The Lecture Title

Scribe: Your Name

Date: Day, Mon, Date Year

1 Computational Complexity

aka. what can computers do?

The *Chomsky Hierarchy* was an early way to formalize ordering the complexity of grammars, or collections of languages. Here is the hierarchy below:

We will look at how to determine how we can separate the hierarchies above. First, a bit of context:

There are four main hierarchies here:

- Regular expressions - like those seen in AI that can detect words of some type. Can be recognized by deterministic finite automata (DFAs).
- Context-Free Languages - like the palindrome question from two days ago, or stuff like $\{a^n b^n | n \in \mathbb{Z}\}$. These are recognizable by non-deterministic push-down automata (NPDAs).
- Context-Sensitive Languages - recognizeable by Turing Machines with linear space on the tape - for example, the language $\{a^n b^n c^n | n \in \mathbb{Z}\}$.
- Recursively Enumerable - Turing Machines can recognize.

We state that a Turing Machine (or anything else, really) can recognize a language iff on any input $x \in \mathcal{L}$, the Turing Machine will say "Yes" eventually, but if $x \notin \mathcal{L}$, the Turing Machine will either say "No" or run forever. This set of languages will be recursively enumerable.

What if we require the Turing Machine to halt? This is the *Halting Problem* that asks "Does $M(x)$ halt, for TM M and input x?"

We claim that this language is recursively enumerable. First of all, all the Turing Machines in the world are enumerable, as are the inputs (all equivalent to integers). The desired language $\mathcal{L} = \{(M, x) | TM(x)\, halts\}$, which we claim is recursively enumerable. If we accept that a Universal Turing Machine exists (UTM) that accepts this input, then clearly if we let the UTM run on this input, then this setup doing the problem shows that this must be recursively enumerable, because it either will say "Yes" when it halts and say "No" or run forever when it runs forever.

We define a *recursive* language as a language such that a Turing Machine acting on some $x \in \mathcal{L}$ will halt with Yes and if $x \not\in \mathcal{L}$ halts with "No".

Is the Halting Problem solved by a recursive language? We assume by contradiction that special machines TM_K exist such that when running on a list of all possible inputs x_i, that these machines will return whether or not the input will exist.

Now, construct a Turing machine TM_α such that when acting on a TM_K and x, such that when one of these special Turing Machine running on these things will return H if the internal Turing Machine will

This means that the set of recursively enumerable languages is strictly larger than the set of recursive languages. This also means that the

The Arithmetic Hierarchy is a hierarchy of languages that involve the problems $\exists$ and $\forall$.

It is unknown whether

NP and coNP.

Pratt's Theorem: The primality problem is in NP.