CSE 396 — Lecture Thu May 2 — Spring 2019

Top Hat 4456

Def^n: Two languages A and B are {many-one / mapping} equivalent, written $A \equiv_m B$, if $A \leq_m B$ and $B \leq_m A$.

AP (Acceptance Problem) ie A_{TM}: INST: $\langle M, w \rangle$ QUES: Is $w \in L(M)$?

HP ie. HP_{TM} (Halting Problem) INST: $\langle M, w \rangle$ QUES: Does $M(w)\downarrow$ ("halt" $\downarrow$)

$A_{TM} \leq_m HP_{TM}$: Map

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

We may assume M is a DTM in the text's (q_{acc}, q_{rej}) form.

We can compute the code of M' given M since it just adds loop arcs

The reduction f is correct:

$\langle M, w \rangle \in A_{TM} \Rightarrow M$ accepts w

$\Rightarrow M(w)$ goes to q_{acc}

$\Rightarrow M'(w)$ goes to $q_{acc} \Rightarrow M'(w)\downarrow$

$\Rightarrow f(\langle M, w \rangle) = \langle M', w \rangle \in HP_{TM}$

$\langle M, w \rangle \notin A_{TM} \Rightarrow M'(w)$ goes to the old q_{rej} which is now a looping state $\boxed{\text{OR}}$ $M'(w)$ never escapes the body of M.

Either way, $M'(w)\uparrow$, so $\langle M', w \rangle \notin HP_{TM}$

Thus $\langle M, w \rangle \in A_{TM} \Leftrightarrow \langle M', w \rangle \in HP_{TM}$, so $A_{TM} \leq_m HP_{TM}$.

ie. M accepts $w \Leftrightarrow M'$ halts on w.

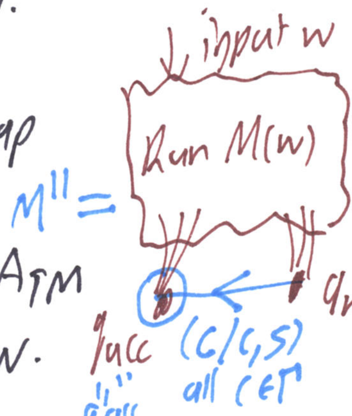
$HP_{TM} \leq_m AP_{TM}$. We need to map

$\langle M, w \rangle \xrightarrow{f} \langle M'', w \rangle$

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

ie. $M(w)\downarrow \Longleftrightarrow M''$ accepts w.

$M'' =$



$\langle M, w \rangle \in HP_{TM} \equiv M(w)\downarrow$

$\Rightarrow$ the computation $M(w)$ comes out at q_{acc} or at q_{rej}

$\Rightarrow M''(w)$ goes to q_{acc}

$\Rightarrow M''$ accepts w. whereas $\langle M, w \rangle \notin HP_{TM} \Rightarrow M''(w)$ still fails to halt within the body of $M \Rightarrow M''(w)\uparrow$

$\Rightarrow M''$ does not accept w.

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

Historically, both have been called "the halting problem".

Similarly, $ALL_{TM} \equiv_m \underline{TOT} = \{\langle M \rangle :$ for all x, $M(x)\downarrow\}$ ie. M is total.

We showed $K_{TM} \leq_m A_{TM}$ via $f(\langle M \rangle) = \langle M, M \rangle$.

We showed $A_{TM} \leq_m NE_{TM}$ and $A_{TM} \leq_m ALL_{TM}$ via ① "The All-or-Nothing Switch"

$\langle M, w \rangle \xrightarrow{f} M' =$ ↓ input x → {Run $M(w)$ — if & when acc.} → Accept x

M accepts w ⇓ $\forall x$ M' acc. x ⇓ $L(M') = \Sigma^* \neq \emptyset$

M does not acc. w ⇒ $L(M') = \emptyset$

Also note that M' accepts its own code if and only if M accepts w.

$\therefore \langle M, w \rangle \in A_{TM} \iff \langle M' \rangle \in K_{TM}$. So $A_{TM} \leq_m K_{TM}$. $\therefore A_{TM} \equiv_m K_{TM}$

Historically, A_{TM} and K_{TM} are "slurred together" as well.

A similar pattern causes many programming problems to be undecidable.

