membership algo:

String → Machine → Yes / No

RL → FA
CFL → PDA
Rec. Lang → HTM
} Halt

RL, CFL & Rec languages. → membership algo exists.
CSL

Recursive / Enumerable → TM

RE → no membership algo

Decidability

Problem → Answer: Yes / NO

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

Eg: Problem: Given a no. 'n' is it prime or output?

Answer: Yes/NO

→ Decidable

Prime → Algo

Reducability

$$P_1 \xrightarrow{(Algo)} P_2$$

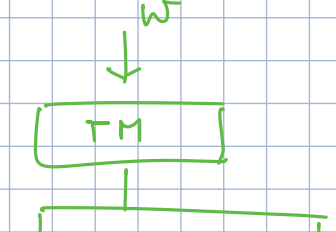
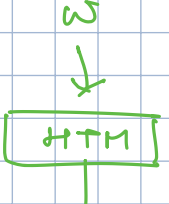
- If P_2 is having an algo (P_2 is decidable) it means P_1 will also have an algo (P_1 is decidable)

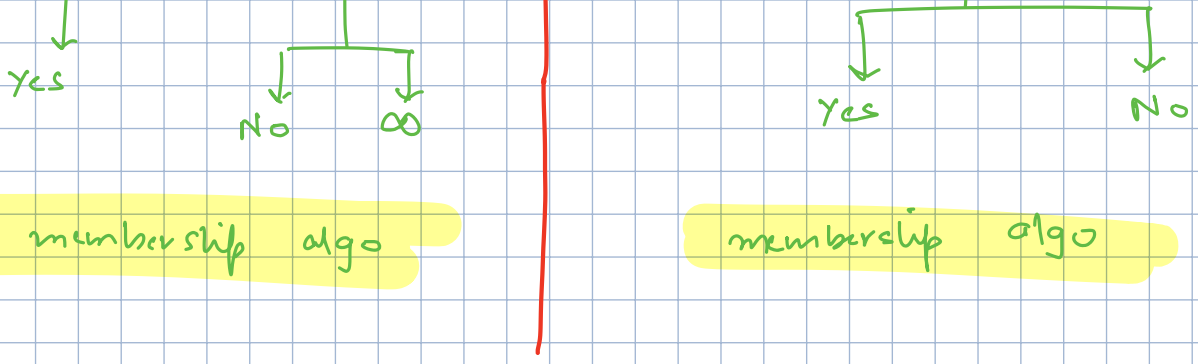
If P_2 is decidable then P_1 is also decidable.

Algo for P_1: (convert P_1 to P_2 + solve P_2)

(convert P_1 to P_2 : Algo; solve P_2 : Algo)

- If it is already proven that P_1 is undecidable then definitely P_2 will also be undecidable.

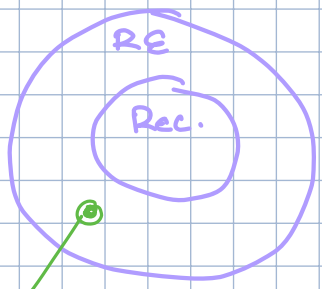
Recursive Enumerable	Recursive
→ Turing Machine	→ Halting TM. (TM which halts)
→ 	→ 



no membership algo

membership algo

Recursive languages are a proper subset of RE languages.



this theorem has already been proven that there exist at least 1 language which is RE but not Recursive

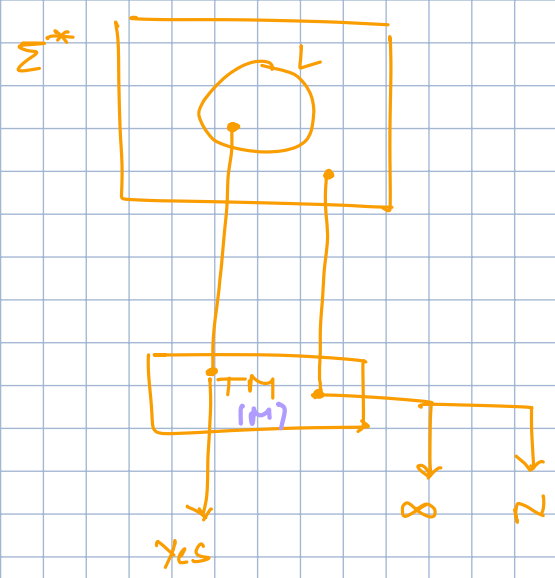
Halting Turing Machine problem is Undecidable:

