The Halting Problem

Given some TM M & a string x,

can you determine if M halts on x

↑ either accepts, or rejects & stops, but no ∞-loop.

$$L_h = \{ M \# x \mid M \text{ is a TM, } x \text{ is some input string, and } M \text{ halts on } x \}$$

↑ language of the halting problem.

is L_h RE? ← the lang of any TM

is L_h recursive? ← the lang of a total TM.

L_h is RE: Let U be a universal machine,

when the input of U is $M \# x$:

if x is accepted on M, then U halts

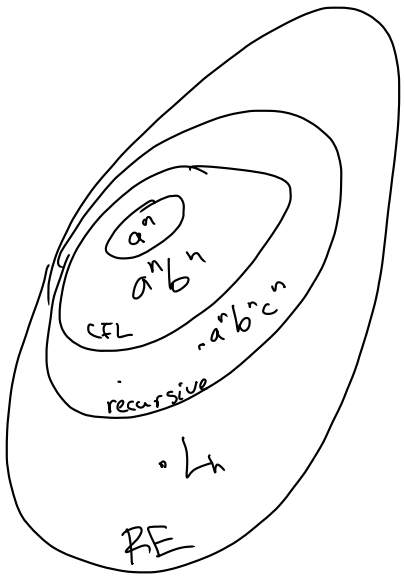
if x is rejected on M, then U halts

if x loops forever on M, then U loops forever.

We can make it so that $\mathcal{U}$ accepts whenever M halts on x.

Then $L(\mathcal{U}) = L_h$,

so L_h is RE.



But L_h is not recursive

Thm L_h is not recursive.

Pf To get a contradiction, assume L_h is recursive, i.e. we have a total TM whose lang. is L_h.

We'll look at what happens for strings like $M \# M$

Using our total TM, we can build a machine H s.t.:

1. H accepts $M \# M$ whenever M does not halt with input M.

2 H loops forever on M#M whenever M halts on input M.

What happens when we run H with input H#H

2 options: it halts or not. either is contradictory

If it halts: by #2, H will loop forever

If H loops forever: by #1, H will accept.

These are both self-contradicting, so this H cannot exist, so L_h is not recursive.

Application: There can be no general purpose ∞-loop detector for real programming languages.

Think: If there were an ∞-loop detector,

$6 = 3+2+1$ ← "perfect #"

are there ∞-ly many perfect #'s?

```
n=2
while true:
```

```
if n is perfect:
    break
n = n+1
```

So a TM can actually be improved by making it able to solve a halting problem.

"A TM with halting oracle"

Real computers have the same limitations as TMs.

What about people?

The brain is a physical object & (probably) behaves according to the laws of physics.

So probably, the brain is a TM.