Announcements

1. Homework 8 due Friday

Recursive, Recursively Enumerable, not even R.E. in review

1. Today, we will continue to explore how to categorize the computability of various languages.

2. We have been working with three categories: decidable languages, recognizable languages and languages that are not recognizable.

<table>
<thead>
<tr>
<th>Turing-Recognizable / Recursively Enumerable</th>
<th>Not Recognizable but Complement is R.E.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turing-Decidable / Recursive</td>
<td>CFS</td>
</tr>
<tr>
<td>Neither Recognizable nor Complement Recognizable</td>
<td>REG</td>
</tr>
</tbody>
</table>

3. Our first real accomplishment was to establish that the language

\[ A_{BTM} = \{ \langle M \rangle w \mid \langle M \rangle \text{ is a binary encoding of a binary TM, } \& \ w \in L(M) \}. \]

is recognizable but not decidable.

- We accomplished this using a diagonalization argument. We showed that assuming this language was decidable would make it possible to construct an inherently impossible Turing machine that differed from every Turing machine on some input.

- Let’s quickly review that argument:
  
  **Theorem:** \( A_{BTM} \) is undecidable.

  **Proof:** Suppose that \( A_{BTM} \) was decidable. Then there would exist some TM \( N \) that always halted such that \( A_{BTM} = L(N) \).

  - Given \( N \), we could construct another TM \( D \) which on any input \( w \), made a copy of \( w \) after its original input to form \( ww \) and then ran \( N \) on the result. This machine would decide the language
    
    \[ L(D) = \{ \langle M \rangle \mid \langle M \rangle \text{ is an encoding of a binary TM } \& \langle M \rangle \in L(M) \}. \]

  - Now, suppose that we alter \( D \) just a bit to produce a new machine named \( \overline{D} \). \( \overline{D} \) will be identical to \( D \) except its accept and reject states will be interchanged. Since all of these machines are deciders, we can say
    
    \[ L(\overline{D}) = \{ \langle M \rangle \mid \langle M \rangle \text{ is not an encoding of a binary TM or } \langle M \rangle \notin L(M) \}. \]

  - Now, consider what happens when we apply \( \overline{D} \) to its own description. That is, we apply \( \overline{D} \) to the input \( \langle D \rangle \). Since \( \langle D \rangle \) is clearly an encoding of a binary TM, we can see that \( \langle D \rangle \in L(\overline{D}) \iff \langle D \rangle \notin L(D) \).

  - This is nonsense! Or better yet a contradiction. As a result, we can state that our original assumption that \( A_{BTM} \) was decidable must be false.

4. While I rationally accept the correctness of the preceding argument, it always induces a lingering sense of confusion in my brain, so I want to
off what I find to be a more concrete way to understand $D$ (which may still induce a sense of confusion).

5. We can imagine an infinite table whose rows correspond to TMs in some ordering and whose columns correspond to binary encodings of these same Turing machines.

$$
\begin{array}{ccccccc}
M_1 & M_2 & M_3 & M_4 & M_5 & M_6 & \ldots \\
\text{reject} & \text{reject} & \text{reject} & \text{reject} & \text{reject} & \text{reject} & \ldots \\
M_2 & \text{reject} & \text{reject} & \text{reject} & \text{reject} & \text{reject} & \ldots \\
M_3 & \text{accept} & \text{reject} & \text{accept} & \text{accept} & \text{accept} & \ldots \\
M_4 & \text{accept} & \text{accept} & \text{accept} & \text{accept} & \text{accept} & \ldots \\
M_5 & \text{reject} & \text{reject} & \text{reject} & \text{reject} & \text{reject} & \ldots \\
M_6 & \text{reject} & \text{accept} & \text{reject} & \text{accept} & \text{reject} & \ldots \\
M_7 & \text{accept} & \text{accept} & \text{reject} & \text{reject} & \text{accept} & \ldots \\
M_8 & \text{reject} & \text{accept} & \text{accept} & \text{reject} & \text{accept} & \ldots \\
\end{array}
$$

Each cell indicates whether the TM for the row accepts the input corresponding to the encoding of the TM for that column.

6. The machine $D$ we described in our proof that $A_{BTM}$ is undecidable corresponds to the list of accept/reject results listed along the diagonal of this table (all shown in bold font).

7. If the machine $\overline{D}$ we described in our proof by contradiction existed, its language $L(D)$ would be described by the opposite of the sequence of “reject”s and “accept”s found along the diagonal of this table. Thus, it will be different from every row of the table in at least one position, but all TMs are included in the rows of this table so no such TM could exist.

