Announcements

1. Homework 9 due Friday
2. Delayed office hours again today.

Mapping Reductions

1. We have observed that many of the proofs of undecidability and non-recognizability we have explored have a very similar structure.

2. We can formalize these similarities in the notion of mapping reducibility and then use this idea to “simplify” the proofs for many results involving decidability and recognizability.

3. First we must define the idea of a computable function.

**Definition:** A function $f$ from $\Sigma^* \rightarrow \Sigma^*$ is *computable* if and only if some Turing machine $M$ on every input $w$, halts with $f(w)$ on its tape.

4. This definition is mainly an admission that Turing machines can do interesting things other than just accept and reject.

   - This is not new. One of the first TMs we considered implemented a computable function. It took input strings and did its best to insert a # in the middle of them.
   - In each of the non-recognizability proofs we have given, we have embedded such a computable function. Namely, the computation that generated $\langle M' \rangle$ given some $\langle M, w \rangle$.

5. Given the notion of computable functions, we can capture the essence of what our $M'$s are really about.

**Definition:** Language $A$ is *many-to-one reducible* to language $B$ (written $A \leq_m B$) if there exists a computable function $f : \Sigma^* \rightarrow \Sigma^*$ such that for every $w \in \Sigma^*$

$$w \in A \Leftrightarrow f(w) \in B.$$

In this case we call $f$ a reduction.

6. Let me share two handy memory aids for dealing with the notation.

   (a) The point of the $\leq$ goes in the direction opposite the function arrow.
   (b) It helps to read the $\leq$ as “is easier than” rather than “is less than”.

7. The following diagram illustrates what this definition requires and allows.

   - It must map members of $A$ to members of $B$.
   - It must map strings that are not in $A$ to strings that are not in $B$.
   - It can map multiple input strings to the same output.

8. As a simple example, the computable function

   $$f$$
\( f(<M, w>) = <M'> \) where \( M' \) is a TM that ignores its input, runs \( M \) on \( w \) and accepts its input if \( M \) accepts \( w \).

shows that \( A_{TM} \leq_m ALL_{TM} \) since it maps any \( <M, w> \) that belongs to \( A_{TM} \) to an \( <M'> \) that belongs in \( ALL_{TM} \).

9. Note that there is nothing that inherently ties this computable function to \( A_{TM} \) and \( ALL_{TM} \). As long as we can identify two languages \( A \) and \( B \) such that \( w \in A \iff f(w) \in B, A \leq_M B \). As an easy example of this, note that this function also shows that \( A_{TM} \leq_m ALL_{TM} \).

10. As a more diverse example, this function also shows that \( A_{TM} \leq_m E_{TM} \) since it maps any \( <M, w> \) that belongs in \( A_{TM} \) to an \( <M'> \) that belongs in \( E_{TM} \).

11. With these definitions, we can succinctly formalize the technique we have been using in all our proofs for the last few classes:

**Theorem:** If \( A \leq_m B \) and ...

(a) \( B \) is decidable, then \( A \) is decidable,
(b) \( A \) is undecidable, then \( B \) is undecidable,
(c) \( B \) is recognizable, then \( A \) is recognizable, and
(d) \( A \) is not recognizable, then \( B \) is not recognizable

- We won’t give a detailed proof of these claims, but they are all just obvious applications of the proof techniques we have been employing.

12. As an example of how we might use such a mapping reduction, consider the language

\[ DISJOINT_{TM} = \{ <M, N> | \ M \& N \text{ are TMs and } L(M) \cap L(N) = \emptyset \} \]

- We will show that \( DISJOINT_{TM} \) is not recognizable by showing that \( E_{TM} \leq_M DISJOINT_{TM} \).

- To do this, we need a mapping that will take any \( <M> \) to a pair of TM descriptions \( <N, N'> \) in such a way that \( L(N) \) and \( L(N') \) are disjoint if and only if \( L(M) \) is empty.
- Let \( ACCEPT \) be a TM that accepts all strings.
- Consider the function \( f(<M>) = <M, ACCEPT> \).
- This is clearly computable.
- It is also clear that \( <M, ACCEPT> \in C \) if and only if \( \langle E \rangle \in E_{TM} \).
- Given that we know that \( E_{TM} \) is not recognizable, we can conclude that \( DISJOINT_{TM} \) is not recognizable.

