)$

$L = D_1 \ (0)$

The codes for the seven transitions may be listed in any order, giving us 5040 codes. for M.

Diagonalization Language:-

The language L_d, the diagonalization language, is the set of strings w_i such that w_i is not in $L(M_i)$

L_d Consists of all strings w such that the TM M does not accept w as input

We can construct a table for all i & j in which TM M_i accepts input string w_j;

1 means "Yes it does"

0 means "No it doesn't".

$i \downarrow$ \ $j \rightarrow$	1	2	3	4	...
1	0	1	1	1	...
2	1	1	0	0	...
3	0	0	1	1	...
4	0	1	0	1	...
⋮	⋮	⋮	⋮	⋮	⋱

To construct L_d, we complement the diagonal, From the figure diagonal values are 0 1 1 1

Complemented Values 1 0 0 0

The complementented diagonal value "1 0 0 0" will not fallen any where in the table. That mean this is not accepted by TM which mean it is Not Recursive Enumera

It works because the complement of the diagonal is itself a characteristic vector describing membership in some language L_d.

The complement of the diagonal cannot be the characteristic vector of any TM.

Universal Language

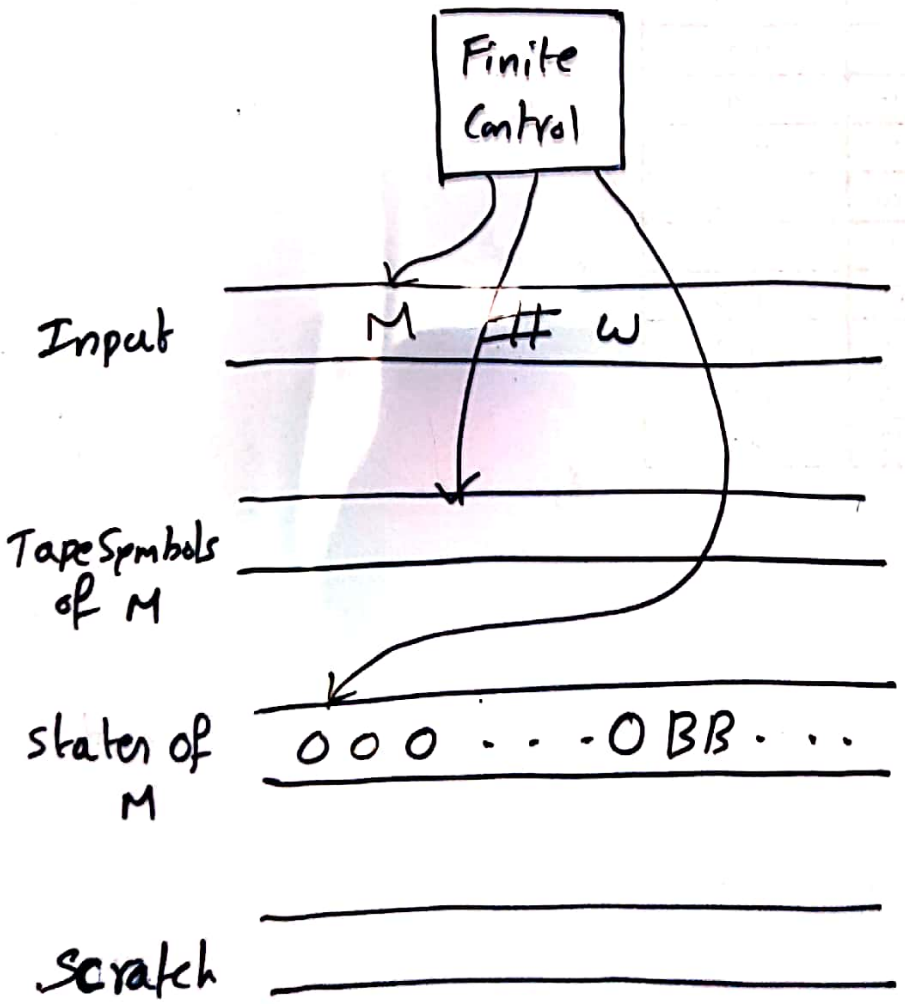
An Undecidable Problem That is RE

Definition :-

L_u is the set of strings representing a TM and an input accepted by that TM. There is a TM U, often called the universal Turing Machine, such that $L_u = L(U)$.

It is easiest to describe U as a multitape TM.

1. First tape holds $\langle M, w \rangle$, M → TM & w is the input string
2. Second tape holds tape symbols of M.
3. Third tape holds the state of M



The operation U can be summarized as follows.

