CSE491/596 Lecture Mon 10/10 Fall 2022

<u>Prelim I</u> is in next lecture period: Does <u>not</u> cover the current topic of Reductions.

<u>First</u>, an annoying, necessary, but ultimately ignorable point:

$D_{TM} = \{\langle M\rangle : M \text{ does not accept } \langle M\rangle\}$

$= \{x : x$ decodes as a valid Turing machine M, such that M does not accept $x\}$.

$K_{TM} = \{\langle M\rangle : M \text{ does accept } M\}$

$= \{x : x$ decodes as a valid M such that M <u>does</u> accept $x\}$.

D_{TM} is not literally the complement of K_{TM}, because neither set includes strings x that are not valid codes of TMs. Remedies:

- Throw all the invalid codes x into one set or the other. Then they become literally complementary after all.
- Regard invalid codes as giving the DTM $M_0 =$ 

which has $L(M_0) = \emptyset$. Like throwing invalid codes into D_{TM}.
- Enumerate TMs as $M_0, M_1, M_2, M_3, \dots$ so that the binary number (or string) i is taken as the code for M_i. Using such Gödel Numbers, re-define $D_{TM} = \{i : i \notin L(M_i)\}$, $K_{TM} = \{i : i \in L(M_i)\}$. Then <u>complementary</u>

The Best Way, IMHO:

- Regard "Just a Machine $\langle M \rangle$" as the <u>Instance Type</u>. Complementation means within that type.

Other Instance Types: "A Machine and a String" $\langle M, x \rangle$
"Two Machines" $\langle M_1, M_2 \rangle$.

✲ Reduction functions can count as $f : \underline{\underline{\Sigma^*}} \to \Sigma^*$ ✲ even if we only expressly define them on an instance type. This also enables a pictorial way of presenting reductions.

<u>Example</u>: $A_{TM} \leq_m HP_{TM} = \{\langle M', x \rangle : M'(x)\downarrow\}$ "halts"

Construction: where $x' = x$

$\langle M, x \rangle \xrightarrow{f} \langle M', x' \rangle$ where M' =

Computable — f is a code translation hence obviously computable.

Correctness: We need that for all M and x:

$\langle M, x \rangle \in A_{TM} \iff f(\langle M, x \rangle) = \langle M', x \rangle \in HP_{TM}$

ie. M accepts x if and only if $M'(x)\downarrow$.

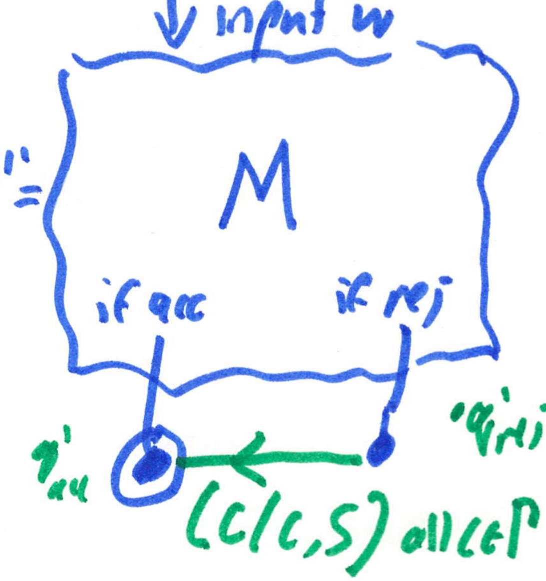
M accepts $x \Rightarrow M'(x)\downarrow$ in q'_{acc}, and:

M does not accept $x \Rightarrow M(x)\uparrow$, so $M'(x)\uparrow$ or $M(x)$ goes to q_{rej}, when $M'(x)\uparrow$ in the loop, so either way $M'(x)\uparrow$. ∎

¿ Does the same construction also show $HP_{TM} \le_m A_{TM}$?

No, but this one does:

$\langle M, w\rangle \hookrightarrow \langle M'', w\rangle$ where $M'' =$



Correctness Logic: need

$M(w)\downarrow \iff M''$ accepts w

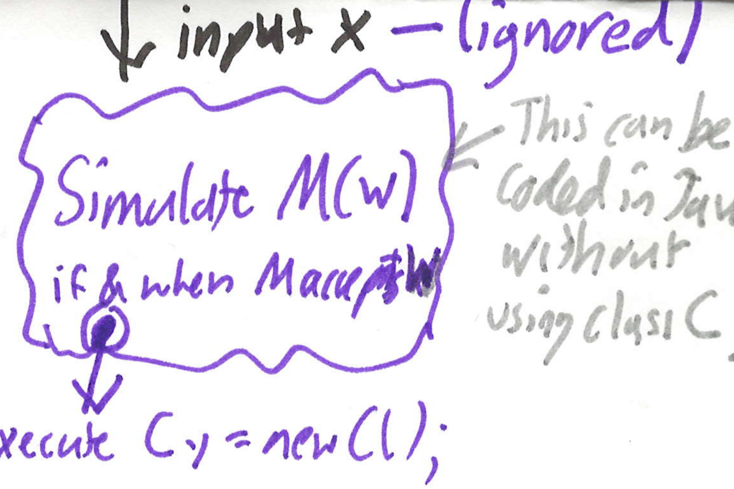
Thus we have shown $A_{TM} \le_m HP_{TM}$ and $HP_{TM} \le_m A_{TM}$.

Defn: Two languages A and B are <u>mapping-equiv't</u>, written $A \equiv_m B$, if $A \le_m B$ and $B \le_m A$.

Example from Fri 10/14

$\langle P, C\rangle$

$\langle M, w\rangle \xrightarrow{f} P$:



instance of A_{TM} problem

Logical Goal:

$\langle M, w\rangle \in A_{TM} \iff (P, C) \in$ UsefulCode

$\quad$ ||| $\qquad\qquad\qquad\qquad$ |||

M accepts w $\qquad$ there is an x s.t. $P(x)$ creates a new C.