Lecture 25 : Halting and Membership Problem

December 9, 2020 10:26 AM

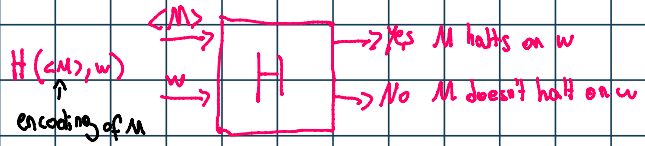
- Every language that is recursive is also recursively-enumerable.
- What about the other way around? can we always convert and get rid of infinite loops. (Answer is No)
  - To answer we want to find a language that is recursively enumerable but not recursive
- There are two problems to consider ① Halting Problem ② Membership problem

* Halting problem

- Given an input Turing machine M and a string w. Does M halt on input w?
  - This is only doable for some special cases. It's undecidable, No general procedure to solve this problem.

what are we trying to prove? The Halting problem is undecidable (not recursive).

- We start by assuming that it is decidable. (There is a Turing Machine that solves this and always halts call it H)



The initial tape content of H is the encoding of the other Turing Machine M and string w.

So $\langle M\rangle, w$. H will enter one state if M halts and another if M doesn't halt. H has to do this in finite time. It can't go in an infinite loop cuz thats the whole point.

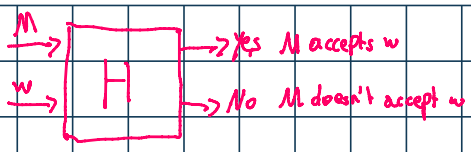
- We apply the opposite logic to constructing another machine H' that takes the same input $\langle M\rangle, w$.
  - if H returns yes, H' enters an infinite loop.
  - if H returns no, H' halts.
- Now construct $\hat{H}(\langle M\rangle)$ that runs H' with the input being the turing machine M and itself as a input string
  - so $\hat{H}(\langle M\rangle) \Rightarrow H'(\langle M\rangle, \langle M\rangle)$, which results in this logic for $\hat{H}$
    - if M halts on input $\langle M\rangle$, $\hat{H}$ loops forever
    - else it halts.
- Now we run $\hat{H}$ on itself instead of M so we get $\hat{H}(\langle \hat{H}\rangle)$
  - if $\hat{H}$ halts on $\langle \hat{H}\rangle$ then loop forever
  - else $\hat{H}$ halts. This makes no sense. Contradiction. So Halting problem is undecidable.

* The Membership problem

- Given an input Turing machine M and a string w. Does M accept w ($w \in L(M)$)?
  - This is only doable for some special cases. It's undecidable, No general procedure to solve this problem.
  - → we don't care why it's not being accepted.

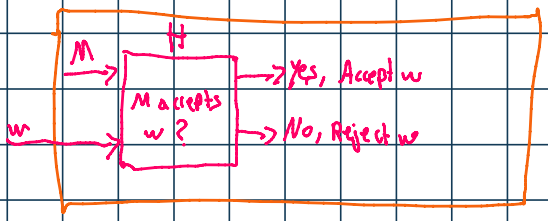
what are we trying to prove? The membership problem is undecidable (not recursive).

- We start by assuming that it is decidable. (There is a Turing Machine that solves this and always halts call it H)



. if such a TM exists then every recursively enumerable language is also recursive
. We already know that this isn't true from the counting argument so thats the contradiction we're going for.

. Let L be a recursively enumerable language.
. let M be a turing machine that accepts L. If this M halts on every input then the recursively enumerable language is also recursive. but what if it enters a loop ??
. Using our Assumption we prove that L is also recursive so we need to describe a TM that accepts L and always halts.
. To do that we need a decider TM for H that takes an input w. Then it feeds that input to H along with the encoding of M's description.



. Therefore L is also recursive, so every recursively enumerable language is found to be recursive. But we know that there exists recursively enumerable languages that are not recursive. So we get our contradiction. So the membership problem is undecidable.

* Important relationships between Recursive and RE languages.

① Every Recursive language is also Recursively enumerable.
② if L is recursive, then $\bar{L}$ is also recursive.
③ if both L and $\bar{L}$ are recursively enumerable then L is recursive. $RE \cap \overline{RE} = R$
④ There is a language L that is recursively enumerable but its complement $\bar{L}$ is not.

* Linear-Bounded Automatas (LBA)

. Similar to Turing machines except the space on the tape is restricted to only the input string's length. + left/right end-markers.
→ Its linear because if we have input of size n, we operate on a tape of size c·n, like multi-track tapes by extending alphabet by c.
. LBAs are defined to be non-deterministic.