CS 361 Meeting 27 — 11/13/17

Announcements

1. Homework 9 is available.

A Language that is Neither Recognizable nor co-Recognizable

1. We have now identified specific examples of languages that are decidable, recognizable but not decidable, and not recognizable but have recognizable complements. I would like to round things out by showing a specific language such that neither the language nor its complement is recognizable. There are lots of languages for us to think about!

2. REALLY! LOTS!

3. If you think about the counting argument I discussed in our first class this semester, you will realize in that discussion, I suggested that there had to be functions that were not computable because there are uncountably infinitely many functions and only countably infinitely many programs (now known as Turing machines).

4. You may mistakenly think that the undecidable or unrecognizable languages we have identified are examples of these uncomputable functions I was discussing during our first class. They aren’t.

5. Consider how many languages there are that are recognizable (including the ones that are decidable).
   
   - Every such language can be “paired” with a Turing machine.
   - Every Turing machine can be encoded as a string over some preselected alphabet.
   - There are only countably infinitely many strings over any language.
   - This implies that there are only countably many recognizable languages (including all the undecidable ones).

6. A similar argument says that the elements of the set of all languages that are not recognizable but have recognizable complements can be paired with the TMs that recognize their complements. Therefore the set of all such languages is also countable.

7. Basically, we still haven’t identified a single language that belongs to what must be the largest category in our taxonomy — language that are not recognizable and have complements that are also not-recognizable.

8. Let’s start by trying to develop some intuition about the correct categories for the following eight languages in the hope that we might find some candidates for the languages that are neither recognizable or co-recognizable:

   - \( EQ_{CFG} = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are CFGs and } L(A) = L(B) \} \)
   - \( EQ_{CFG} = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are not CFGs or } L(A) \neq L(B) \} \)
   - \( ALL_{CFG} = \{ \langle M \rangle \mid M \text{ is a CFG and } L(A) = \Sigma^* \} \)
   - \( ALL_{CFG} = \{ \langle M \rangle \mid M \text{ is not a CFG or } L(A) \neq \Sigma^* \} \)
   - \( EQ_{TM} = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are TMs and } L(A) = L(B) \} \)
   - \( EQ_{TM} = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are not TMs or } L(A) \neq L(B) \} \)
   - \( ALL_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(A) = \Sigma^* \} \)
   - \( ALL_{TM} = \{ \langle M \rangle \mid M \text{ is not a TM or } L(A) \neq \Sigma^* \} \)

9. With a little thought (and luck) you should be able to guess that:

   - None of these examples are decidable.
   - \( EQ_{CFG} \) and \( ALL_{CFG} \) are probably recognizable since we can convert the grammars into Chomsky normal form and then enumerate all strings checking to see if some string is in one language but not the other (for \( EQ_{CFG} \)) or that a given string is not in \( ALL_{CFG} \) and accepting as soon as we find an example.
   - \( EQ_{CFG} \) and \( ALL_{CFG} \) are probably not recognizable (since if there is no counter-example, the procedure described in the preceding bullet point loops forever).
11. Let’s consider $ALL_{TM}$. Our hunch is that this language and its complement are both not recognizable. That means that proving they are undecidable won’t help much. That would tell us that one of them must not be recognizable, but we would not know which one was not recognizable or that both of them are actually not recognizable.

12. Instead, we will seek two separate reduction proofs that $ALL_{TM}$ and its complement are not recognizable.

13. To start, we will first assume that $\overline{ALL_{TM}}$ is recognizable and try to show that if this was true we could construct a TM to recognize some language like $\overline{A_{TM}}$ or $E_{TM}$ that we already know is not recognizable.

- We will use $\overline{A_{TM}} = \{(M, w) \mid M \text{ is not a TM or } w \notin L(M)\}$
- Since we will be using proof by contradiction, we begin by assuming that $\overline{ALL_{TM}}$ is recognized by some TM $M_{\overline{ALL}}$.
- Next, we construct a machine $M_{\overline{A_{TM}}}$ that will use $M_{\overline{ALL}}$ to recognize $\overline{A_{TM}}$. This machine behaves as follows:
  - Accept the input if it is not a valid TM + input description
  - On input $\langle M, w \rangle$, construct a description, $\langle M' \rangle$ of a TM $M'$ that behaves as follows:
    - On input $w'$, simulate $M$ on input $w$. Accept $w'$ if $M$ accepts $w$. Otherwise, reject.
    - Run $M_{\overline{ALL}}$ on $\langle M' \rangle$ and accept if it does.
- The way we construct $M'$ in our $M_{\overline{A_{TM}}}$ ensures that $L(M') = \Sigma^*$ exactly when $w \in L(M)$ and $L(M') = \emptyset$ exactly when $w \notin L(M)$. Therefore, $\langle M' \rangle \in \overline{ALL_{TM}}$ exactly when $w \notin L(M)$.