Reduction Proofs

1. Fortunately, to establish that other languages are undecidable and/or unrecognizable, we don’t have to repeat the diagonalization process. Instead, we use two other approaches.

   - The most important technique is a combination of problem reduction and proof by contradiction. If we can show by assuming that some language is decidable that another language that is decidable and the second language is known not to be decidable, then we can conclude that the original language is not decidable. (The same approach works if we replace all the “decidable”s in this paragraph with “recognizable”.)

   - The other technique is to exploit the fact that if both a language and its complement are recognizable, they both must be decidable. Given this fact, we can show that a language is not recognizable by showing that its complement is recognizable and that either the language or its complement is not decidable.

2. We used the first technique together with the knowledge that $A_{BTM}$ is undecidable to show that $A_{TM}$ is undecidable:

   - Suppose that $A_{TM}$ were decidable. In this case, there would be some machine $M$ that decided $A_{TM}$.
   - Suppose we constructed a machine $M'$ with a binary input alphabet that first rejected its input if it was not a valid binary encoding of a binary TM and its input, $\langle B \rangle u$. Otherwise, it would write the translation of this input in the encoding used by $M$ on its tape and then run $M$.
   - Since we assumed $M$ decided $A_{TM}$, $M'$ would have to decide $A_{BTM}$. We just proved, however, that it is impossible to decide $A_{BTM}$. Based on this contradiction, we can conclude that no decider for $A_{TM}$ exists. That is, we have shown that $A_{TM}$ is undecidable.

3. Note that this proof has two important components:

   (a) A scheme for translating an input for a machine that decides (recognizes?) a language we know to be undecidable into an input for a machine that decides a language we wish to prove undecidable.
   (b) The running of a purported decider for the language we wish to provide undecidable on the translated input.

4. We applied almost the same approach to show that

$$E_{TM} = \{ \langle M \rangle \mid \langle M \rangle \text{ is a TM description and } L(M) = \emptyset \}$$
is not recognizable.

- First, we argued that \( E_{TM} \) was clearly a recognizable language since given a valid Turing machine description, we could dovetail a simulation of its execution on all inputs and if there was any input in the machine’s language we would eventually find it and accept the machine’s description.

- Given that if a language and its complement are both recognizable, the language must be decidable, this gives us two approaches to showing that \( E_{TM} \) is not recognizable:
  - We could show that \( E_{TM} \) is not decidable using a reduction argument, or
  - We could show that \( E_{TM} \) is not decidable (or not recognizable) using a reduction argument.

We took the later approach. We showed that if we could recognize \( E_{TM} \) then it would be possible to recognize \( A_{TM} \), a language we know is not recognizable because its complement, \( A_{TM} \) is not decidable.

**Proof:**

- Assume that \( E_{TM} \) is recognizable. In this case, there must be some Turing machine \( M_E \) that recognizes \( E_{TM} \).

- Given \( M_E \) construct a machine \( M_{ATM} \) as follows:
  - On input \( \gamma \), the machine will first check that \( \gamma \) is a valid encoding of a Turing machine and its input. That is, \( \gamma = < M, w > \) for some \( M \) and \( w \). If this is not the case, the machine accepts \( \gamma \).
  - Assuming the input is of the form \( < M, w > \), the machine will construct on its tape a description of a new Turing machine \( M' \) that behaves as follows.
    - \( M' \) will ignore the contents of its input, \( w' \), and instead begin simulating \( M \) on input \( w \).
    - If \( M \) eventually accepts \( w \), then \( M' \) will accept \( w' \).
  - Once the description of \( M' \) is written on the tape, \( M_{ATM} \) will run \( M_E \) on the description of \( M' \). If \( M_E \) accepts \( < M', w > \), then \( M_{ATM} \) will accept \( < M, w > \).

- The language accepted by the \( M' \) whose description is constructed by \( M_{ATM} \) will be \( \Sigma^* \) if \( w \in L(M) \) and will be \( \emptyset \) if \( w \notin L(M) \). Therefore, \( M_E \) will accept the input \( < M' > \) and \( M_{ATM} \) will accept the input \( < M, w > \) exactly when \( w \notin L(M) \). That is, \( L(M_{ATM}) = A_{TM} \). We know, however, that this is impossible since \( A_{TM} \) is not recognizable. Therefore, our assumption that \( E_{TM} \) was recognizable must be false.

5. It is important to identify the distinct roles of the four machines \( M_E \), \( M \), \( M' \) and \( M_{ATM} \) involved in this proof.

- \( M_E \) is what we assume to exist based on the (false) assumption that \( E_{TM} \) is recognizable. We do not assume anything about the structure of this machine.