**Rice’s Theorem**

1. While we have categorized a large number of languages as decidable, recognizable, or not recognizable, there are still plenty of additional examples we could consider:

- \( REGULAR_{TM} = \{ <M> | L(M) \text{ is regular } \} \)
- \( CONTEXT-FREE_{TM} = \{ <M> | L(M) \text{ is context-free } \} \)
- \( PALINDROME_{TM} = \{ <M> | \ w \in L(M) \iff w^R \in L(M) \} \)
- \( EVEN_{TM} = \{ <M> | \ w \in L(M) \Rightarrow \ |w| = 2 \} \)
- \( PRIME_{TM} = \{ <M> | \ w \in L(M) \Rightarrow \ |w| \text{ is prime } \} \)

2. There is a single theorem that will quickly allow us to show that all of the languages listed above are undecidable.

- This result is known as Rice’s Theorem.
- It is presented as an exercise in Sipser (including a solution). We will consider its proof and application today.

3. Informally, Rice’s Theorem says that any nontrivial property of a Turing machine’s language is undecidable.

- Nontrivial means that the languages of some but not all TMs have this property.
• The fact that it is a property of the language rather than the TMs means that it must be based strictly on the set of strings a given TM accepts rather than on how the TM is designed or operates.

4. We can formalize this notion as:

**Rice’s Theorem.** Suppose that $L$ is a language with

$$\emptyset \subset L \subset \{\langle M \rangle \mid \langle M \rangle \text{ is a valid Turing machine}\}$$

such that if $L(M) = L(N)$ then $\langle M \rangle \in L \iff \langle N \rangle \in L$ then $L$ is undecidable.

5. Recall that the set of decidable sets is closed under complement. So if $L$ is decidable, then $\overline{L}$ is also decidable and vice versa. Therefore, we could restate the theorem as:

**Rice’s Theorem’.** Suppose that $L$ is a language with

$$\emptyset \subset L \subset \{\langle M \rangle \mid \langle M \rangle \text{ is a valid Turing machine}\}$$

such that if $L(M) = L(N)$ then $\langle M \rangle \in L \iff \langle N \rangle \in L$ then $L$ and $\overline{L}$ are undecidable.

6. This minor change is useful, because it allows us to add an assumption about $L$ that will make the proof easier without reducing the strength of the theorem. In particular, we would like to assume that Turing machines whose languages are empty are in $\overline{L}$ rather than $L$.

**Rice’s Theorem:** Suppose that $L$ is a language with

$$\emptyset \subset L \subset \{\langle M \rangle \mid \langle M \rangle \text{ is a valid Turing machine}\}$$

such that if $L(M) = L(N)$ then $\langle M \rangle \in L \iff \langle N \rangle \in L$ and $L(M) = \emptyset \Rightarrow \langle M \rangle \notin L$, then $L$ and $\overline{L}$ are undecidable.

7. The proof of this theorem is a good opportunity to utilize the notion of mapping reducibility we discussed last class. In particular, to prove this theorem we just need to show that some undecidable set like $A_{TM}$ is reducible to the $L$ in the statement of the theorem. That is, we must show that

$$A_{TM} \leq_m L$$

8. To do this, we need to find a computable function $f(\langle M, w \rangle) = \langle M' \rangle$ such that

• if $w \in L(M)$ then $\langle M' \rangle \in L$, and
• if $w \notin L(M)$ then $\langle M' \rangle \notin L$

9. Let $\langle M_{inL} \rangle$ be any member of $L$. Since we assumed that $L$ was not empty we know such a machine description must exist.

10. Define $f$ as follows. Given $\langle M, w \rangle$, $f$ will produce a description of a machine $M'$ that first ignores its input $w'$ and simulates $M$ on $w$ and loops if $M$ either loops or rejects. If, however, the simulation of $M$ would have accepted $w$, $M'$ will then run $M_{inL}$ on $w'$ and accept or reject $w'$ as $M_{inL}$ would.

11. If $w \notin M$, then $L(M') = \emptyset$ and therefore $f(\langle M, w \rangle) \notin L$ as desired.

12. if $w \in M$, then $L(M') = L(M_{inL}) \Rightarrow \langle M' \rangle \in L$.

13. This completes the proof that $A_{TM} \leq_m L$ and therefore shows that neither $L$ nor $\overline{L}$ is decidable.