"USEFUL CODE^2" UC_{Java}

INST: A Java program P and a particular line ℓ in P.

QUES: Is there an input x for which line ℓ gets executed?

Show $A_{TM} \leq_m UC_{Java}$.

$\langle M, w \rangle \xrightarrow{f} P_{M,w}$: ↓ input x → {Run $M(w)$ (using TK it)} → if & when accepts → ℓ: Brew Coffee.

$\langle M, w \rangle \in A_{TM} \Rightarrow$ for all x, $P_{M,w}(x)$ reaches line ℓ. But

$\langle M, w \rangle \notin A_{TM} \Rightarrow$ for all x, $P_{M,w}(x)$ never reaches line ℓ, so it never gets executed..

② "Waiting for..." Reduction

A Third Reduction ③ "Delay Flip-Switch" Design Pattern.

$\langle M \rangle \xrightarrow{f} M' =$

↓ input x

[Calculate $n = |x|$]

[Run M on $\langle M \rangle$ for n steps (up to)]

Did it accept? — yes → Reject x; no → Accept x.

From K_{TM} not A_{TM}.

This code construction is computable. Analysis:

$\therefore D_{TM} \leq_m ALL_{TM}$

$\therefore ALL$ is not c.e. and not co-c.e.

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

$\Rightarrow$ for all x s.t. $|x| \geq t$, M' sees the acceptance and hence rejects x.

$\Rightarrow L(M')$ is finite, so in particular $\Rightarrow L(M') \neq \Sigma^*$. Whereas,

$\langle M \rangle \notin K_{TM}$, i.e. $\langle M \rangle \in D_{TM} \Rightarrow M$ never accepts $\langle M \rangle \Rightarrow$ for all x, however long, $M'(x)$ never sees acceptance $\Rightarrow \forall x$ M' accepts x $\therefore L(M') =$

<u>Theorem</u>: For all $A \in RE$, $A \leq_m A_{TM}$, and so by transitivity $A \leq_m K_{TM}$.

<u>Proof</u>: By $A \in RE$, we can take a TM M_a such that $L(M_a) = A$. Then map any $x \in \Sigma^*$ to $f(x) = \langle M_a, x \rangle$. [$M_a$ is fixed, so this just appends x to]

Then $x \in A \iff M_a$ accepts $x \iff \langle M_a, x \rangle \in A_{TM}$. So f reduces A to A_{TM}. ☒

<u>Defn</u>: A language B is <u>complete</u> for a class $\mathcal{C}$ if • $B \in \mathcal{C}$ and • for all $A \in \mathcal{C}$, $A \leq_m B$. $\therefore$ A_{TM} and K_{TM} are RE-complete

ALL_{TM} is <u>RE-hard</u> since every $A \in RE$ reduces to it, but not complete because $ALL_{TM} \notin RE$.

<u>PREVIEW of next week</u>: ① If we write every second ID in a computation backwar[ds]

$$\vec{C} = [I_0(x)][I_1^R][I_2][I_3^R][I_4][I_5^R][I_6]\ldots$$

$\vdash \quad \vdash \quad \vdash \quad \vdash \quad \vdash \quad \vdash$

then the language of <u>valid</u> {accepting / 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 DP[DA]

D_1 checks $I_t \vdash_M I_{t+1}$ for even t, while D_2 checks $I_t \vdash_M I_{t+1}$ for odd t.

(given I_t, I_{t+1}^R) Checking $I_t \vdash_M I_{t+1}$ is mostly like checking marked palindromes except for the one or two places where M made changes according to its δ. And check last ID has q_{acc}

Then $M \in E_{TM} \equiv L(M) = \emptyset \iff M$ has no valid accepting comp^{n}s

$\iff L(D_1) \cap L(D_2) = \emptyset$

$\iff \sim(L(D_1) \cap L(D_2)) = \widetilde{L(D_1)} \cup \widetilde{L(D_2)} = \Sigma^*$.

Now $\widetilde{L(D_1)} \cup \widetilde{L(D_2)}$ 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 <u>computable</u> function f such that $f(\langle M \rangle) = \langle G \rangle$. Thus $\boxed{E_{TM} \leq_m ALL_{CFG}}$.

② P and NP are like DEC and RE under <u>polynomial-time</u> reductions $\leq_m^p$. But we don't know if NP ≠