UNIVERSAL TURING MACHINE:

Turing Thesis: TM is as powerful as a computer

Computer: can run any program

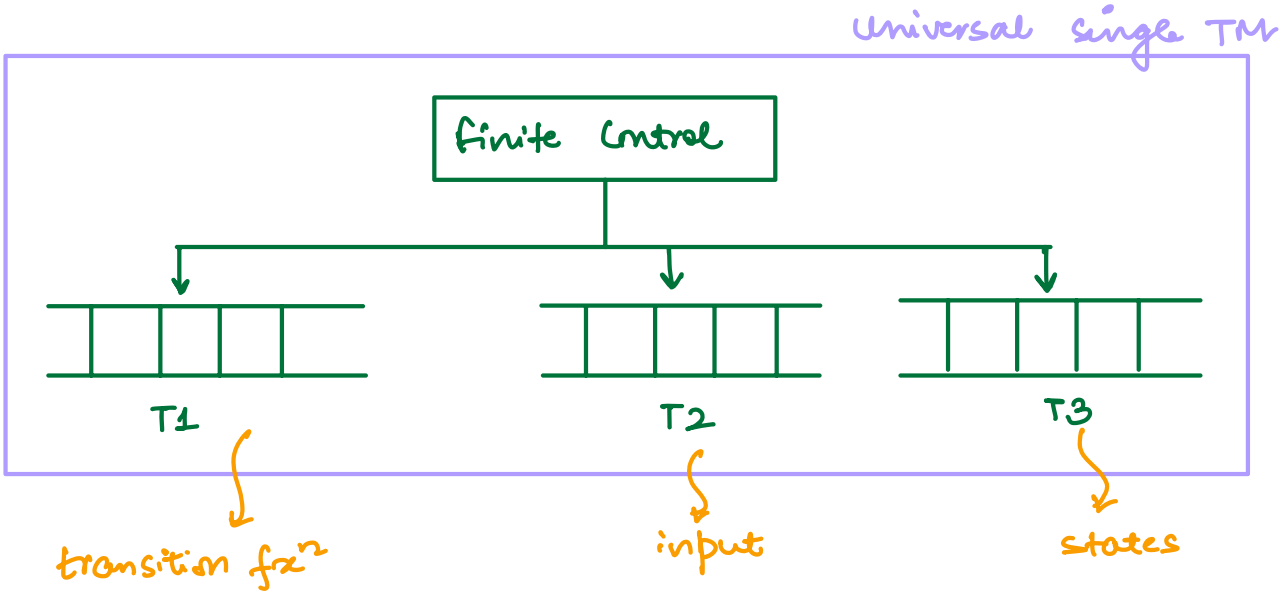
- (Addition) +
- (Multiplication) *
- (Divisn) /

TM works for a single problem

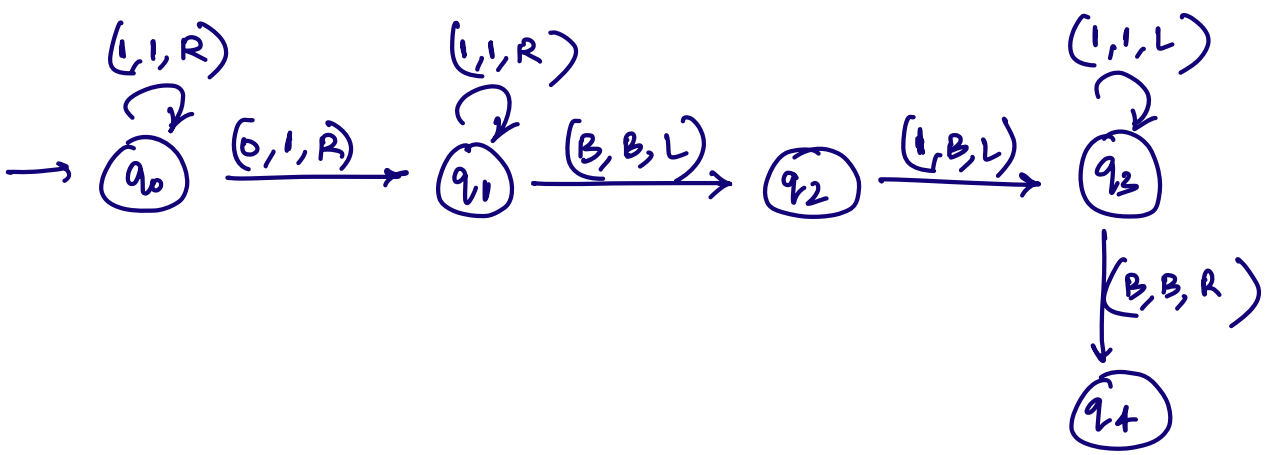
- $a^n b^n$ → TM1
- a+b → TM2

Aim: one TM which can solve every problem

↳ Universal TM



a+b:



2+4:

11 0 1111 Tape 2 : input

Tape 3: states

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

$\Gamma = \{a_1, a_2, a_3 \cdots\}$

Encode every symbol

$q_0 \to 1$ $\quad$ $a_1 \to 1$

$q_1 \to 11$ $\quad$ $a_2 \to 11$

$q_2 \to 111$ $\quad$ $a_3 \to 111$

$q_3 \to 1111$

Tape 1: Transition fx^{n}s

$\delta(q_1, a_1) = (q_2, a_2, R)$

$L \to 1$

$R \to 11$

11 0 1 0 111 0 11 0 11

q_1 a_1 q_2 a_2 R

Entire transition fx^n can be written as a string of 0s & 1's.

Entire TM can be represented as string of 0's & 1's.

TM is one of the string of Σ^* $\quad$ $\Sigma = \{0, 1\}$

Not every string of 0's & 1's is a TM.

Machine → Language

FA → Regular Lang.

PDA → CFL (Context free Language)

TM → RE or Recursive

TM is powerful than PDA.

CFL is a subset of RE language.

Exception

→ TM doesn't have power to accept ϵ, but ϵ can be accepted by PDA.

→ CFL are a subset of RE language (not considering ϵ)

Recursively Enumerable and Recursive Language:

RE

Language accepted by TM is called as RE language.

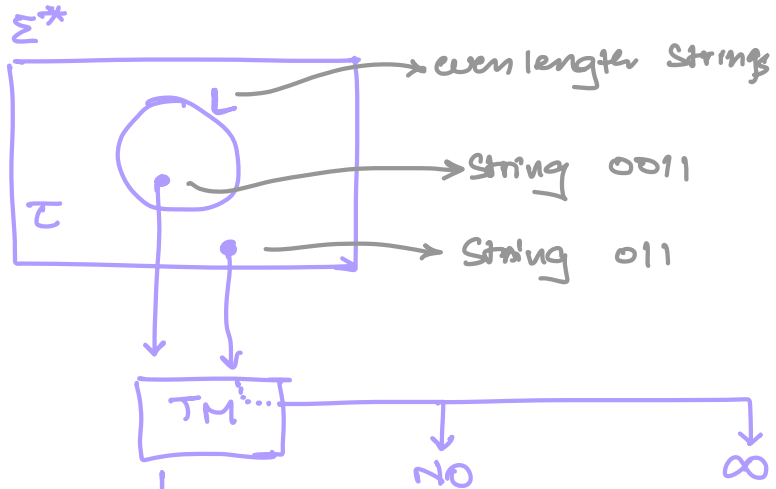
RE V/S Recursive:

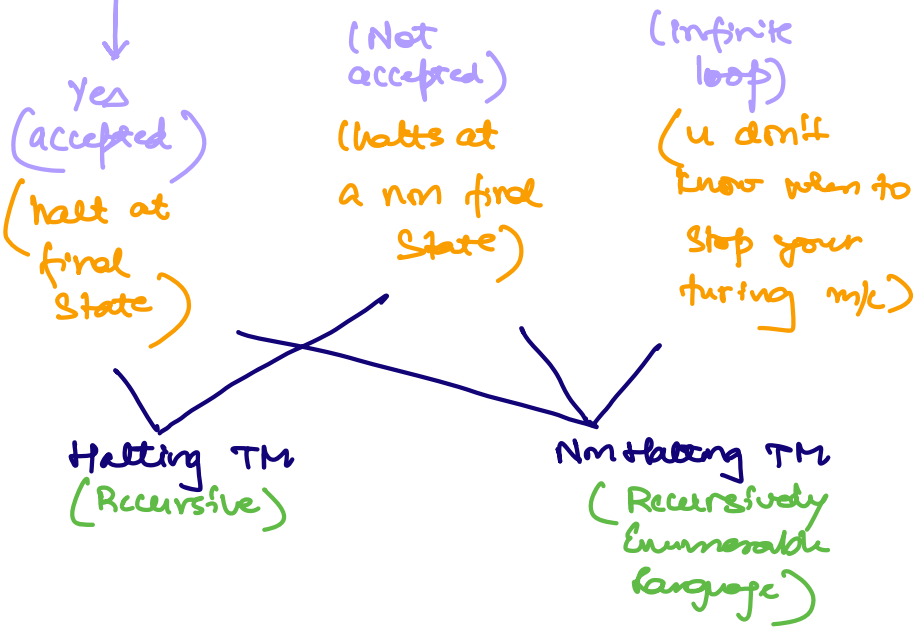
$\Sigma = \{0,1\}$

Σ^* = set of all strings possible → 0, 1, 00, 01, 10, 11, 100 ....

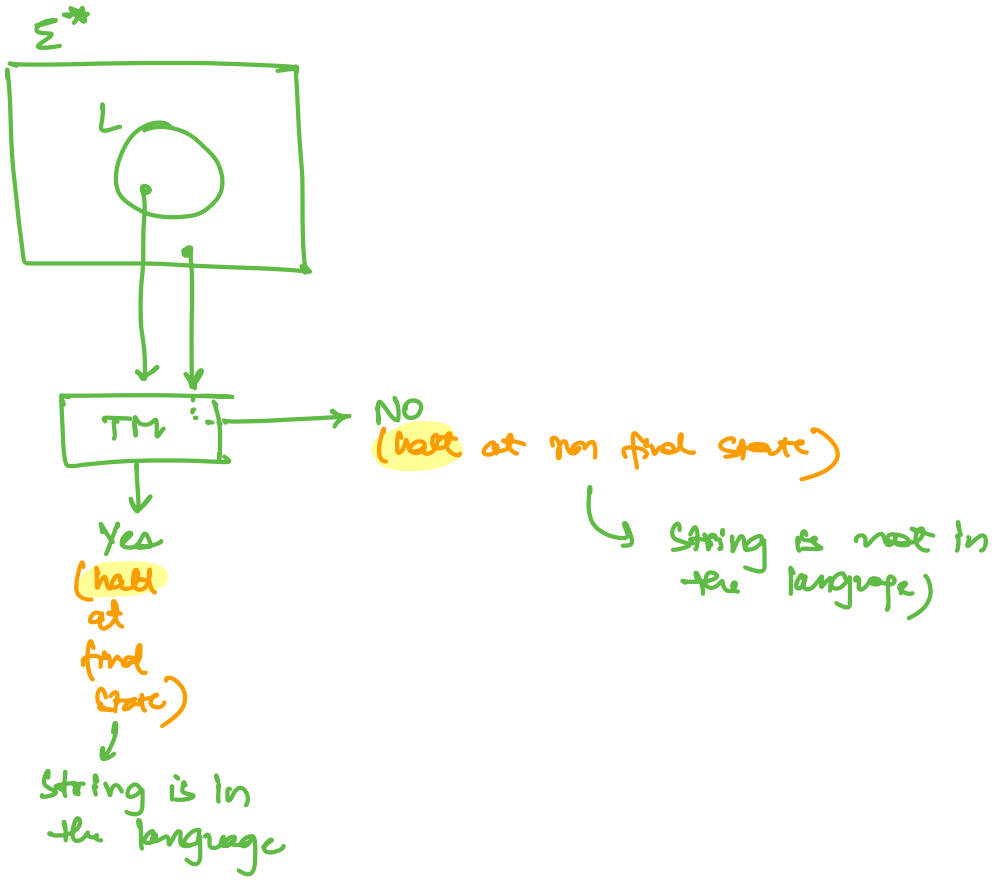
L = Subset of Σ^* → i.e. even length

00, 11, 0000, 1111 .....

