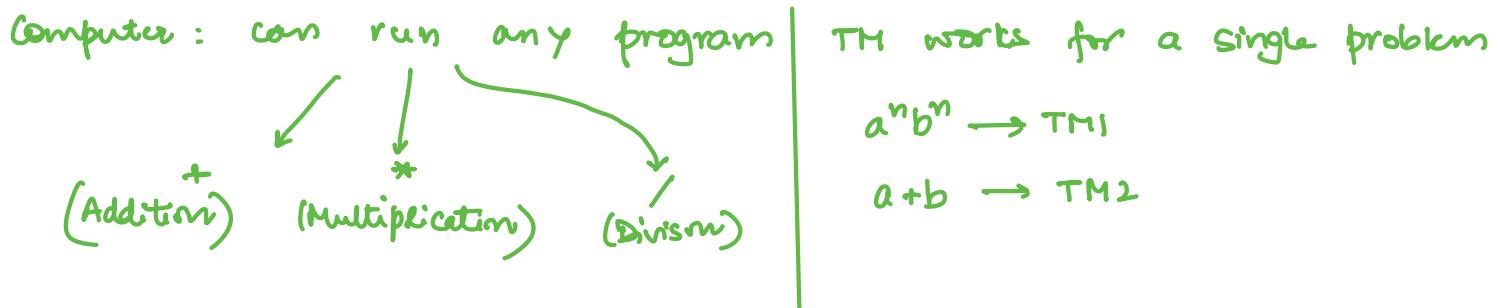


UNIVERSAL TURING MACHINE:

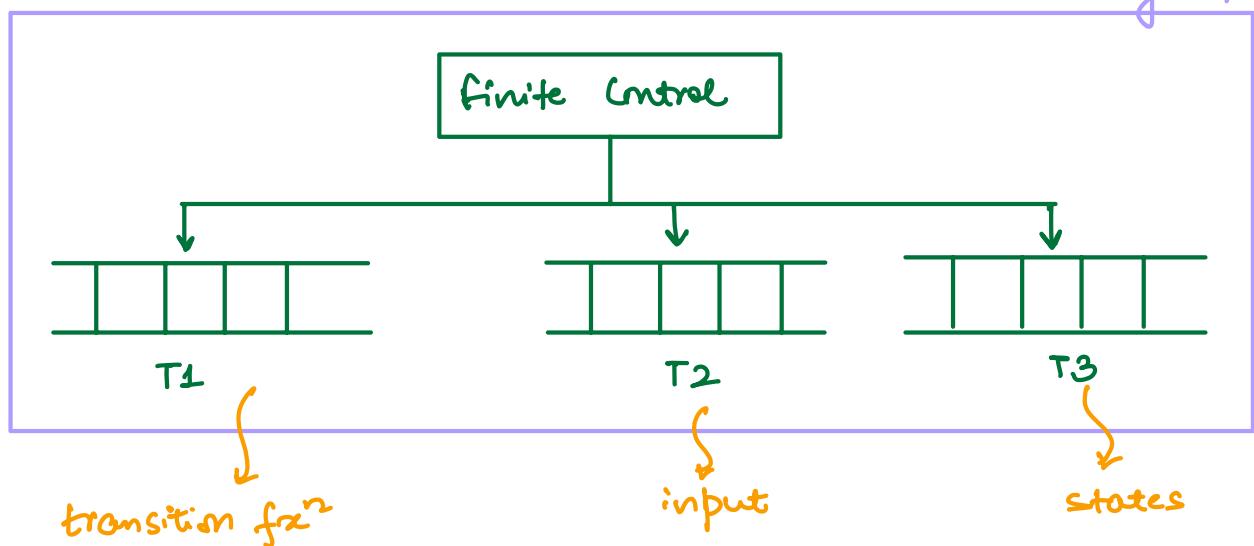
Turing Thesis: TM is as powerful as a computer



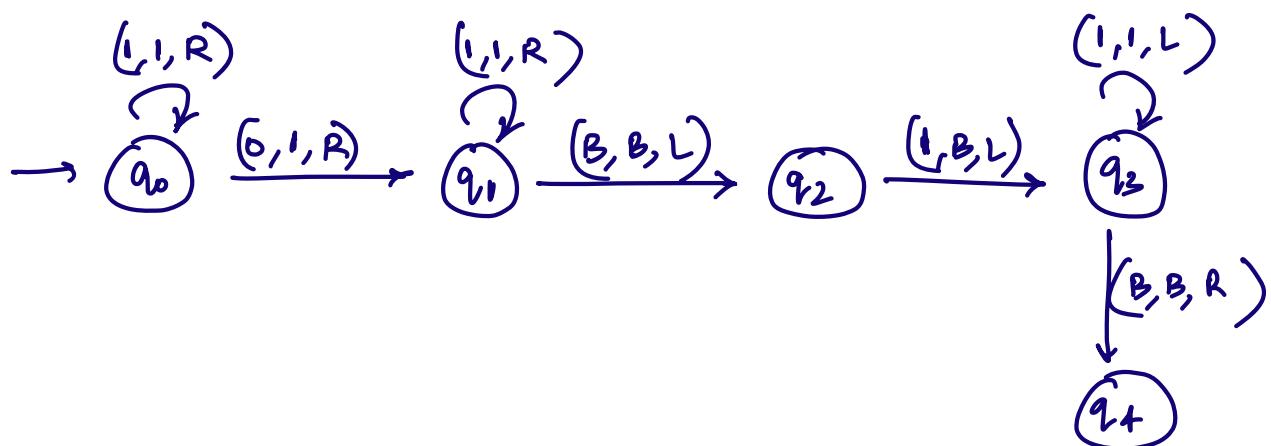
Aim: one TM which can solve every problem

