L 12. 2022-10-07

Reductions.

Can we prove other languages not decidable? We'll use reductions.

Idea: use a TM for A as a subroutine to solve B.

If we know how to solve problem A
Then we know " " B.

In algorithms design we often use this "forward" direction, where A is a library call and B is a problem we want to solve.

e.g. if we can sort an array
then we can find its unique elements.

But to prove impossibility results, we use the "reverse" direction, or contrapositive.

If it is impossible to solve problem B
Then " " " A.

e.g. if we can decide HALT, we can decide UNI. But we cannot decide UNI. Therefore we cannot decide HALT either.

Let us elaborate on the example.

def $HALT = \{ \langle M, w \rangle : M \text{ halts on input } w \}$.

Assume H is a TM that decides HALT. Let U be a TM that recognizes UNI. Let U' be the following TM:

Input: $\langle M, w \rangle$.

1. Run H on input $\langle U, \langle M, w \rangle \rangle$
2. If H rejects: reject
3. If H accepts: run U on input $\langle M, w \rangle$
4. Answer like U.

Obs this works: if $\langle M, w \rangle \in UNI$ then U accepts, hence $\langle U, \langle M, w \rangle \rangle \in HALT$, and we go to step 3. O/w U may reject, in which case we go to step 3, or does not stop, in which case we go to step 2 and reject.

We have shown that: ※
If we can decide HALT, we can decide UNI. But we cannot decide UNI. Therefore we cannot decide HALT either.

Other undecidable problems:
→ Is the language accepted by M empty? regular?
→ Are the languages accepted by two TMs equal?

Let us formalize the idea of reductions. There are different types, depending on what we are trying to prove with them. For now we look at many-to-one aka mapping reductions.

So far we worked with TMs that recognize a language (and said $L(M) = \{ s : M \text{ accepts } s \}$. That is, they only answer acc/rej.

We are also interested in TMs that transform their input, so that we can chain them together like functions.

def M computes a function $f : \Sigma^* \to \Sigma^*$ if for every input s, M accepts and the tape contains (exactly) f(s).
We say that such a function is computable.

def A mapping reduction from A to B is a computable function f st $s \in A$ iff $f(s) \in B$.

def A is (mapping-) reducible to B ($A \leq_m B$) if there is a mapping reduction from A to B.