- \( M_{ATM} \) is a machine we describe how to construct that would accomplish something we know is impossible to do by exploiting the (false) assumption that \( M_E \) exists. In most proofs of this form, the last thing the impossible machine we are trying to construct \( (M_{ATM} \text{ in this case}) \) will do is run \( M_E \) (or its equivalent).

- \( M \) isn’t really A machine. It can actually be any machine. We are going to try to show that we could construct \( M_{ATM} \) if \( M_E \) existed. \( M_{ATM} \) takes an input composed of the description of a Turing machine and of a possible input to that Turing Machine. \( M \) is the name we use to refer to the machine described in some input to \( M_{ATM} \) as we try to describe how \( M_{ATM} \) would process its input.

- \( M' \) is a machine we never actually run or construct. Instead we argue that \( M_{ATM} \) would be able to construct a description of \( M' \) (i.e., \( < M' > \)) on its tape. The form of this machine depends on the input to \( M_{ATM} \) in such a way that \( < M' > \) will belong to the language of \( M_E \) exactly when the input to \( M_{ATM} \) belongs to \( A_{TM} \).

**Too Many Languages!**

1. Up until now, we have identified three classes of languages relative
to the power of Turing machines: decidable, recognizable, and non-
recognizable.

2. The two examples of non-recognizable languages we have identified are
complements of recognizable languages.
• Such languages are identified as co-RE or “complement of recursively enumerable” languages.

3. An interesting question is whether there are any languages that cannot be recognized by Turing machines whose complements are also unrecognizable.
• If you think about our counting argument from the first class, you should realize that most languages must fall in this category since there are only countably many Turing machines and countably many recognizable languages.

4. We would like to go find an example of a language that fits in this last category.

5. There are lots of languages for us to think about!

6. We can start with the following four “language templates”:
• \( A_{???} = \{ \langle M, w \rangle | M \text{ is a } ??? \text{ and } w \in L(M) \} \)
• \( E_{???} = \{ \langle M \rangle | M \text{ is a } ??? \text{ and } L(M) = \emptyset \} \)
• \( EQ_{???} = \{ \langle A, B \rangle | A \text{ and } B \text{ are } ???s \text{ and } L(A) = L(B) \} \)
• \( ALL_{???} = \{ \langle M \rangle | M \text{ is a } ??? \text{ and } L(A) = \Sigma^* \} \)

7. Then we can fill in the question marks with all of our favorite schemes for describing languages
• \( A_{DFA} = \{ \langle M, w \rangle | M \text{ is a DFA and } w \in L(M) \} \)
• \( E_{DFA} = \{ \langle M \rangle | M \text{ is a DFA and } L(M) = \emptyset \} \)
• \( EQ_{DFA} = \{ \langle A, B \rangle | A \text{ and } B \text{ are DFAs and } L(A) = L(B) \} \)
• \( ALL_{DFA} = \{ \langle M \rangle | M \text{ is a DFA and } L(A) = \Sigma^* \} \)
• \( A_{CFG} = \{ \langle M, w \rangle | M \text{ is a CFG and } w \in L(M) \} \)
• \( E_{CFG} = \{ \langle M \rangle | M \text{ is a CFG and } L(M) = \emptyset \} \)
• \( EQ_{CFG} = \{ \langle A, B \rangle | A \text{ and } B \text{ are CFGs and } L(A) = L(B) \} \)
• \( ALL_{CFG} = \{ \langle M \rangle | M \text{ is a CFG and } L(A) = \Sigma^* \} \)
• \( A_{TM} = \{ \langle M, w \rangle | M \text{ is a TM and } w \in L(M) \} \)
• \( E_{TM} = \{ \langle M \rangle | M \text{ is a TM and } L(M) = \emptyset \} \)
• \( EQ_{TM} = \{ \langle A, B \rangle | A \text{ and } B \text{ are TMs and } L(A) = L(B) \} \)
• \( ALL_{TM} = \{ \langle M \rangle | M \text{ is a TM and } L(A) = \Sigma^* \} \)

8. If that is not enough, we can consider the complements of each of these languages!

9. The actual proofs that languages are decidable, recognizable, not recognizable, etc. are often subtle (involving double and triple negatives at time).

10. Before undertaking such a proof, it is best to make sure one has a good intuition that you are trying to prove the right thing. That is, it helps to make an educated guess what category a language will belong to first.
11. Before we look at more of the details of the reduction techniques that will enable us to verify our intuition about the difficulty of recognizing or deciding a language, I would like to practice the process of intuitively categorizing languages by “guessing” which category fits each of the following languages best.

12. As our examples, consider the following eight languages:

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