14. Next, we will assume that $ALL_{TM}$ is recognizable and try to show that if this was true we could construct a TM to recognize some language like $\overline{A_{TM}}$ or $E_{TM}$ that we already know is not recognizable.

15. This isn’t as easy as the first proof. A first guess would be that we could just reverse the way the $M'$ used in our argument for $ALL_{TM}$ behaved.

- That is given $\langle M, w \rangle$, construct a description, $\langle M' \rangle$ of a TM $M'$ that behaves as follows:
  - On input $w'$, simulate $M$ on input $w$. Accept $w'$ if $M$ rejects $w$. Otherwise, reject.
- Unfortunately, this machine cannot do what its description says (accept $w'$) in the case that $M$ rejects $w$ by looping because it can never be certain that $M$ is looping rather than just running for a long time.

16. We need to somehow turn $M$ looping into $M'$ accepting all strings. We can do this by making $M'$ accepts strings as long as $M$ takes more steps to decide how to handle $w$ than some function that grows without bounds as a function of $w$. Length of $w$ is such a function.

- That is given $\langle M, w \rangle$, construct a description, $\langle M' \rangle$ of a TM $M'$ that behaves as follows:
  - On input $w'$, simulate $M$ on input $w$ for $|w'|$ steps. Reject $w'$ if $M$ accepts $w$ within the period of simulation. Otherwise, accept $w'$.
- If $M$ rejects or loops on $w$, $L(M') = \Sigma^*$. If $M$ accepts $w$, then $L(M')$ is the finite set of strings whose length is less than the number of steps $M$ executes before accepting $w$.

17. The complete proof looks like:

- We will use $\overline{A_{TM}} = \{(M, w) \mid M \text{ is not a TM or } w \notin L(M)\}$
- Since we will be using proof by contradiction, we begin by assuming that $ALL_{TM}$ is recognized by some TM $M_{ALL}$. 


• Next, we construct a machine $M_{\overline{ATM}}$ that will use $M_{ALL}$ to recognize $\overline{ATM}$. This machine behaves as follows:
  – Accept the input if it is not a valid TM + input description
  – On input $\langle M, w \rangle$, construct a description, $\langle M' \rangle$ of a TM $M'$ that behaves as follows:
    * On input $w'$, simulate $M$ on input $w$ for $|w'|$ steps. Accept $w'$ if $M$ has not accepted $w$ during the simulation. Otherwise, reject.
  – Run $M_{ALL}$ on $\langle M' \rangle$ and accept if it does.

• The machine we have described will accept $\langle M, w \rangle$ iff $w \notin L(M)$ since if $w \notin L(M)$, $M'$ will accept all input since no matter how long the input is, simulating $M$ will not accept $w$ if simulated for as many steps as the length of the input. On the other hand, if $M$ does eventually accept $w$, then $M'$’s language will be finite.

18. It should be clear that the fact that neither $ALL_{TM}$ nor its complement is recognizable puts $EQ_{TM}$ in the same category.

• If $EQ_{TM}$ was recognizable by $M_{EQ_{TM}}$, we could use $M_{EQ_{TM}}$ to recognize $ALL_{TM}$ by building a machine that would take its input $\langle M \rangle$ and convert it into an input of the form $\langle M, M_{\Sigma^*} \rangle$ for $M_{EQ_{TM}}$ where $M_{\Sigma^*}$ is a machine that accepts $\Sigma^*$ by simply consisting of a single state that is both a start state and the accept state.

• We could do the same trick for $\overline{EQ_{TM}}$ by using a machine that could recognize $\overline{EQ_{TM}}$ to build a machine that could recognize $\overline{ALL_{TM}}$. 