Top Hat 4703

Three Reduction "Toolkit" Ideas.

① "Waiting for Godot" style Reductions

Consider the following kind of "Code Cleanup" problem

CC:

INST: An OOP program P and a class C in P.

QUES: Is there any possible input x so that $P(x)$ eventually creates an object of class C?

[Or, can we just delete class C by now?]

[The language CC __is__ c.e.]

__Theorem__: $A_{TM} \leq_m CC$, so the CC problem is undec.

Suppose we had a decider R for [all cases of] the CC problem.

Then we would get a decider Q for the A_{TM} problem as follows:

__Construction:__

1. Input $\langle M, w\rangle$
2. Build P as shown.
3. Run the decider R on $\langle P\rangle$.
4. If R accepts, accept; else reject.

But Q cannot exist since A_{TM} is undecidable. So R cannot exist.

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

input x

Class C is not used here

Simulate $M(w)$ if and when it accepts

eg using Turing Kit

C y = new C(---);

f is __computable__ because the Turing Kit is a fixed routine.

The reduction is correct because for all M and w:

$\langle M, w\rangle \in A_{TM} \implies M$ accepts $w \implies$ on ~~any~~ some x, $P(x)$ creates a C obj.

$\langle M, w\rangle \notin A_{TM} \implies$ for __all__ x, $P(x)$ does not create a C obj.

Because A_{TM} is undecidable, it follows that CC is undec.

② "All or Nothing Switch": Revisit $A_{TM} \leq NE_{TM}$: ②

$\langle M, w \rangle \xrightarrow{f} M'$: $\downarrow$ input x (ignored) $\boxed{\text{Sim } M(w) \text{ if \& when acc}} \rightarrow$ accept x.

$\langle M, w \rangle \in A_{TM} \equiv M$ acc $w \implies (\forall x)\, M'$ accepts $x \implies L(M') \neq \emptyset$

Also $\implies L(M') = \Sigma^*$.

$\langle M, w \rangle \notin A_{TM} \implies L(M') = \emptyset \implies \langle M' \rangle \notin NE_{TM}$

$\implies \langle M' \rangle \in ALL_{TM}$

So we also have $\implies \langle M' \rangle \notin ALL_{TM}$

$A_{TM} \leq_m ALL_{TM}$ <u>via</u> the <u>same</u> reduction fn. f.

Also: $\langle M, w \rangle \in A_{TM} \implies L(M') = \Sigma^* \implies M'$ accepts $\langle M' \rangle$

$\langle M, w \rangle \notin A_{TM} \implies L(M') = \emptyset \implies M'$ does not accept $\langle M' \rangle$

$\therefore A_{TM} \leq_m K_{TM}$. Since we already had $K_{TM} \leq_m A_{TM}$

we write $A_{TM} \equiv_m K_{TM}$ and say they are <u>mapping-equivalent</u>.

Is ALL_{TM} equivalent? Is it c.e.? Let's go to Technique ③.

<u>FYI</u>: The All or Nothing Switch can be composed with other routines.

$\langle M, w \rangle \xrightarrow{f} M'$: $\downarrow$ input x $\boxed{\text{Sim } M(w), \text{ if \& when acc}} \rightarrow \boxed{\text{Accept } x \text{ iff } x \text{ is a palindrome}}$

$\langle M, w \rangle \in A_{TM} \implies L(M') = PAL$

$\langle M, w \rangle \notin A_{TM} \implies L(M') = \emptyset$.

$\therefore A_{TM} \leq_m$ "Given a TM M', is $L(M')$ <u>non</u> regular?" $\therefore REG_{TM}$ in text HW is undecidable.

③ "Delay Switch": Show $D_{TM} \le_m ALL_{TM}$ ③

which is the same as showing $K_{TM} \le_m \sim ALL_{TM}$.

$w = \langle M \rangle$

$\downarrow$ input x, $n = |x|$.

$\langle M, \cancel{w} \rangle \xrightarrow{f} M'$ = $\boxed{\text{Simulate } M(\cancel{w}\ \langle M \rangle) \text{ for up to } n \text{ steps. Did it accept?}}$

yes → Reject x.

no → Accept x

This code transformation is likewise computable.

For correctness:

$\langle M \rangle \in D_{TM} \Rightarrow M$ does not accept $w = \langle M \rangle$

" $\langle M \rangle \notin K$ $\Rightarrow$ for all n, M does not accept $\langle M \rangle$ within n steps!