→ T.M. M, string w

no algo

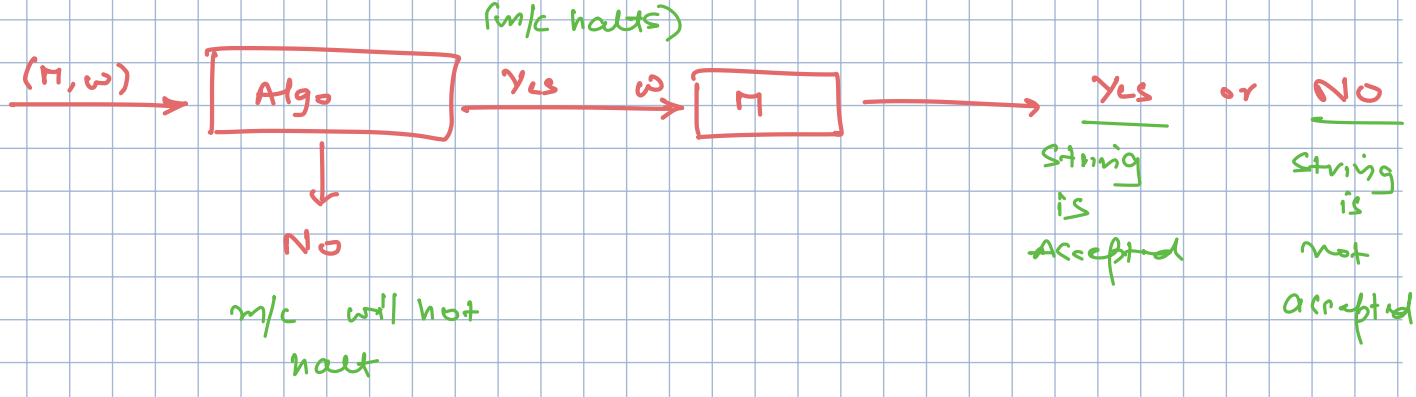
There is no algo which can tell us whether our Turing m/c M will halt or not when string w is provided to it.

Proof: Language L which is RE, exist a TM for it



Assume: Halting problem of TM is decidable.

If Halting problem of TM is decidable it means there exist an algo which can say M will halt (Yes) or not (No).



Algo will definitely halt.

RE L there exist a membership algo.

All RE languages are Recursive

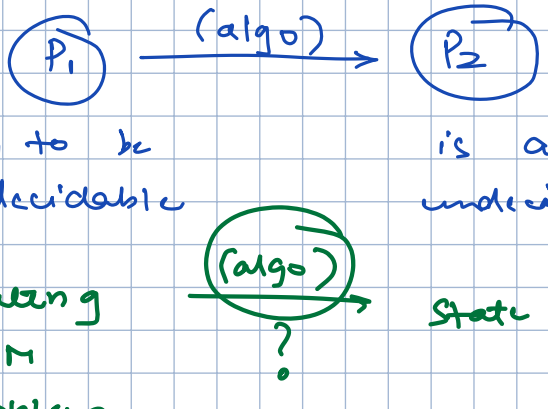
↳ but this contradicts our fact that there exist atleast 1 language which is RE but not Recursive.

Hence, u can say our initial assumption was wrong. You can say Halting problem of TM is undecidable.

State Entry Problem of TM is undecidable

Given a TM, a state $q \in Q$ and $w \in \Sigma^+$, decide whether or not the state 'q' is ever entered when 'w' is applied to 'M'.

There is a TM and one of the state of TM is 'q' and input is w, find out that state q is ever entered by TM or not ?

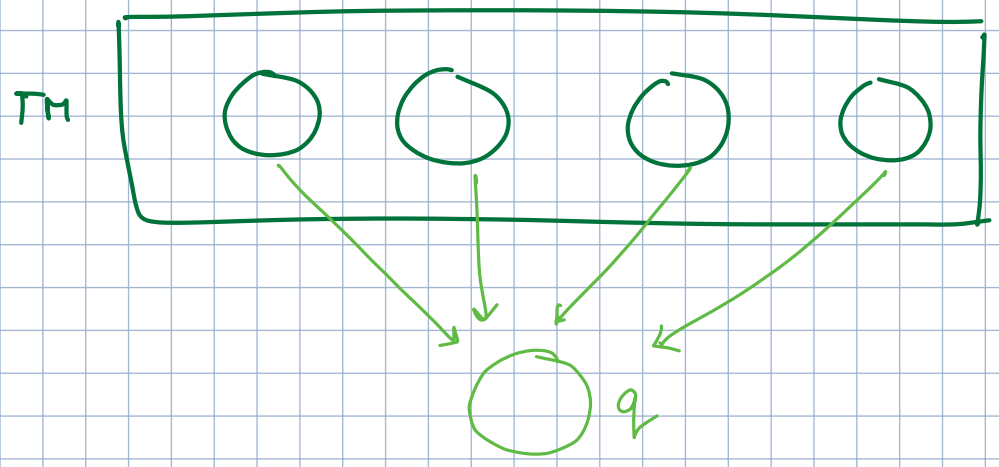


We want to convert Halting TM Problem to State Entry Problem.

→ TM halts when it reaches a dead configuration.

→ m/c is at a state and u are looking at some input symbol, and no transition has been defined for state & input symbol.

All the final states of TM are dead configurations; bcz once u reach final state and whatever symbol you look at, no transition is defined.



for every state where there is a dead configuration we give a transition to state q.

Actual halting problem is now reduced to halting in state q.

Halting TM Problem → (algo) → State entry Problem.

(Undecidable)

Hence, this could be undecidable.

Almost any problems related to RE language is undecidable. For RE there is no membership algo.