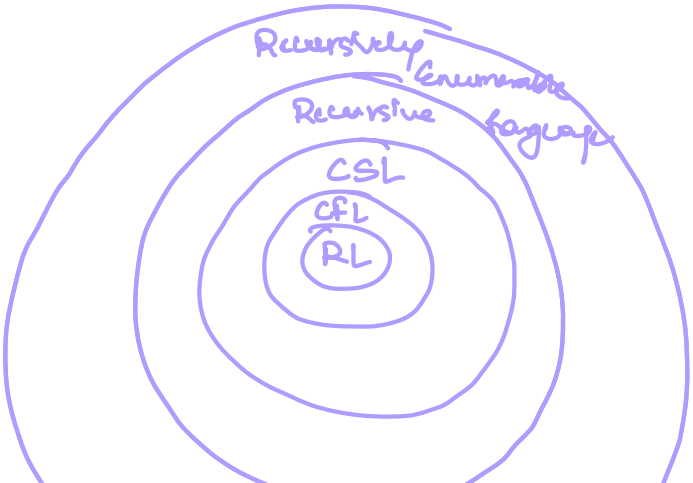




Halting TM

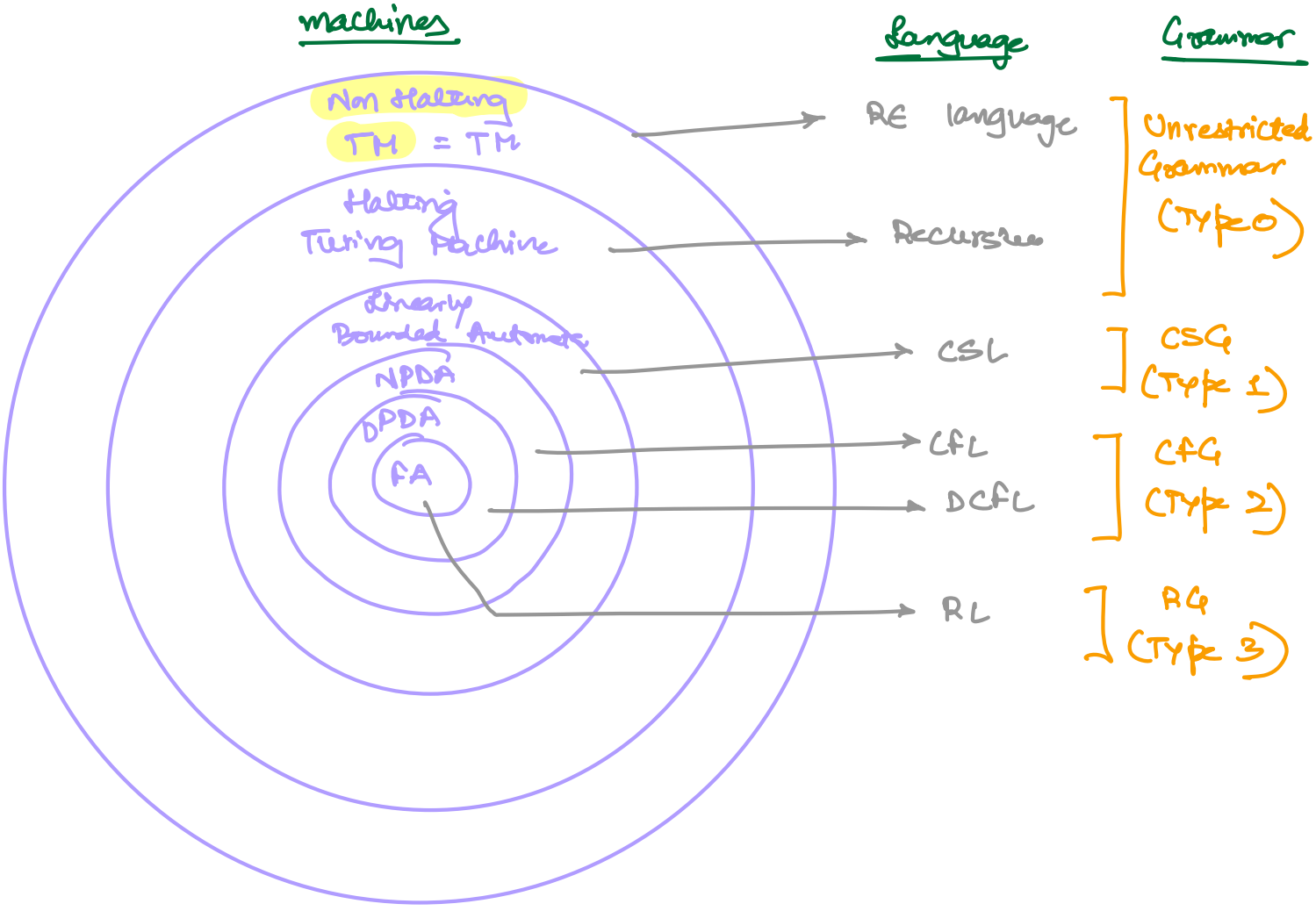


Language:



Every Recursive language is Recursively Enumerable.