↳ Universal TM

universal single TM



$a+b$:



2+4:
11 0 1 1 1 1 Tape 2 : input

Tape 3: states

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

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

Encode every symbol

$$\begin{array}{ll}
 q_0 \rightarrow 1 & a_1 \rightarrow 1 \\
 q_1 \rightarrow 11 & a_2 \rightarrow 11 \\
 q_2 \rightarrow 111 & a_3 \rightarrow 111 \\
 q_3 \rightarrow 1111 & \vdots \\
 \vdots & \vdots
 \end{array}$$

Tape 1: transition func

$$\delta(q_i, a_j) = (q_k, a_l, R)$$

$$\begin{array}{ccc}
 \curvearrowleft & & L \rightarrow 1 \\
 \frac{\overline{11} \ 0 \ \overline{1} \ 0}{q_1 \ a_1} & \frac{\overline{111} \ 0 \ \overline{11} \ 0}{q_2 \ a_2} & R \rightarrow 11
 \end{array}$$

Entire transition func can be written as a string of 0's & 1's.

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

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

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

Machine \rightarrow Language

FA \rightarrow Regular Lang.

PDA \rightarrow CFL (Context Free Language)

TM \rightarrow RE or Recursive

TM is powerful than PDA.

CFL is a subset of RE language.

Exception

\rightarrow TM does not have power to accept ϵ , but ϵ can be accepted by PDA.

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

Recursively Enumerable and Recursive language:

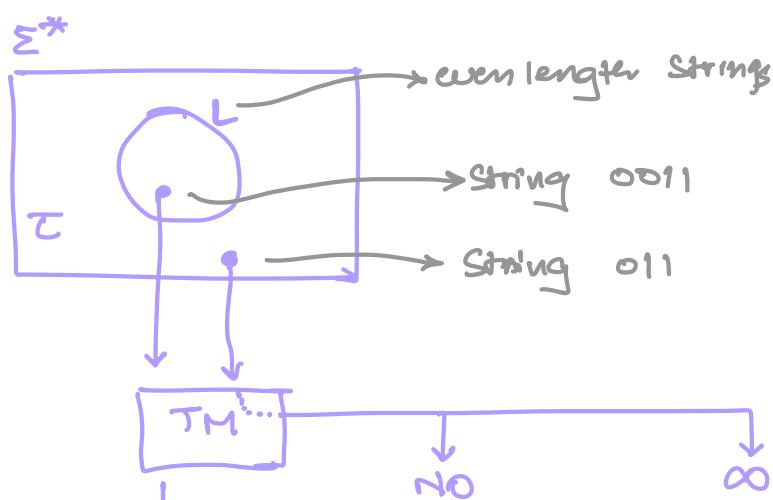
Language accepted by TM is called as RE language.

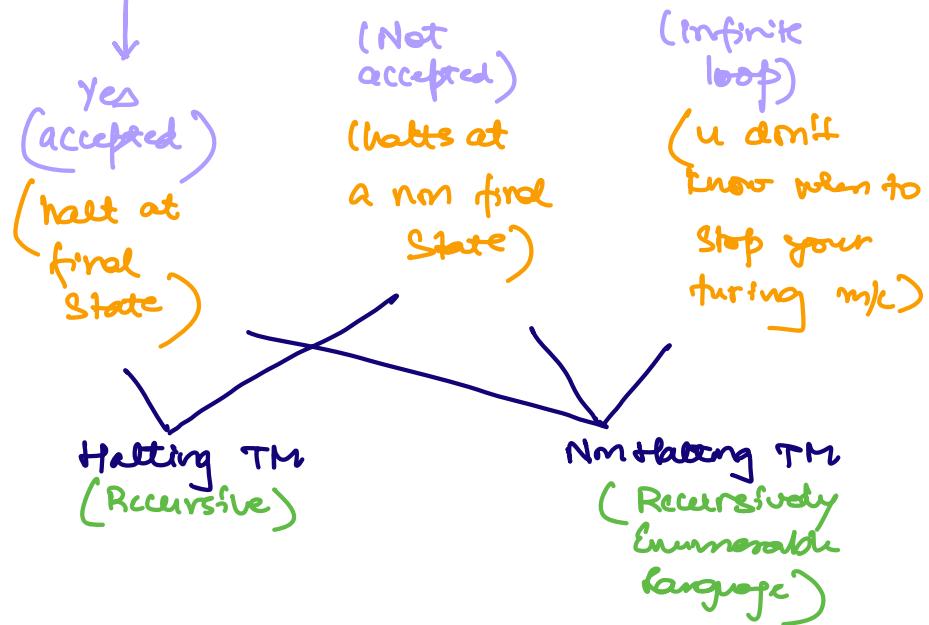
RE v/s Recursive:

