Lecture 24 : Recursive and Recursively Enumerable languages

December 8, 2020 6:29 PM

- Are there any languages that are not accepted by any Turing Machine?

  How do we accept? $x \in L$ if and only if M halts on x in a final state

  $x \notin L$ if and only if either M halts on x in a non-final state OR M loops on x.

  such a language is called recursively enumerable, recognizable, semi-decidable, or simply accepted by TM.

- Countable sets:

→ Infinite sets are Countable or uncountable.

Countable can be if the set is finite OR countably infinite if we can put it in order. 1-to-1 correspondance with $\mathbb{N}$

- An enumeration procedure for S is a turing machine that generates all strings of S one by one in finite time.

  ↳ Given the set S, if there is an enumeration procedure for S then S is countable.

- We want to show that the set of all Turing Machines is countable.

→ we know that any turing machine can be represented/encoded with a binary string of 0s and 1s.

→ we now need an enumeration procedure for the set of TM strings.

① Generate binary strings of 0s and 1s in increasing order of length. (Gives us all possible binary strings)

② Check if the string describes a TM. (The transition functions from last lecture is encoded).

③ if yes, then print the string on output tape.

④ if not, go to next string loop again from ①

Since we were able to do this, then it is countably infinite.

- Uncountable sets:

→ A set is uncountable if it is not countable.

Let S be an infinite countable set, the powerset 2^S of S is uncountable.

Proof by contradiction: We assume that the powerset 2^S of an infinite countable set IS countable.

- Since S itself is countable, we can write its elements in order. $S = \{s_1, s_2, s_3, s_4, \dots\}$
- Elements of the powerset have the form $\{s_1, s_3\}$, $\{s_5, s_7, s_9, s_{10}\}$ ... (there infinite of them cuz S is infinite).
- Now we encode each element of the powerset with a binary string of 0s and 1s
- We assumed that the powerset is countable so we should be able to list its elements, all possible subsets.

- Now we encode each element of the powerset with a binary string of 0s and 1s
- We assumed that the powerset is countable so we should be able to list its elements, all possible subsets.

	Encoding				
powerset element	s_1	s_2	s_3	s_4	...
$\{s_1\}$	1	0	0	0	
$\{s_2, s_3\}$	0	1	1	0	
$\{s_1, s_3, s_4\}$	1	0	1	1	

What we just did - Assume for contradiction that the powerset is countable. Then we can enumerate the elements of the powerset and we want to show that we must have missed 1.

powerset element	Encoding				
t_1	1	0	0	0	0
t_2	1	1	0	0	0
t_3	1	1	0	1	0
t_4	1	1	0	0	1

Now we take the powerset element whose bits are the complement bits in the diagonal.
Take the diagonal, flip its bits, now we have a new element. So new element is 0011
By taking the complement, we are sure that this new element is different than t_1, t_2, t_3 and t_4
-> This new element must be some t_i of the powerset (By our assumption that its countable)
-> But we see that this is impossible because we want i to be equal to its complement. so its a contradiction
* So the powerset 2^S of S is uncountable.

-> Now we can apply this to our languages. For example All strings $S = \{a,b\}^*$, this is generated by the alphabet $\{a,b\}$. We know this is infinite and countable (cuz we can list all possible strings, has an enumeration procedure).
-> A language $L = \{aa, ab, aab\}$ is a subset of S. So the powerset 2^S contains all these possible languages.
-> We already showed that the powerset of this countable set is uncountable.
-> we also know that the set of all turing machines is countable. So there are more languages than TMs.

* Recursive language (decidable)
-> its a more practical sub-class of languages. We require the TM to always halt. No rejection by loop.
L is decidable if there is a TM M: $x \in L$ iff M halts on x in a final state
$x \notin L$ iff M halts on x in non-final state