Big Picture



Every Recursive language is RE
Every CSL " is Recursive
" CFL " is CSL
" RL " is CFL

Unrestricted Grammar:

A grammar is called unrestricted grammar if all the productions are of the form $u \rightarrow v$

$u \in (V \cup T)^+$

$v \in (V \cup T)^*$

V = Variables
T = Terminals

→ LHS of production can't be empty.

and atleast
1 variable

Q: what language does the following unrestricted grammar derive?

$S \rightarrow S_1 B$

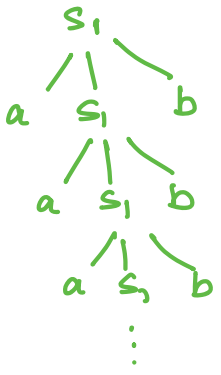
$S_1 \rightarrow a S_1 b$

$bB \rightarrow bbbB$

$aS_1 b \rightarrow aa$

$B \rightarrow \lambda$

} unrestricted grammar



$a^n S_1 b^n$

taking 1 b → generating 2 b's

b's are increasing with a common ratio of 2.

S

$S_1 B$

$a^n S_1 b^n B$

$a^{n-1} a S_1 b b^{n-1} B$

$a^{n-1} aa b^{n-1} B$

$a^{n-1} a^2 b^{n-1} B$

$a^{n+1} b^{n-1} B$

$a^{n+1} \cdot b^{n-2} \cdot b \cdot B$

$B \to \lambda$

$a^{n+1} \cdot b^{n-2} \cdot b \cdot \lambda$

$a^{n+1} \cdot b^{n-1} \cdot \lambda$

$bB \to bbbB$

$a^{n+1} b^{n-2} bbbB$

$a^{n+1} b^{n+1} B$

$B \to \lambda$

$a^{n+1} b^{n+1} \lambda$

$a^{n+1} \cdot b^{n} \cdot bB$

$a^{n+1} b^{n} bbbB$

$a^{n+1} b^{n+3} B$

$B \to \lambda$

$a^{n+1} b^{n+3} \lambda$

$L = \{ a^{n+1} b^{n+k} \lambda \; ; \; n \geq 1, k = -1, 1, 3, 5 \ldots \}$

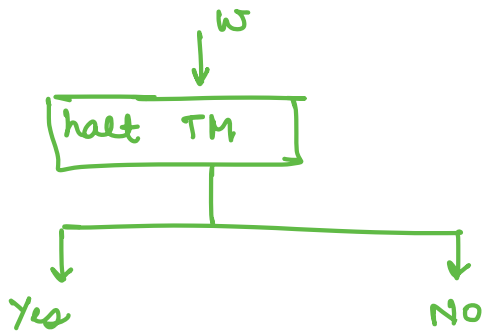
Membership Algorithm:

Given a string and a language, tell whether string belongs to the language or not.

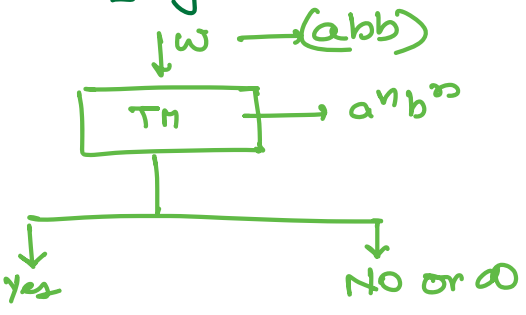
Give an answer in Yes/NO.

String: aabb Language: $a^n b^n$ → TM.