$\{\varnothing, \{1\}\}$

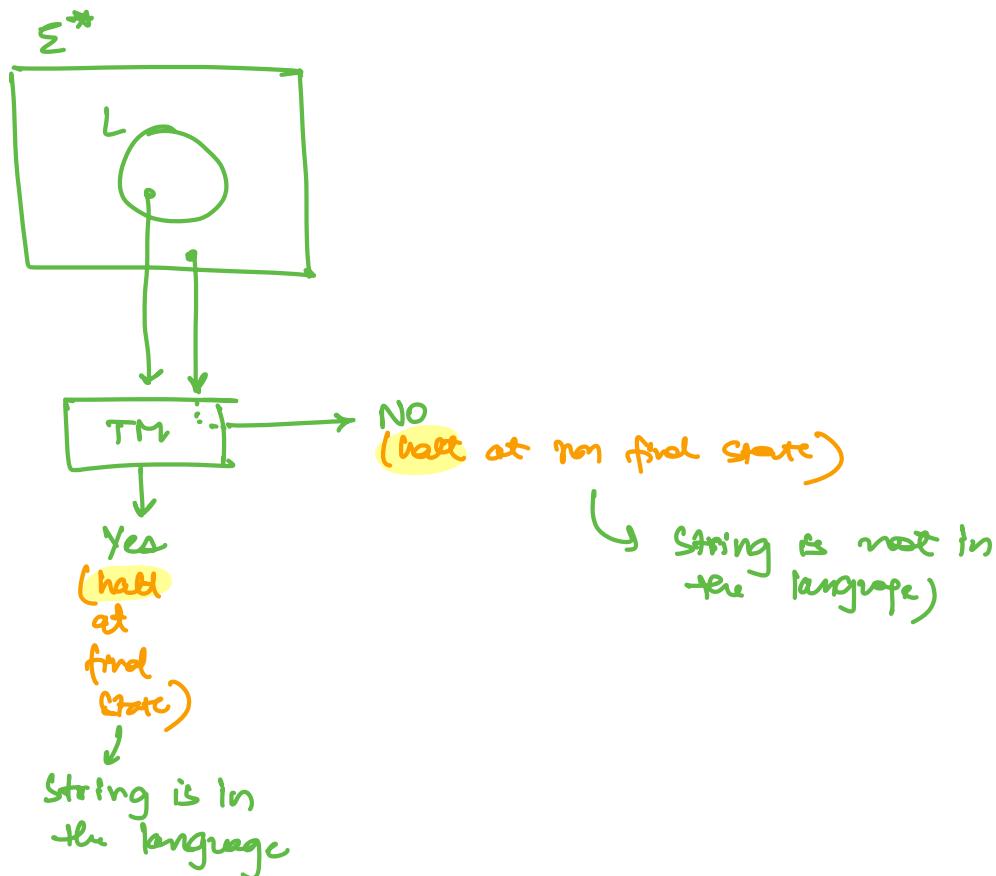
Σ^* = set of all strings possible $\rightarrow 0, 1, 00, 01, 10, 11, 1000 \dots$

