#1 Formalizing Reducibility

We wish to formalize this notion of "reduction" that we have been throwing around. The intuition is that we reduce problems to another. We can often write TMs using other TMs; further, on Tuesday we showed problems were undecidable by reducing them from undecidable problems.

Definition 1 (Computable Functions)**.** *A function $f : \Sigma^* \to \Sigma^*$ is a computable function if there exists Turing machine M that, on every input w, halts with just $f(w)$ on its output tape.*

Recall a computable function uses a computing TM—one that just halts (does not bother to accept or reject).

A function is computable if it's a sort of math or string-to-string function that will decidably give us an output on all inputs. We use it to define mapping reducibility. It will also be very important later in the course when we discuss Karp-reducibility.

Definition 2 (Mapping Reducibility)**.** *The language A is mapping reduible to B, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \to \Sigma^*$ where, for every $w \in \Sigma$,*

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

The function f is called the mapping reduction. N.b.An equivalent way of stating the above condition is instead that

1. *if $w \in A$ then $f(w) \in B$, and*
2. *if $w \notin A$ then $f(w) \notin B$.*

See picture on Sipser pp. 235. I will try to remember to draw it in class.

#2 Example

The next example isn't considerably useful, but I give it to drive home the more mathy-side of this definition.

Consider the languages

$$E = \{w \in \{0,1\}^* \mid w \text{ is an even binary number}\}$$
$$O = \{w \in \{0,1\}^* \mid w \text{ is odd binary number}\}$$

Then $f : \Sigma^* \to \Sigma^*$ can be constructed where M_f

1. Checks if the input is a well-formed binary number.
2. If so, output the input incremented by 1.
3. if not, return the input string unchanged.

Now, for correctness:

1. if $w \in E$ then $f(w) \in O$,
2. if $w \notin E$ then either (i) w is odd, or (ii) w is malformed. If (i), then $f(w) \notin O$, as desired. If (ii) is malformed, then $f(w) = w$ and $w \notin O$ either.

A more interesting example of reducibility is between two problems. For example, that INDSET $\leq_m$ VERTEXCOVER.

$$\text{INDSET} = \{\langle G, k\rangle \mid \exists S \subseteq V(G) \text{ s.t. } |S| \geq k \text{ and } \forall u, v \in S, (u, v) \notin E\}$$
$$\text{VERTEXCOVER} = \{\langle G, k\rangle \mid \exists S \subseteq V(G) \text{ s.t. } |S| \leq k \text{ and } \forall (i, j) \in E, i \in S \text{ or } j \in S\}$$

#3 Properties of Reductions

There are two important uses we have for reductions: showing things are decidable and showing things are undecidable. Alas:

Theorem 1. *If $A \leq_m B$ and B is decidable, then A is decidable.*

And, contrapositively,

Theorem 2. *If $A \leq_m B$ and A is undecidable, then B is undecidable.*

#4 Example undecidability reduction

Let's show that $A_{\text{TM}} \leq_m \mathsf{HALT}_{\text{TM}}$. Let f be the mapping reduction and let M_f implement f. Define M_f as follows.

1. M_f takes input w and checks if it is well-formed as $\langle M, w \rangle$. If not, just return *some string* not in $\mathsf{HALT}_{\text{TM}}$ (doesn't matter which).
2. If so, construct M' as follows.
   (a) M' takes input x.
   (b) Run M on x.
   (c) if M accepts, accept. If M rejects, enter a loop.
3. Output $\langle M', w \rangle$.

For correctness,

1. if $\langle M, w \rangle \in A_{\text{TM}}$ then M accepts w and so too does $\langle M', w \rangle$, hence $f(\langle M, w \rangle) \in \mathsf{HALT}_{\text{TM}}$, and
2. if $\langle M, w \rangle \notin A_{\text{TM}}$ then M rejects w and $\langle M, w \rangle$ runs forever, hence $f(\langle M, w \rangle) \notin \mathsf{HALT}_{\text{TM}}$.

#5 The Post-Correspondence Problem

Finally, we will briefly cover one last classic undecidable problem.

Consider a collection of dominoes, each containing two strings. An individual domino looks like:

$$\left[\frac{a}{ab}\right]$$

and a collection of dominoes looks like:

$$\left\{\left[\frac{b}{ca}\right], \left[\frac{a}{ab}\right], \left[\frac{ca}{a}\right], \left[\frac{abc}{c}\right]\right\}$$

The Post Correspondence Problem asks if we can make a list of these dominos (with repetitions allowed) such that the string on top equals the string on the bottom. For example, a solution to the above is

$$\left[\frac{a}{ab}\right]\left[\frac{b}{ca}\right]\left[\frac{ca}{a}\right]\left[\frac{a}{ab}\right]\left[\frac{abc}{c}\right]$$

The PCP is undecidable, and hence can also be used for reductions.