Recursive language



Recursively Enumerable language



(w ∈ L)
(halts at a final state)

(w ∉ L)
(halts at a non final state)

(w ∈ L)
(halts at a final state)

(w ∉ L)
may halt at a non final state or it can go in an ∞ loop

Membership algo exists

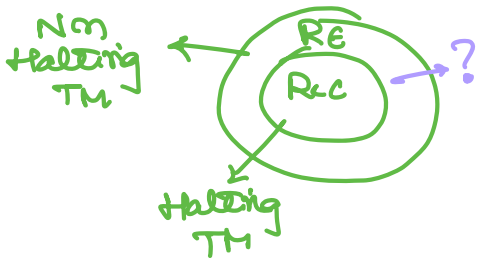
If m/c halts at non final state : NO
If m/c halts at a final state: Yes

membership algo doesnot exist bcz we might go into an ∞ loop, we will keep on waiting we dont get the answer in Yes/NO.

If you are not able to get ans in Yes/No, ⇒ membership algo doesnot exist.

Theorem 1:

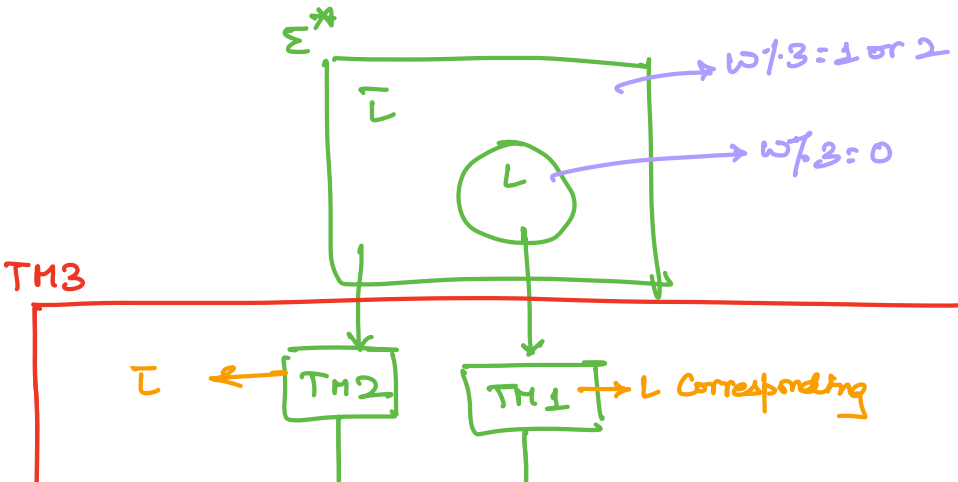
If a language L & its complement $\bar{L}$ both are recursively enumerable then both languages are recursive.

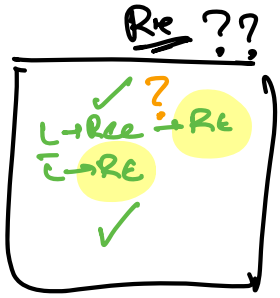


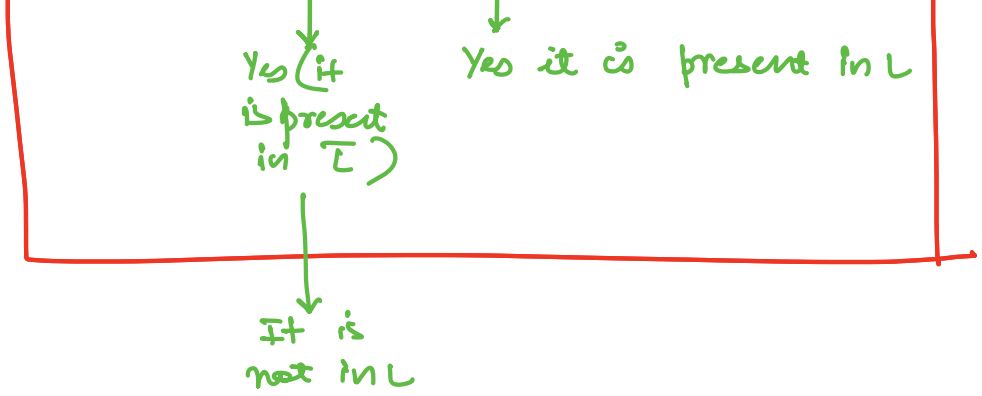
Proof:

$\Sigma = \{0,1\}$

Σ^* = Set of all strings







Using TM1 and TM2 create a new TM3

Give strings to both TM1 & TM2

At least one of them will stop & say string is present in L or $\bar{L}$

If TM1 halts at final state : String is present in L

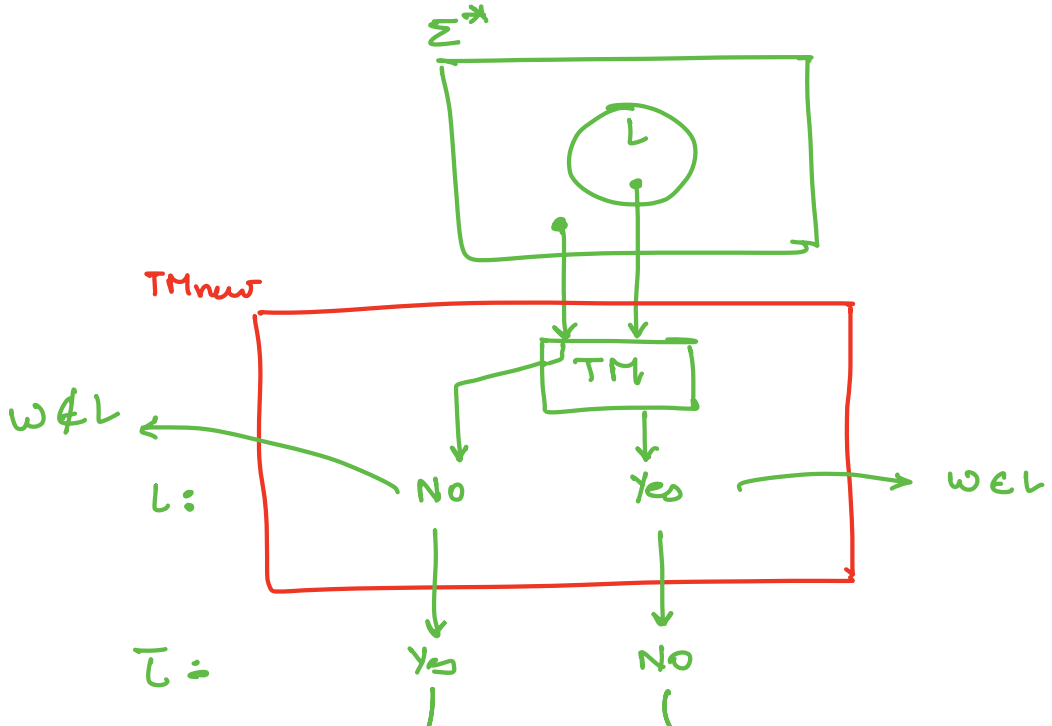
If TM2 ——————— : String is not present in L.

There is a TM3, which will give ans in Yes/NO.

L & $\bar{L}$ are actually recursive.

Theorem 2:

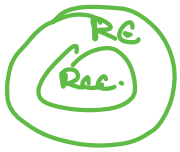
If L is recursive then $\bar{L}$ is also recursive and consequently both are R.E.

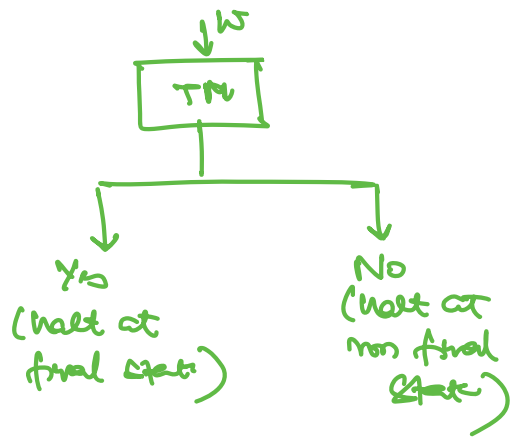
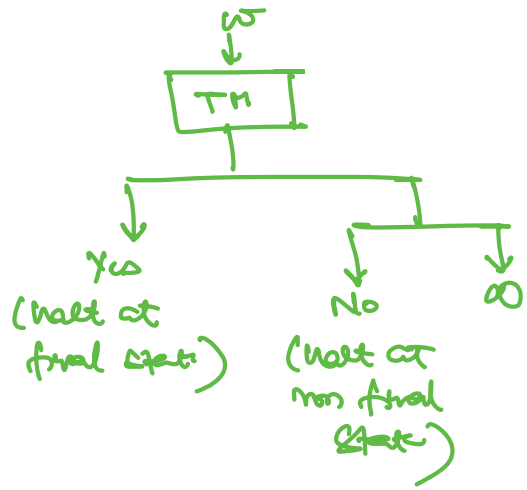


