CS 361 Meeting 27 — 11/14/16

Announcements
1. Homework 9 is available.

Just about Everything is Undecidable
1. It turns out that just about every interesting question about Turing machines is undecidable. We will eventually prove this in a general sense. One consequence of this is that just about half of all interesting questions are not recognizable (because if both a language and its complement are recognizable then both languages are decidable).

2. With this in mind, it should be easy to find some more languages to practice our reduction proof skills.

3. Here are just a few possible examples:
   - \( \text{REVERSIBLE}_{TM} = \{ \langle M \rangle | w \in L(M) \text{ iff } w^R \in L(M) \} \)
   - \( \text{REGULAR}_{TM} = \{ \langle M \rangle | L(M) \text{ is regular} \} \)
   - \( \text{DISJOINT}_{TM} = \{ \langle M, N \rangle | L(M) \cup L(N) = \emptyset \} \)
   - \( \text{PRIME}_{TM} = \{ \langle M \rangle | w \in L(M) \Rightarrow |w| \text{ is prime} \} \)

4. To give us one more bit of practice with this, let’s show that \( \text{REVERSIBLE}_{TM} \) is not recognizable.

   - First, observe that \( \{ab\} \) and \( \{a^n b^n | n \geq 0\} \) are nice examples of sets that are not reversible while \( \emptyset \) and \( \Sigma^* \) are definitely reversible.
   - We have seen several proofs in which the machine \( M' \) we built given an input \( \langle M, w \rangle \) accepted all inputs if \( M \) accepted \( w \) and rejected all inputs (possible by looping) otherwise.
   - It is easy to refine this machine by running another Turing machine as a sub-module after \( M' \) determines whether \( M \) accepts \( w \). For example, we could only accept strings of the form \( a^n b^n \) if \( M \) accepts \( w \).

   - This results in \( M' \in \text{REVERSIBLE}_{TM} \) exactly when \( w \notin L(M) \).
   - This gives us what we need to give a proof that \( \text{REVERSIBLE}_{TM} \) is not recognizable:

     **Proof:** Assume that \( \text{REVERSIBLE}_{TM} \) is recognized by some Turing machine \( M_{RTM} \). Construct a machine \( M'_{\overline{TM}} \) that operates as follows:

     (a) If the machine’s input is not a valid encoding of a Turing machine and its input, accept.

     (b) Otherwise,

        - construct a description of a new machine \( M' \) which behaves as follows:

          * On input \( w' \), simulate \( M \) on \( w \). If \( M \) accepts \( w \), then accept \( w' \) if it is a string of the form \( a^n b^n \).
          * Otherwise, reject.

        - Run \( M_{RTM} \) on \( \langle M' \rangle \) and accept if it does.

     The machine \( M'_{\overline{TM}} \) will accept \( \langle M, w \rangle \) exactly when \( w \notin L(M) \) since in this case, \( L(M') = \emptyset \) is reversible while otherwise \( L(M') = a^n b^n \) which is not regular.

5. It is just as easy to show that \( \text{REVERSIBLE}_{TM} \) is not recognizable.

6. We will stick with \( a^n b^n \) as our non-regular language. We will switch from \( \emptyset \) to \( \Sigma^* \) for our reversible language.

Computation Histories
1. Some very interesting proofs of undecidability rely on the technique of constructing a language that describes the possible computations of a TM on one or more inputs.

2. Recall that a TM configuration is a triple \( (q, u, v) \) with \( q \in Q \) representing the current state of the control, \( u \in \Gamma^* \) representing the contents of the tape to the left of the current head position, and \( v \in \Gamma^* \) representing the tape contents from the head to the right end of the non-blank tape.
3. We can use a sequence of strings that describe configurations to describe the complete computation of a TM.

**Definition:** Given a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ and a string $w \in \Sigma^*$, we define the language of computation histories of $M$ on $w$ as

$$L_{\text{Computation-history}}(M, w) = \{w_0w_1...w_n \mid \text{each } w_i \text{ is a configuration for } M \text{ \, \, } w_0 \text{ is the initial configuration for } w \text{ \, \, } w_n \text{ is a final/accept/reject configuration \, \, } \text{each } w_i \text{ yields } w_{i+1} \text{ according to } \delta \}$$

4. In general, $L_{\text{Computation-history}}(M, w)$ is a pretty complicated language. It is not regular or context-free. However, if we define a slight variation of this language

**Definition:** Given a TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$ and a string $w \in \Sigma^*$, we define the language of reversed computation histories of $M$ on $w$ as

$$L^R_{\text{Computation-history}}(M, w) = \{w_0w^R_1...w_n \mid \text{each } w_i \text{ is a configuration for } M \text{ \, \, } w_0 \text{ is the initial configuration for } w \text{ \, \, } w_n \text{ is a final/accept/reject configuration \, \, } \text{each } w_i \text{ yields } w_{i+1} \text{ according to } \delta \text{ \, \, } \text{every other } w_i \text{ is written backwards} \}$$

5. This language is still quite complex, but somewhat amazingly the complement of $L^R_{\text{Computation-history}}(M, w)$ is context-free.

- The trick is that we can describe a PDA that uses its nondeterminism to guess where the features that would make a string invalid as a computation history and then use the stack of the PDA to validate the guesses.

- The PDA guesses one of the following issues:
  - The whole string is not formatted properly (this is regular),
  - For any pair of consecutive configurations, the earlier configuration does not yield the following configuration (a PDA could nondeterministically guess which pair didn’t match, push the first configuration on its stack and then verify the mismatch as it read the next configuration),
  - The first configuration is not a valid initial configuration for $w$,
  - The last configuration is not a halting configuration.

6. Using this fact, we can show that $ALL_{CFG}$ is not recognizable by reducing $\overline{A_{TM}}$ to $ALL_{CFG}$. To do this, we will assume the existence of a TM $M_{ALL_{CFG}}$ that recognizes $ALL_{CFG}$ and build a TM $M_{\overline{A_{TM}}}$ that recognizes $\overline{A_{TM}}$.

- $M_{\overline{A_{TM}}}$ should accept its input if it is not a valid encoding of a TM description and input.

- On input $\langle M, w \rangle$, $M_{\overline{A_{TM}}}$ should construct a machine $M'$ that is like $M$ except it loops if $M$ would have entered the reject state.

- Next, $M_{\overline{A_{TM}}}$ should create a CFG $G_{\overline{M'}}$ for $L^R_{\text{Computation-history}}(M', w)$.

- Finally, $M_{\overline{A_{TM}}}$ should run $M_{ALL_{CFG}}$ on $\langle G_{\overline{M'}} \rangle$ and accept if $M_{ALL_{CFG}}$ accepts.

- This machine decides $\overline{A_{TM}}$ because $M'$ has a finite computation history on $w$ iff $w \in L(M)$. As a result, $L^R_{\text{Computation-history}}(M', w) = L(G_{\overline{M'}}) = \Sigma^*$ exactly when $w \notin L(M)$.

- Since it would be a contradiction if we could recognize $\overline{A_{TM}}$, it must be impossible to recognize $ALL_{CFG}$.

7. Since we can recognize $\overline{ALL_{CFG}}$ by simply checking all strings looking for one that cannot be derived from the grammar, this implies that $ALL_{CFG}$ is recognizable but not decidable.