L = Subset of Σ^* \rightarrow Given length
 $00, 11, 0000, 1111 \dots$

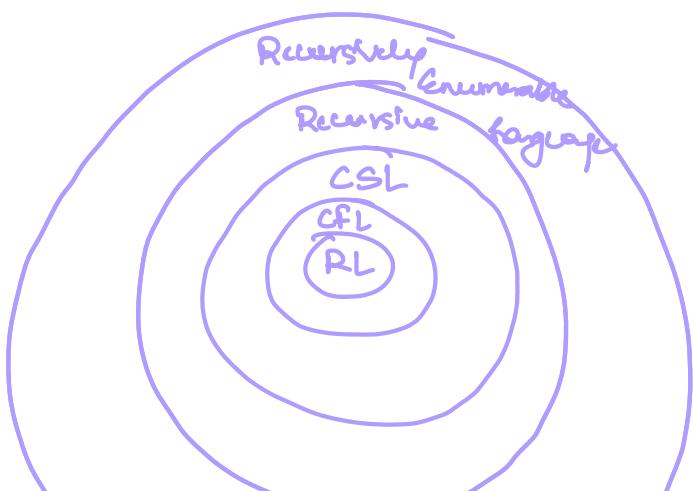




Halting TM

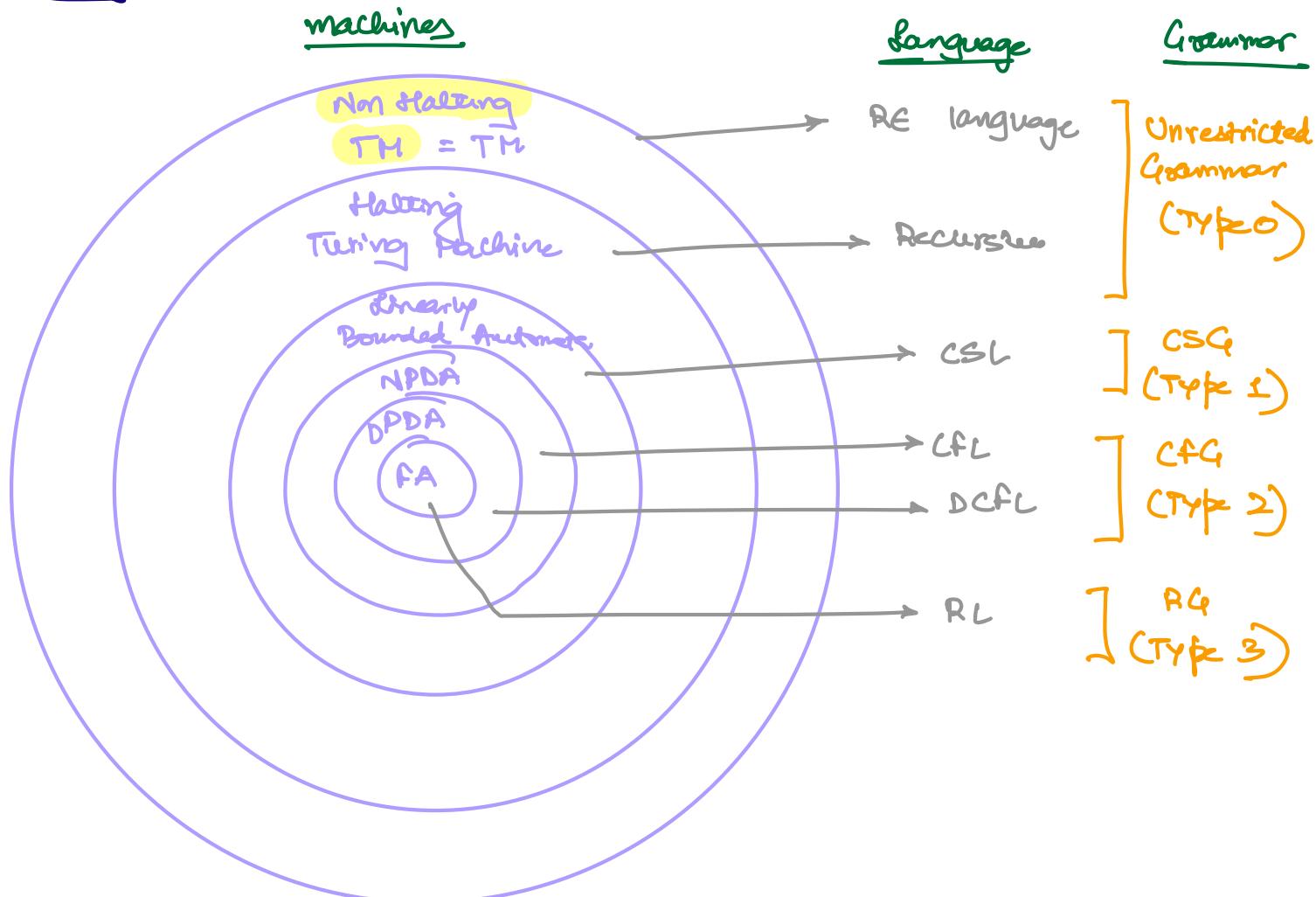


Language:



Every Recursive language is Reversibly Enumerable.

Big Picture



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

Unrestricted Grammar:

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

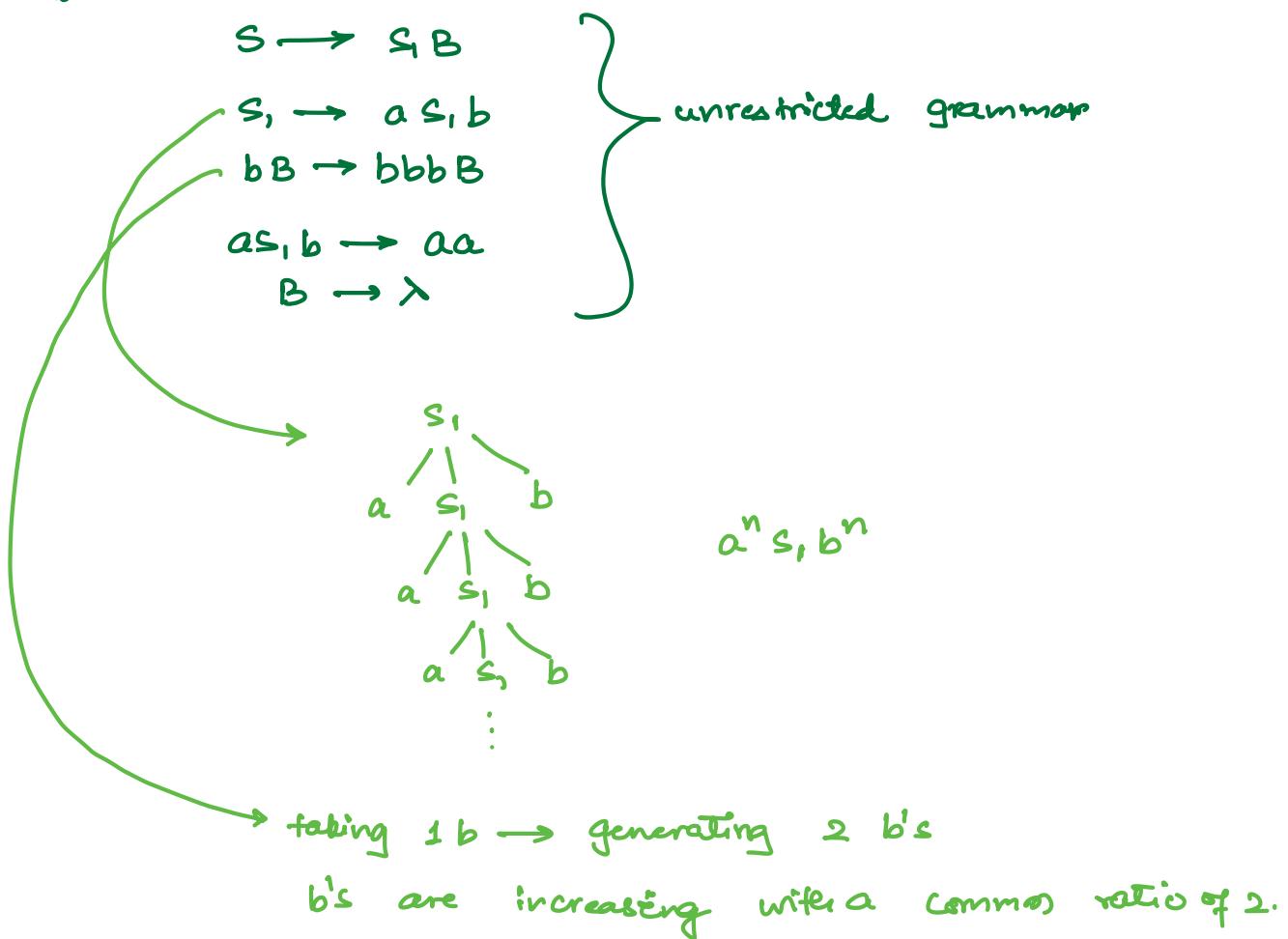
$$u \in (VUT)^+$$

$$v \in (VUT)^*$$

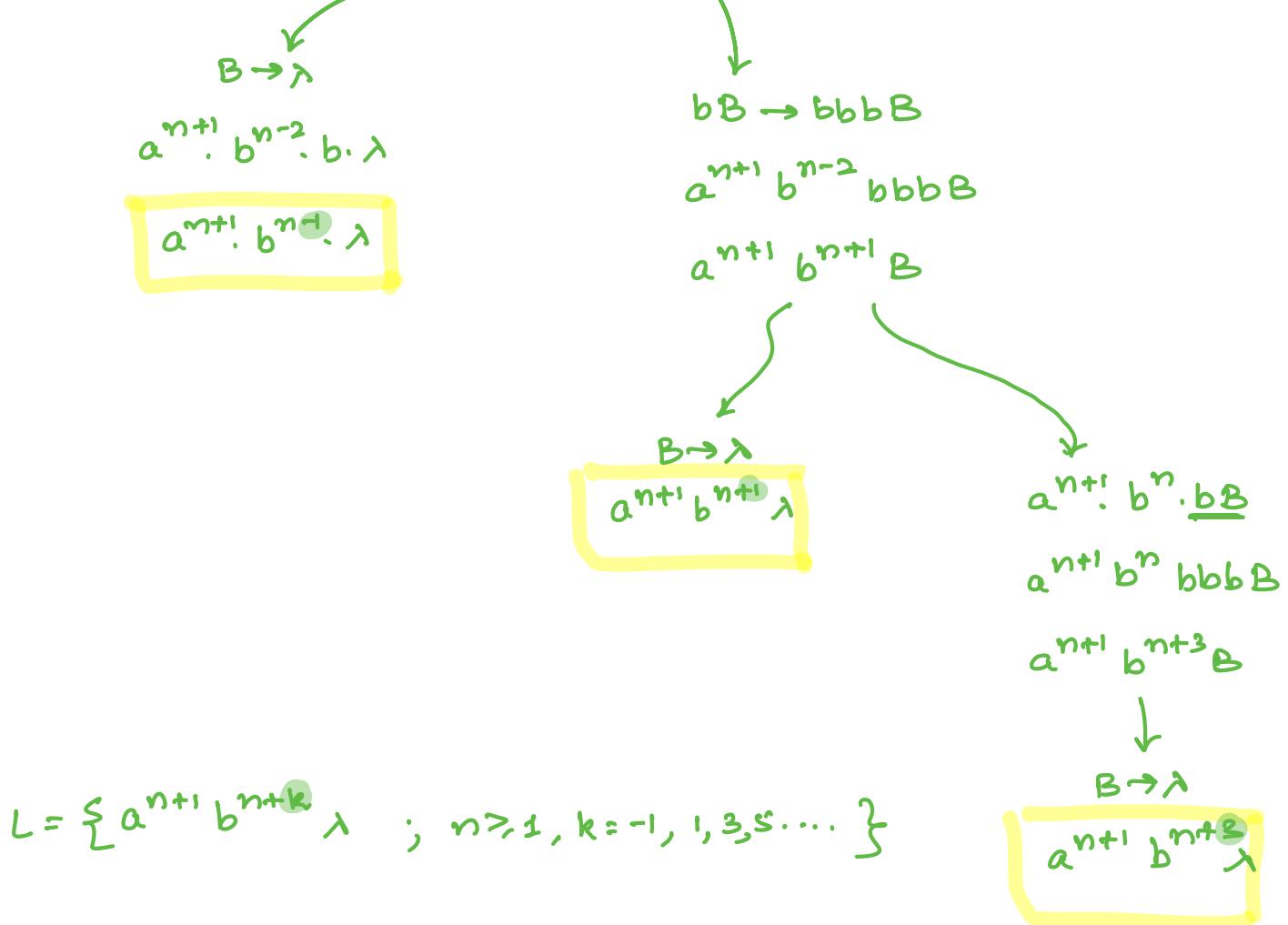
V = Variables
T = Terminals

LHS of production can't be empty.

Q: what language does the following unrestricted grammar derive?



$$\begin{aligned} & S \\ & S_1 B \\ & a^n S_1 b^n B \\ & \underline{a^{n-1} a S_1 b} b^{n-1} B \\ & a^{n-1} aa b^{n-1} B \\ & a^{n-1} a^2 b^{n-1} B \\ & a^{n+1} b^{n-1} B \\ & a^{n+1} \cdot b^{n-2} \cdot b \cdot B \end{aligned}$$



Membership Algorithm:

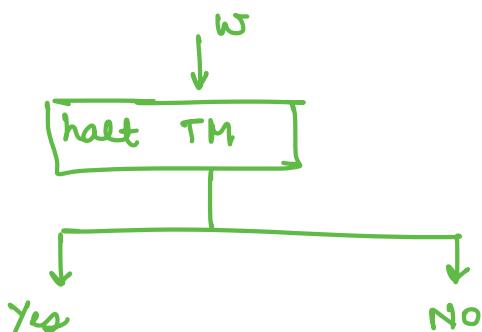
Gives a string and a language, tell whether string belongs to the language or not.

