Sample problems on reductions and undecidability

I promised that I would give you a few sample problems on undecidability, along with their solutions, to aid in your preparation for the second exam. Here they are.

As usual, you should assume that some alphabet Σ, which you can take to be $\Sigma = \{0, 1\}$ if you like, has been fixed, and that for an arbitrary DTM M (with any number of elements in its input alphabet), the string $\langle M \rangle \in \Sigma^*$ denotes an encoding of M relative to some fixed encoding scheme.

Problem 1. Define the language

$$\text{COMP} = \Big\{ \langle M_1, M_2 \rangle \,:\, M_1 \text{ and } M_2 \text{ are DTMs with } L(M_1) = \overline{L(M_2)} \Big\}.$$

(COMP stands for "complement.") Prove that COMP is not Turing-recognizable.

Solution. One of the easiest ways to prove something is not decidable or not Turing-recognizable is to establish a mapping reduction from some other language that you already know to be undecidable or non-Turing-recognizable.

In this particular case, there is a simple reduction from the language

$$\text{E} = \{ \langle M \rangle \,:\, M \text{ is a DTMs with } L(M) = \varnothing \},$$

which we proved in class is not Turing-recognizable. Define a function

$$f(\langle M \rangle) = \langle M, M_{\text{all}} \rangle,$$

where M_{all} is some fixed DTM that accepts all strings over the input alphabet of M. It is trivial to compute a description $\langle M_{\text{all}} \rangle$ of such a DTM from the description of M, and so f is a computable function. We have

$$\overline{L(M_{\text{all}})} = \varnothing,$$

and so it is immediate that $\langle M \rangle \in \text{E}$ if and only if $f(\langle M \rangle) \in \text{COMP}$. (As always, we assume that f is defined appropriately so that $x \in \text{E} \Leftrightarrow f(x) \in \text{COMP}$ when the input x does not encode a DTM at all. We usually don't even bother mentioning this sort of case because it is obvious that f should be defined like this.) We have shown that f is a reduction from E to COMP, and so

$$\text{E} \leq_m \text{COMP}.$$

As E is not Turing-recognizable, we have that COMP is also not Turing-recognizable, as required.

Problem 2. Define the language

$$\text{MNH} = \{ \langle M \rangle \,:\, M \text{ is a DTM, and there is some string } x \text{ on which } M \text{ runs forever} \}.$$

(MNH stands for "might not halt.") Prove that MNH is not Turing-recognizable.

Solution. We can follow the same strategy as in the previous question, but using a different language. There are several reasonable choices, one being the complement of this language:

$$\text{HALT} = \{ \langle M, x \rangle \,:\, M \text{ is a DTM that halts on input } x \}.$$

We know that HALT is not decidable, but it is Turing-recognizable, and this implies that $\overline{\text{HALT}}$ is not Turing-recognizable.

Our goal will be to prove $\overline{\text{HALT}} \leq_m \text{MNH}$, which is equivalent to $\text{HALT} \leq_m \overline{\text{MNH}}$. For every DTM M and input x, let us define a DTM M_x as follows:

On input y: run M on x.

In other words, M_x completely ignores its input string y and runs M on x. (We used precisely this DTM construction in Lecture 12.) If we define a function

$$f(\langle M, x\rangle) = \langle M_x\rangle$$

then we have that f is computable—it is easy, in principle, to come up with a description of M_x given a description of both M and x.

Now we have that $\langle M, x\rangle \in \text{HALT}$ implies that M_x halts on all inputs, and $\langle M, x\rangle \notin \text{HALT}$ implies that M_x runs forever on every input (and therefore runs forever on some input string). Thus,

$$\langle M, x\rangle \in \text{HALT} \Leftrightarrow f(\langle M, x\rangle) \in \overline{\text{MNH}}.$$

We have proved that $\text{HALT} \leq_m \overline{\text{MNH}}$, and so MNH is not Turing-recognizable.

Problem 3. Define a language

$$\text{APAL} = \Big\{\langle M\rangle \,:\, M \text{ is a DTM accepting at least one palindrome } x = x^{\text{R}} \text{ over its input alphabet}\Big\}.$$

Prove that APAL is undecidable.

Solution. If you actually saw a problem like this on the exam, you would be very happy, at least if you remembered Rice's theorem (from Lecture 11). Specifically, define $\mathcal{L}$ to be the class of all languages that contain at least one palindrome. There are obviously examples of Turing-recognizable languages A and B such that $A \in \mathcal{L}$ and $B \notin \mathcal{L}$ (such as A being the language of all palindromes over some alphabet and B being $\overline{A}$). Rice's theorem then implies that $\{\langle M\rangle \,:\, L(M) \in \mathcal{L}\} = \text{APAL}$ is not decidable.

In the interest of getting some more practice, let's pretend that we don't know about Rice's theorem, and answer the question using a reduction. In this particular case, it is not hard to see that APAL is Turing-recognizable, because we may dovetail over all palindromes and running times to search for a palindrome accepted by a given M. What this means is that you shouldn't bother trying to reduce some non-Turing-recognizable language to APAL, because it will never work.

Let us reduce

$$\text{A} = \{\langle M, x\rangle \,:\, M \text{ is a DTM that accepts input } x\}$$

to APAL. This is a reasonable thing to try because we know that A is Turing-recognizable, but not decidable. This turns out to be easy using exactly the same function

$$f(\langle M, x\rangle) = \langle M_x\rangle$$

used in the previous question, with M_x being defined the same way as before. We have that $\langle M, x\rangle \in \text{A}$ implies that $L(M_x) = \Sigma^*$, which of course contains at least one palindrome. Therefore $f(\langle M, x\rangle) \in \text{APAL}$ in this case. On the other hand, if $\langle M, x\rangle \notin \text{A}$, then $L(M_x) = \varnothing$, which does not contain a palindrome, so $f(\langle M, x\rangle) \notin \text{APAL}$ in this case. Therefore

$$\langle M, x\rangle \in \text{A} \Leftrightarrow f(\langle M, x\rangle) \in \text{APAL}$$

and so we have shown that $\text{A} \leq_m \text{APAL}$. This implies that APAL is undecidable.

Here are a few more problems for you to try on your own.

Problem 4. Define a language

$$\text{OPAL} = \Big\{\langle M\rangle \;:\; M \text{ is a DTM that } \underline{\text{only}} \text{ accepts palindromes over its input alphabet}\Big\}.$$

Prove that OPAL is not Turing-recognizable.

Problem 5. Define a language

$$\text{NEI} = \{\langle M_1, M_2\rangle \;:\; M_1 \text{ and } M_2 \text{ are DTMs with } L(M_1) \cap L(M_2) \neq \varnothing\}.$$

Prove that NEI is not decidable.

Problem 6. Define a language

$$\text{TRIV} = \{\langle M\rangle \;:\; M \text{ is a DTM with } L(M) = \varnothing \text{ or } L(M) = \Sigma^*\}.$$

In other words, $\text{TRIV} = \text{E} \cup \text{ALL}$. Prove that TRIV is not Turing-recognizable.

Hint. You should not try to leverage the fact that $\text{TRIV} = \text{E} \cup \text{ALL}$ to find a shortcut. Just find a reduction from some non-Turing-recognizable language to the language TRIV as it is defined.