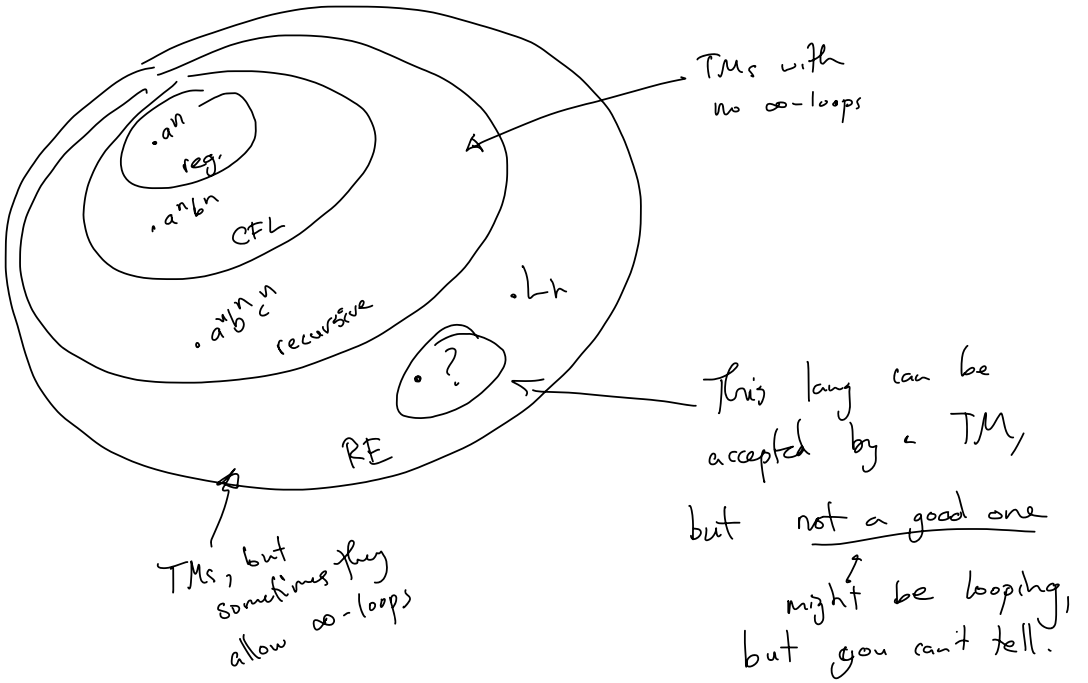

About simulating other TMs on a TM.

This requires taking one TM: M,

"encode it" as a string,

also encode the input to M: x

Our universal machine takes input like

$M \# x$

M: a string encoding of M

x: some input string

& we can simulate it: i.e.

$\mathcal{U}$ takes $M \# x$,

then it is accepted on $\mathcal{U}$ if M accepts x
rejected " " " rejects x
loops forever " " " loops forever on x.

Our non-recursive language is based

the Halting Problem

Given a TM M & some input x,
determine if M halts with input x.

A language-version:

$$L_h = \{ M \# x \mid M \text{ is a TM which halts on } x \}$$

is L_h recursive, is it RE?

L_h is RE:

We can make a TM which simulates M,
so if M halts on x,
our simulation can be made to accept x.

So our simulation accepts any $M \# x$ if M halts on x.
so it accepts L_h.

L_h is not recursive

To get a contradiction, assume L_h is recursive,

i.e. we have a TM N

which accepts $L_h = \{ M \# x \mid M \text{ halts on } x \}$

and never loops.

so N can always give yes/no answer

whether M halts on x.

(what happens when we run N on $N \# N$?)

We can create some TM K so that

if input is $M \# M$, then

- K halts if M doesn't halt on M
- K doesn't halt if M does halt on M.

Now what happens with $K \# K$?

- K halts if K doesn't halt
- K doesn't halt if K halts.

This is a contradiction

No TM can decide whether or not another TM halts.

A interpreter/compiler will tell if you have errors in the code. deft

It won't tell you about if your code creates an infinite loop.

This would be impossible because no total TM can solve the halting problem.

L_h is RE ← we can make a TM to verify that a TM halts.

L_h is not recursive ← we can't make a TM that answers yes/no for whether a TM halts.

A difference between verifying an answer vs deciding the answer.

Unsolved Problem: P vs NP conjecture

/ can create the answer in polynomial time

\ can verify an answer in poly. time.

∃ $P = NP$?

We could actually improve a TM by adding a magical "halting oracle"

The TM can follow arrows as usual, or it can solve a halting problem.

Called a TM with halting oracle

Turns out, there are other languages that such a TM can't compute.

There's many other non-computable problems.