Give ans answer in Yes/No.

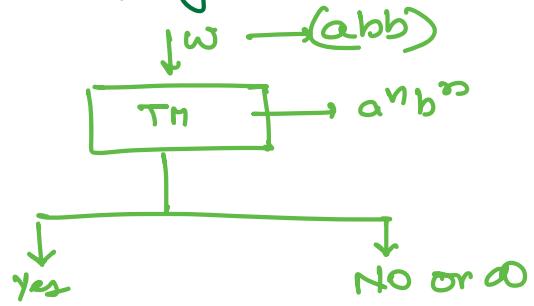
String: aabb

language: $\frac{a^n b^n}{\hookrightarrow \text{TM}}$

Recursive Language



Recursively Enumerable Language



$(w \in L)$
(halts at a final state)

$(w \notin L)$
(halts at a non final state)

$(w \in L)$
(halts at a final state)

$(w \notin L)$
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

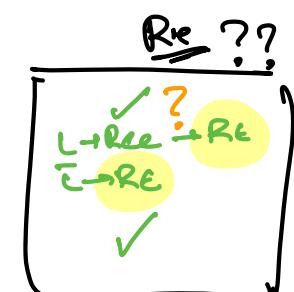
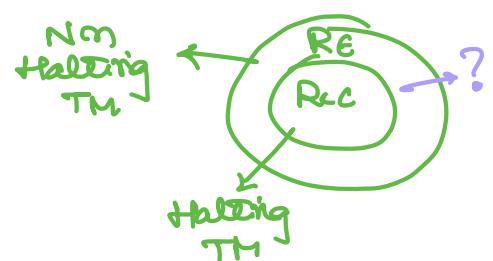
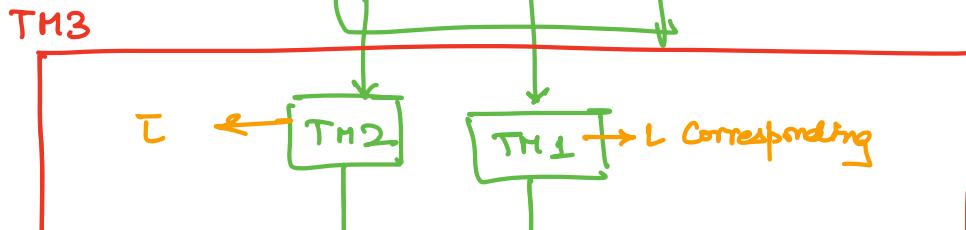
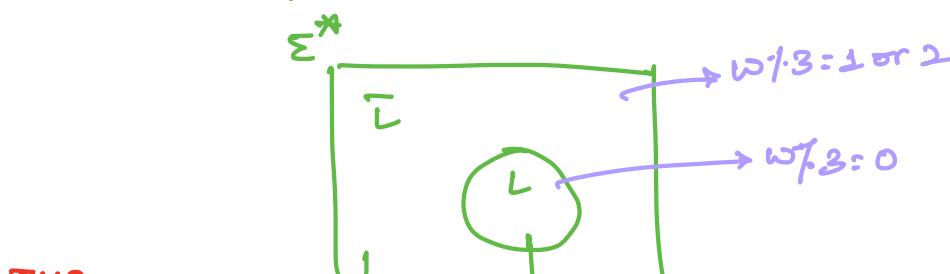
membership algo doesn't exist bcz we might go into an ∞ loop, we will keep on waiting we don't get the answer in Yes/No.