$\Rightarrow$ for all x, $M'(x)$ does not take the yes branch

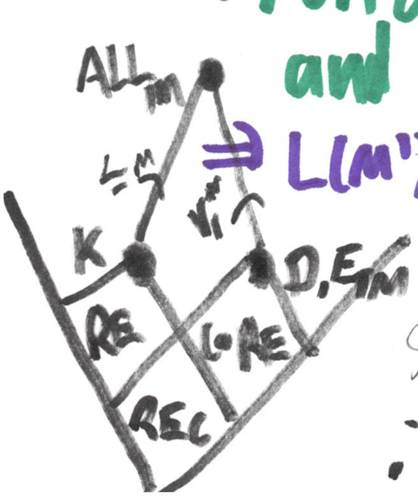
$\Rightarrow L(M') = \Sigma^* \Rightarrow \langle M' \rangle = f(\langle M \rangle) \in ALL_{TM}$

$\langle M \rangle \notin D_{TM} \Rightarrow M$ does accept $w = \langle M \rangle$

" $\Rightarrow$ there is an n s.t. M accepts $\langle M \rangle$ within n steps

$n \in K_{TM}$ $\Rightarrow$ on any x, $|x| \ge n$, $M'(x)$ detects this acceptance and so takes the yes branch and rejects x.

$\Rightarrow L(M') \ne \Sigma^*$, indeed $L(M')$ is finite, $\Rightarrow f(M') \notin ALL_{TM}$



Since $K \le_m ALL_{TM}$ and K is not co-ce, ALL_{TM} is not co-ce.

Since $D \le_m ALL_{TM}$ and D is not ce, ALL_{TM} is not ce.

$\therefore$ ALL_{TM} is neither c.e. nor co-c.e. ☒

<u>Def^n</u>: A language B is <u>hard</u> <u>for</u> a class $\mathcal{C}$ of languages if for all $A \in \mathcal{C}$, $A \leq_m B$. (under $\leq_m$)
If also $B \in \mathcal{C}$, then B is <u>complete</u> for $\mathcal{C}$.

<u>Theorem</u>: A_{TM} is complete for RE <u>under</u> $\leq_m$.

Proof: $A_{TM} \in RE$. ✓ Let any $A \in RE$ be given.
Goal: show $A \leq_m A_{TM}$.

By $A \in RE$, we can by defn <u>take</u> a TM M_a s.t. $L(M_a) = A$.
Given any $x \in \Sigma^*$, map

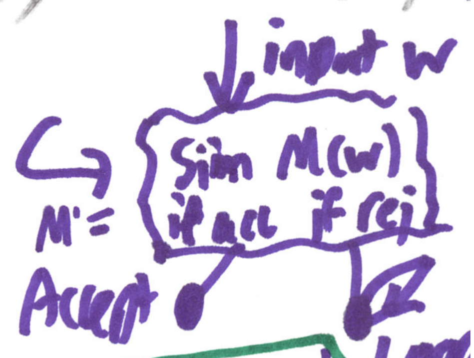
$$f(x) = \langle M_a, x \rangle. \quad \text{Then}$$

$x \in A \equiv M_a$ accepts $x \iff \langle M_a, x \rangle = f(x) \in A_{TM}$.
So $A \leq_m A_{TM}$ via f, and since $A \in RE$ is arbitrary, A_{TM} is <u>RE-complete</u>.

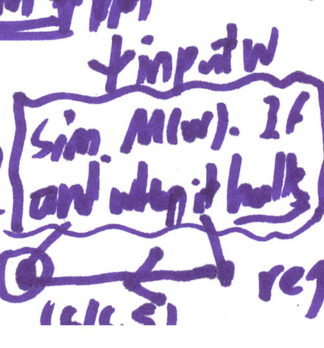
<u>Fact</u>: If B is complete and $C \equiv_m B$, then C is complete (because $\leq_m$ is transitive).

HALTING PROBLEM
Inst $\langle M, w \rangle$
Ques Does $M(w)\downarrow$?

$A_{TM} \leq HP_{TM}$: ~~$HP \leq_m A_{TM}$~~ : $\langle M, w \rangle \hookrightarrow M' =$



$HP_{TM} \leq_m A_{TM}$: $\langle M, w \rangle \xrightarrow{f} M'' =$



M acc $w \iff M'(w)\downarrow$
$M(w)\downarrow \iff M''$ acc w.