1. Recall that \( L_u = \{ \langle M \rangle \# w \mid M(w) \text{ accepts} \} \) is not decidable. In class we showed reductions from \( L_u \) to various languages \( L \) to show that \( L \) was undecidable. One of the languages shown undecidable was the “halting language” \( L_{\text{halt}} = \{ \langle M \rangle \mid M \text{ halts on blank input} \} \).

In order to show a language \( L \) is undecidable, it is often just as easy, or even easier, to show a reduction from \( L_{\text{halt}} \) to \( L \).

**Example:** We show that \( L_{1^*} = \{ \langle M \rangle \mid L(M) = 1^* \} \) is not decidable by showing \( L_{\text{halt}} \leq L_{1^*} \).

**Reduction:** We show how a decider for \( L_{1^*} \) could be used to decide \( L_{\text{halt}} \). The reduction takes an instance of \( L_{\text{halt}} \) (i.e., a TM \( M \) that we’d like to know if it halts on blank input) and outputs an instance of \( M' \) for \( L_{1^*} \).

We need the following to be true: \( M \text{ halts on blank input} \) if and only if \( L(M') = 1^* \).

Here is the (partial) code for \( M' \). Fill in the blank, and then answer parts (a), (b), and (c).

\[
M'(x: \text{ string}) \\
\quad \text{run } M \text{ until, if ever, it halts} \\
\quad \text{if } M \text{ halted, then accept } x \text{ iff } x \ ___________
\]

(a) If \( M \) doesn’t halt when run on blank input, what is \( L(M')? \) ___________

(b) If \( M \) halts when run on blank input, what is \( L(M')? \) ___________

(c) Briefly argue that no decider for \( L_{1^*} \) can exist.

2. Let \( L_{\text{even}} = \{ \langle M \rangle \mid L(M) = \{ w : |w| \text{ is even} \} \}. \)

Prove that \( L_{\text{even}} \) is not decidable by showing that \( L_{\text{halt}} \leq L_{\text{even}} \).

3. Let \( L_h = \{ \langle M \rangle \# w \mid M(w) \text{ halts} \} \). Show how to use a decider for \( L_h \) to build a decider for \( L_u \).

Prove that the following languages are undecidable using Rice’s Theorem:

**Rice’s Theorem.** The language \( \{ \langle M \rangle \mid L(M) \text{ satisfies property } P \} \) is undecidable for any property \( P \) that is satisfied by at least one, and not all, recursively enumerable languages.

1. \( \text{ACCEPTRegular} := \{ \langle M \rangle \mid L(M) \text{ is regular} \} \)

**Example solution:** We need to show that the property of being regular is satisfied by at least one, but not all, r.e. languages, and then Rice’s theorem will be applicable. Clearly for any regular language \( R \) there is a TM \( M \) such that \( L(M) = R \), so the property of being regular holds for at least one r.e. language (namely, \( L(M) \)). Just as clearly, there is a TM \( M' \) that accepts \( \{ 0^n1^n \mid n \geq 0 \} \), a nonregular language, so \( L(M') \) is an r.e. language that is not regular. Since at least one, but not all, r.e. languages satisfy the property of being regular, we can apply Rice’s theorem and conclude that \( \text{ACCEPTRegular} \) is not decidable.

2. \( \text{ACCEPTILLINI} := \{ \langle M \rangle \mid \text{ILLINI} \in L(M) \} \)

3. \( \text{ACCEPTPALINDROME} := \{ \langle M \rangle \mid M \text{ accepts at least one palindrome} \} \)

4. \( \text{ACCEPTTHREE} := \{ \langle M \rangle \mid M \text{ accepts exactly three strings} \} \)

5. \( \text{ACCEPTUNDECIDABLE} := \{ \langle M \rangle \mid L(M) \text{ is undecidable} \} \)