Def: Two languages $A$ and $B$ are mapping equal, written $A \equiv_m B$, if $A \leq_m B$ and $B \leq_m A$.

The reduction $f$ is correct:

1. $f(a) = M$ accepts $w$
2. $M(w)$ goes to $\text{acc}$
3. $M'(w)$ goes to $\text{acc}$
4. $M'(w)$ goes to $\text{acc}$

Thus $A_{TM} \equiv_m H_{TM}$. We need to map

$\langle M, w \rangle \overset{c}{\rightarrow} \langle M', w \rangle$

$s.t. \langle M, w \rangle \in H_{TM} \iff \langle M', w \rangle \in A_{TM}$

ie. $M(w) \uparrow \iff M'(w) \text{ accepts } w$.

Thus $A_{TM} \equiv_m H_{TM}$.

Historically, both have been called "the halting problem."
We showed $K_{TM} \leq_m A_{TM}$ via $F(K_{TM}) = \langle M, M \rangle$.
Also note that $M_1$ accepts its own code if and only if $M$ accepts $W$.

The All-Or-One "Nothing Switch" Accept $X$. We showed $A_{TM} \leq_m \text{NETM}$ and $A_{TM} \leq_m \text{ALL}_{TM}$.

A similar pattern causes many programming problems to be undecidable.

A Third Reduction "Delay Flip-Switch" Design Pattern. $\langle M \rangle \not\equiv_{m} M'$.

This code construction is computable. Analysis:

$\langle M \rangle \in K_{TM} \Rightarrow M$ accepts $\langle M \rangle \Rightarrow M$ accepts $\langle M \rangle$ within some number of steps.

For all $x$, $|x| \geq t$, $M_1$ sees the acceptance and hence rejects $X$.

$\forall x, M_1 \text{ accepts } \langle M \rangle$.

$L(M')$ is finite, so in particular $L(M') = \emptyset$. Whereas, $\langle M \rangle \in K_{TM}$, i.e. $L(M') = \emptyset$. $M$ never accepts $\langle M \rangle$. For all $x$, however long, $M$ never sees acceptance = $\forall x, M_1 \text{ accepts } x = L(M') = \emptyset$. 

\[ \langle M, w \rangle \not\equiv_{m} M' \iff \text{ acceptance of } M \text{ on } w \]
Theorem: For all $A \subseteq \text{RE}$, $A \leq_m \text{ATM}$, and so by transitivity $A \leq_m \text{K}_{\text{TM}}$.

Proof: By $A \subseteq \text{RE}$, we can take a TM $M$ with $M$ is fixed, so this just appends $x$ to $\text{L}(M) = A$. Then map any $x \in \Sigma^*$ to $f(x) = \langle M, x \rangle$.

Then $x \in A \iff M$ accepts $x \iff \langle M, x \rangle \in \text{ATM}$. So $A$ reduces $A$ to $\text{ATM}$.

Definition: A language $B$ is complete for a class $\mathcal{C}$ if $B \in \mathcal{C}$ and for all $A \in \mathcal{C}$, $A \leq_m B$. \quad \therefore \text{ATM and K}_{\text{TM}}$ are $\text{RE}$-complete.

ALL-$\text{TM}$ is $\text{RE}$-hard since every $A \subseteq \text{RE}$ reduces to it, but not complete because ALL-$\text{TM} \not\subseteq \text{RE}$.

---

PREVIEW of next week: 1. If we write every second ID in a computation backwards, the language of valid halting computations—by a given TM $M$ on some input $x$—becomes an intersection $L(D_1) \cap L(D_2)$ of two DCFLs. The OIA checks $L_m I_{t+1}$ for even $t$, while $(L_1, L_2)$ is mostly like checks for marked palindromes except for the one at two places where $M$ makes changes according to its $S$. And check $L_2$'s $L_m I_{t+1}$ for odd $t$.

Then $M \in F_{\text{TM}} \iff L(M) = \emptyset \iff M$ has no valid accepting computations $\iff L_1 \cap L_2 = \emptyset \iff (L_1 \cup L_2) = L'(1) \cup L'(2) = \Sigma^*$.

Now $L'(1) \cup L'(2)$ is the union of two DCFLs, hence it is a CFL. By chaining theorems in the text, we can build a CFG $G$ for it. And "we can build" means there is a computable function $f$ such that $f(<M>) = <G>$. Thus $E_{\text{TM}} \leq_m \text{ALL}_{\text{RE}}$.