1) Examine the input to make sure that the code for M is legitimate code for some TM. If not, U halts without accepting.
2) Intialize the second tape to contain the input ω, in its encoded form.

zero (0) $\rightarrow$ 10
One (1) $\rightarrow$ 100
Blank (B) $\rightarrow$ 1000

3) Place 0, the start state of M, on the third tape, and move the head of U's second tape to the first simulated cell.
4) To simulate a move of M, U searches on its first tape for a transition $0^i 1 0^j 1 0^k 1 0^l 1 0^m \Rightarrow (q_i, j) = (q_k, l, m)$ such that 0^i is the state placed in tape3.
0^j is the tape symbol placed in tape2.

The transition can be done by U as follows.

a) Change the contents of tape3 to 0^k. i.e state change
b) Replace 0^j on tape 2 by 0^l i.e Modify tape symbol.
c) Move the head on tape 2 to the position L (or) R

5) If M has no transitions that matches the simulated state & tape symbol then in (4) no transition will be found. Thus M halts in the simulated configuration. & U must do like wise.
6) If M enters its accepting state, then U accepts.

In this manner, U simulates M on ω. U accepts the coded pair (M, ω) if and only if M accepts ω.

Halting Problem of TM :-

It is stated as follows : "Given a TM in an arbitary configuration will it eventually halt?". This problem is said to be undecidable. in the seme that there can not exist an algorithm which will take as input a description of a TM M and an input ω and say whether M on ω will halt (or) not.

$$HALT_{TM} = \{(M, \omega) \mid \text{The TM } M \text{ halts on input } \omega\} \text{ is undecidable.}$$

Church Turing Thesis :-

"No computational procedure will be considered an algorithm unless it can be represented as a TM"

(or)

If there is no TM that decides problem P, there is no algorithm that solves problem P.

(or)

The set of languages that can be decided by a TM is identical to the set of languages that can be decided by any mathematical computing machine

RICE THEOREM

Theorem:- Every nontrival property of the RE languages is undecidable.

Proof:-

A property of the RE languages is simply set of RE languages i.e The property of being contextfree is formally the set of all CFL's

The property being empty is the set $\{\emptyset\}$ consisting of only the empty language.

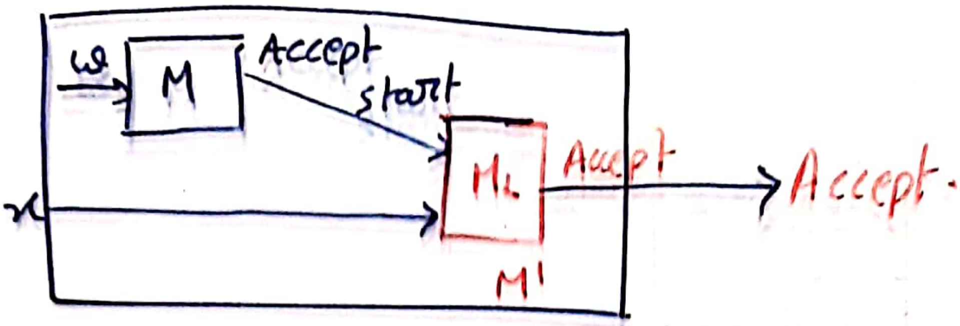
→ A property is trival if it either empty or is all RE languages. otherwise it is nontrival.

→ let P be a nontrival property of RE languages. P is nontrival, there must be some non empty lang. L that is in P. Let M_L be a TM accepting L.

→ We shall reduce L_u to L_p, thus proving that L_p is undecidable, since L_u is undecidable.

→ The algorithm to perform the reduction take as input a pair (M, w) and produces a TM M'.

→ $L(M')$ is $\emptyset$ if M doesnot accept w.
$L(M')$ is L if M accepts w.



1. M' is a two tape TM. One tape is used to simulate M on w.
2. The simulation of M on w is built into M';
2. The other tape of M' is used to simulate M_L on input x to M', if necessary.

Turing Machine M' is constructed to do the following.

1. Simulate M on input w. Note that w is not the input to M'; rather M' writes M and w onto one of its tapes & simulates the universal TM. U on that pair.

2. If M does not accept w then M' does nothing else M' never accepts its [illegible] input x, so $L(M') = \emptyset$

3. If accepts w, then M' begins simulating M_L on its own input x. Thus M' will accepts exactly the language L, since L is in P, the code for M' is in L_p.