Undecidable Proof Strategies

So far, we have learned about undecidability and the *reduction* technique which allows us to show that a problem is solvable given another problem. In this unit, we use reductions to establish that one problem is undecidable by showing that it reduces to another problem known to be undecidable. We have some intuition of what a reduction is—solving the second problem by using a helper function that solves the first problem. However, it is not straightforward to map that intuition into a strategy for building reducibility proofs. In this reading, I outline a particular strategy for developing these proofs so that they are less magical and more mechanical.

An Example Reduction

Consider the following language:

$$A = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \Sigma^*\}$$

To prove that this language is undecidable, we will reduce a known, undecidable language to it. A common choice that we'll use for this problem is A_{TM}:

$$A_{\mathsf{TM}} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$$

Thus our goal is to show that $A_{\mathsf{TM}} \leq A$, that is, A_{TM} is reducible to A.

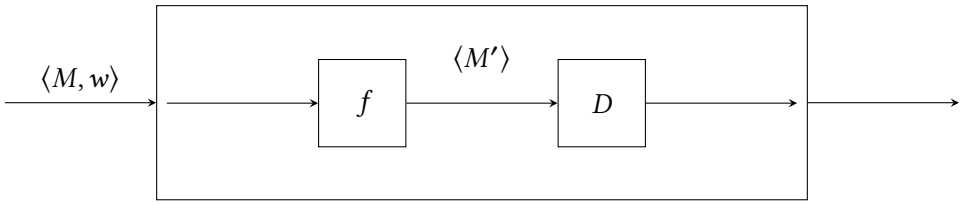
Recall that (mapping) reducibility says that to establish this fact, we need to create a function f that maps a string w in A_{TM} to a string in A such that:

1. f itself is decidable (*i.e.*, never goes into an infinite loop on any input) and
2. $w \in A_{\mathsf{TM}} \iff f(w) \in A$

A_{TM} accepts TM descriptions and inputs $\langle M, x \rangle$ and A accepts TM descriptions $\langle M \rangle$. Therefore, our function can be written as $f(\langle M, w \rangle) = \langle M' \rangle$ where:

$$M \text{ accepts } w \iff L(M') = \Sigma^*.$$

Note that all we need to do is create this function f to complete the proof. This is because we are using f in a standard way to build a decider for A_{TM} given a decider D for A:



Our decider for A_{TM}, the outer-most box, proceeds by transforming its input using f and then runs that output through D. The decider then outputs whatever D outputs. This decides A_{TM} but is a contradiction because we know A_{TM} is not decidable. Therefore, we can conclude that D must not exist, *i.e.*, A is not decidable. For more complex problems, we will need to generalize this construction so that the contents of the decider for A_{TM} is arbitrary (given D), but for now, this simple form of a reduction is sufficient for our purposes.

Now, our proof consists of constructing f with the two properties listed above. To do so, I find it best to operate in three steps:

1. As a starting point, build a TM P that it obeys the right-hand side of the biconditional and a TM Q that does not obey this property. We will base M' off of the behavior of these *target* TMs.
2. Next, construct M', the output of f, that conditionally acts like P if the left-hand side of the biconditional is true and Q if the left-hand side of the biconditional is false.
3. Verify that this M' fully obeys the two conditions of mapping reducibility.

For simple reducibility proofs, this construction strategy allows us to quickly build an appropriate mapping function f. More complicated proofs will require additional manipulation, but this strategy at least gives us a starting point in our investigation.

For our example, we must first build a simple machine P such that $L(P) = \Sigma^*$. This is straightforward to do:

P = "On input x:
1. **Accept**."

P accepts any string x given to it, therefore, its language is Σ^*.

Let's also build a simple machine Q such that $L(Q) \neq \Sigma^*$. We have more options here, *i.e.*, Q could accept some finite number of strings. However, it is simpler to have Q simply accept nothing:

Q = "On input x:
1. **Reject**."

Q rejects any string x given to it, therefore, its language is $\emptyset$.

Next, let's build M' that is the output of our mapping function $f(\langle M, w\rangle)$. Based on our biconditional, whenever M accepts w, we should emulate P's behavior; otherwise, we emulate Q's behavior. Define our mapping function as $f(\langle M, w\rangle) = \langle M'\rangle$ where:

M' = "Ignore the input.
1. Run M on w.
2. If M accepts, **accept**.
3. If M rejects, **reject**."

Finally, let's check that M' has the appropriate behavior:

- If M accepts w, then the conditional of M' is always true, therefore the machine always accepts. Thus $L(M') = \Sigma^*$.
- If M rejects w then the conditional of M' is always false, therefore the machine always rejects. Thus $L(M') = \emptyset \neq \Sigma^*$.
- If M loops on w then M' never proceeds past line 1 and thus never accepts any string. Thus $L(M') = \emptyset \neq \Sigma^*$.