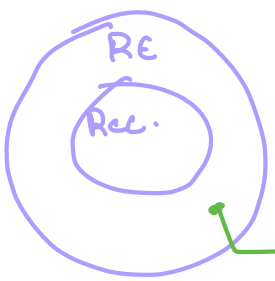
$w \in \bar{L}$ $\quad$ $w \notin \bar{L}$

$\bar{L}$ is Recursive $\Rightarrow$ Halting TM for $\bar{L}$.

Recursive is a subset of RE. Hence, everything that is Recursive is also RE.



Recursive	Recursively Enumerable
→ Halting TM (TM which halts)	→ TM (may halt or may not halt)
	
membership algo exists	no membership algo.



Recursive languages are a proper subset of RE language.

→ this has already been proven that there exist at least 1 language which is RE but not recursive.

Decidability :

Problems → Ans: Yes/NO

→ If there exist an algo to solve this problem then you can say problem is decidable.

eg: number 'n' is prime or not ?

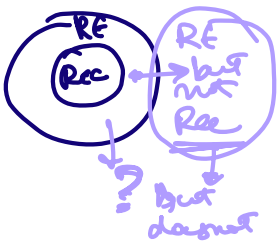
→ Algo do exist

→ Decidable.

Halting problem of TM is undecidable.

TM String w — no algo exists.

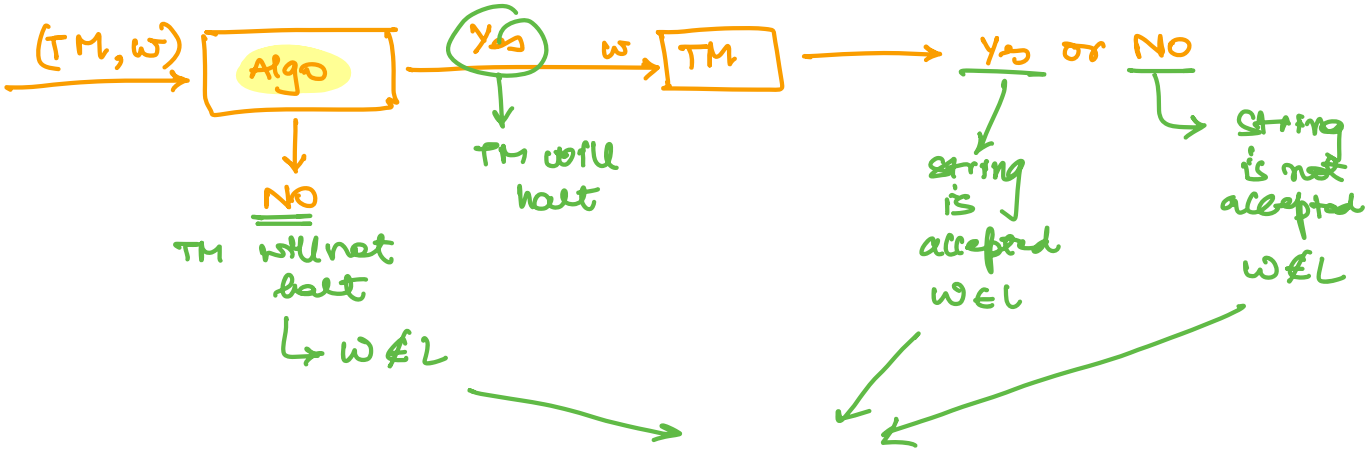
There is no algo which can tell us whether our during m/c will halt or not when string w is provided to it.



Proof by Contradiction:

Assume: Halting Problem is Decidable,

If Halting problem of TM is decidable it means there exist an algo which can tell if M will halt or not.



RE there exist a membership algo.

All RE languages are recursive

↓

but this contradicts our fact that there exist atleast 1 language which is RE but not recursive.

Hence, our initial assumption is wrong and Halting problem of TM is undecidable.