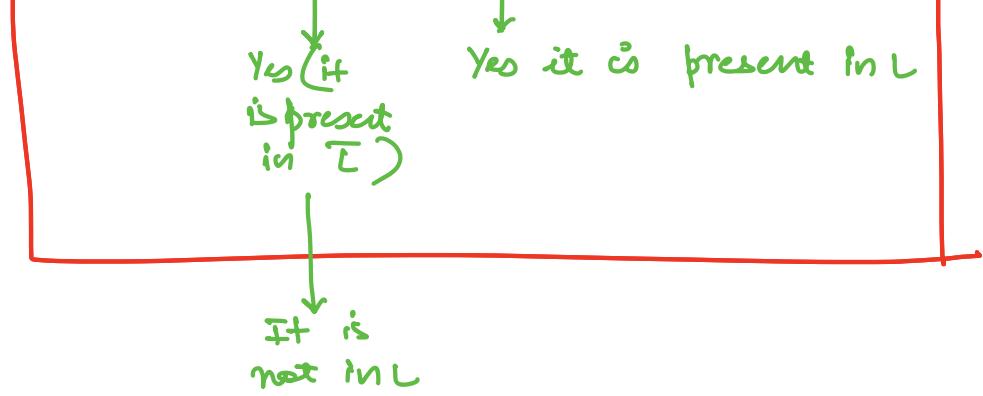
If you are not able to get ans in Yes/No, \Rightarrow membership algo doesn't exist.

Theorem 1:

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

Proof: $\Sigma = \{0,1\}$
 $\Sigma^* = \text{set of all strings}$





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 T-bar

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

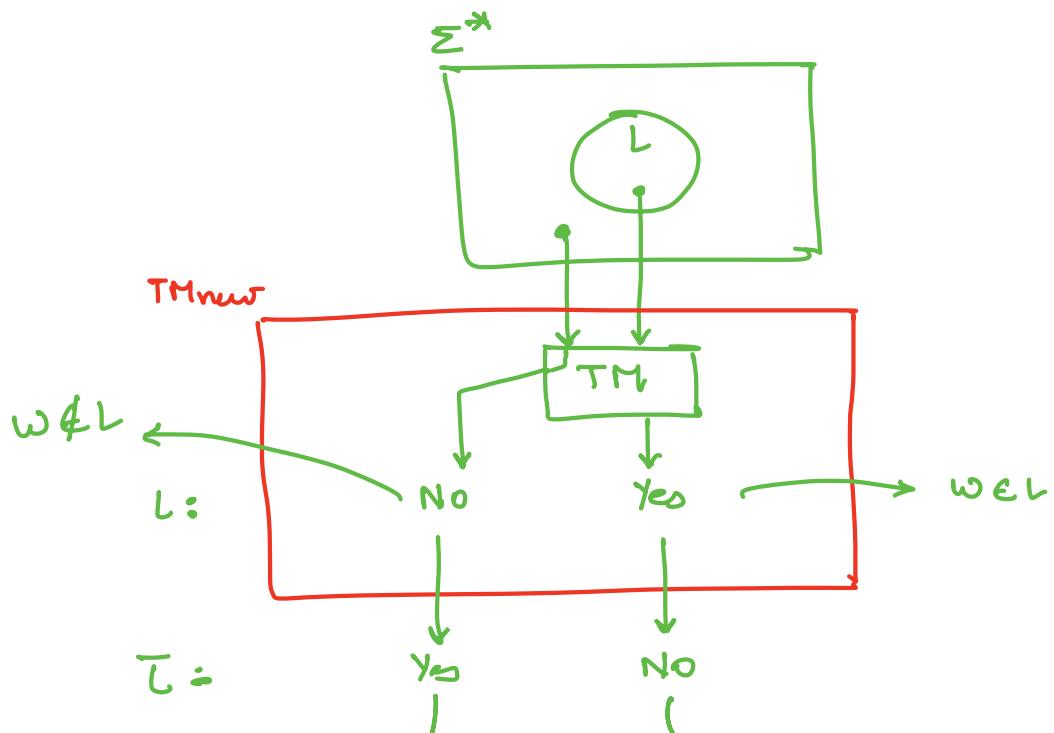
If TM₂ _____ : string is not present in L.

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

L & T-bar are actually recursive .

Theorem 2:

If L is recursive then T-bar is also recursive and consequently both are R.E.

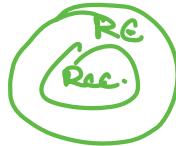


$w \in T$

$w \notin T$

T is Recursive \Rightarrow Halting TM for T .

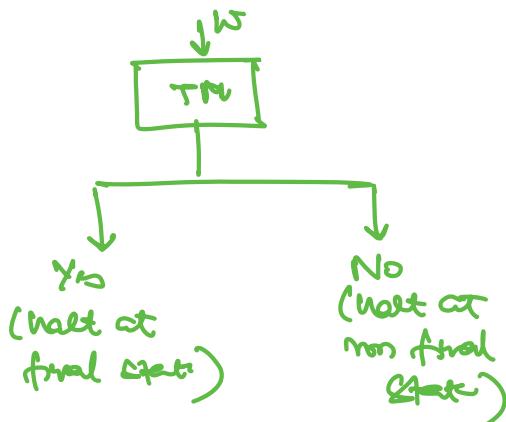
Recursive is a subset of RE. Hence, everything that is Recursive is also RE.



Recursive

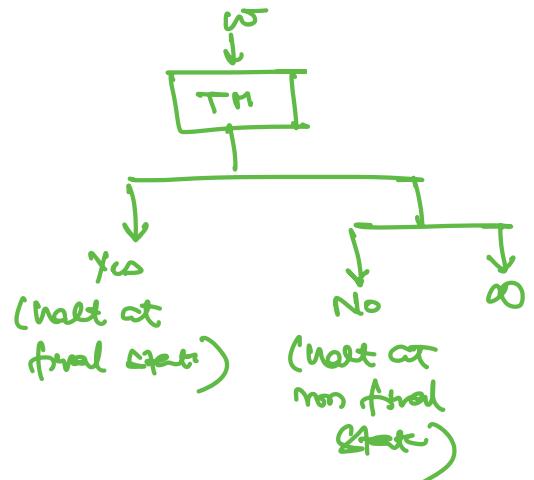
Recursively
Enumerable

\rightarrow Halting TM
(TM which halts)

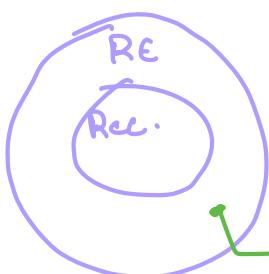


membership algo exists

\rightarrow TM
(may halt or
may not halt)



no membership algo.



Recursive languages are a proper subset of RE languages.

\rightarrow 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 if TM is undecidable.

TM String w
 ↙

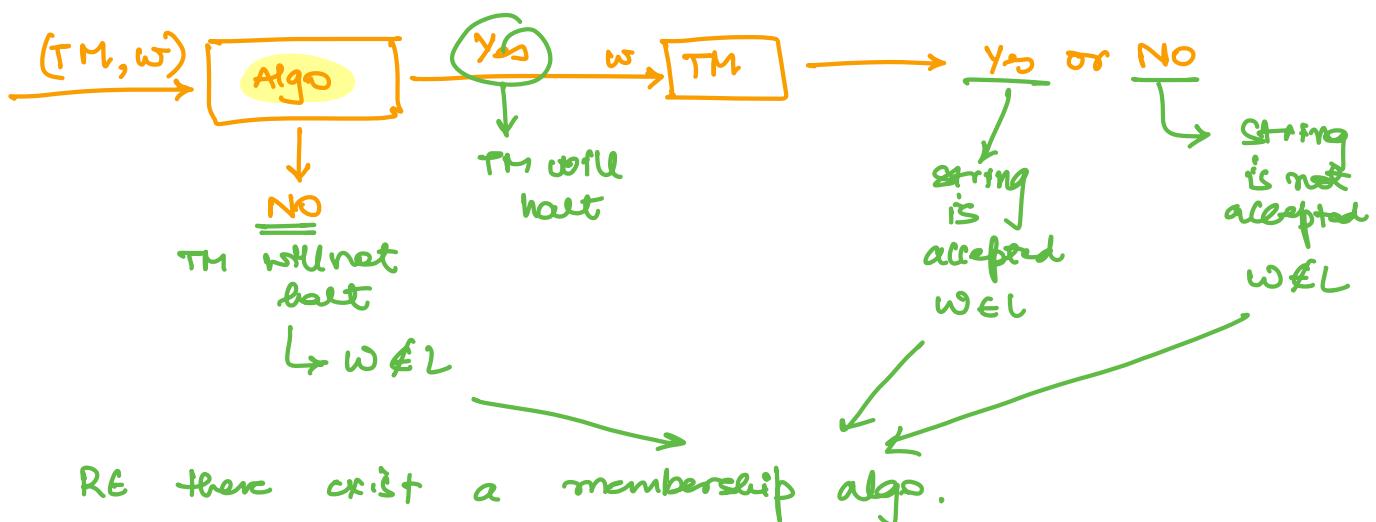
no algo exists.

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

Proof by contradiction:

Assume: Halting Problem is Decidable,

If Halting problem is Decidable then there exist an algo which can tell if M will halt or not.



All RE languages are recursive

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

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