CSE 396 Lecture Thu 5/5 Fall 2017

Def^n: A language A many-one (later (polynomial time)) mapping-reduces to a language B if there is a (polynomial time) <u>computable</u> fn

$f: \Sigma^* \to \Sigma^*$ such that for all x

dom(f) $= \Sigma^*$

$$x \in A \iff f(x) \in B.$$

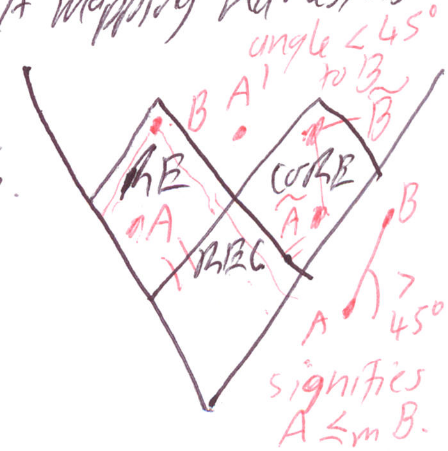
If so, we write $A \leq_m B$, spoken "A mapping reduces to B"

Theorem: Suppose $A \leq_m B$. Then:

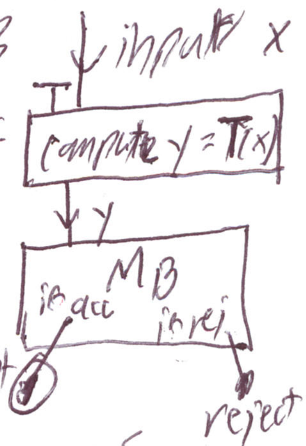
(a) If B is decidable, then A is decidable.

(b) If B is r.e. / c.e. / $\in RE$ then A is r.e. / c.e. / $\in RE$

(c) If $B \in$ co-RE then $A \in$ co-RE



Proof: (a) We can take a <u>total</u> TM M_B s.t. $L(M_B) = B$ and a total TM T that computes $f(x)$. (M_A:



We will build a TM M_A s.t. $L(M_A) = A$ and A is total.

M_A is a composition of two total routines so it is total.

M_A accepts $x \iff M_B$ accepts y (which $= f(x)$)

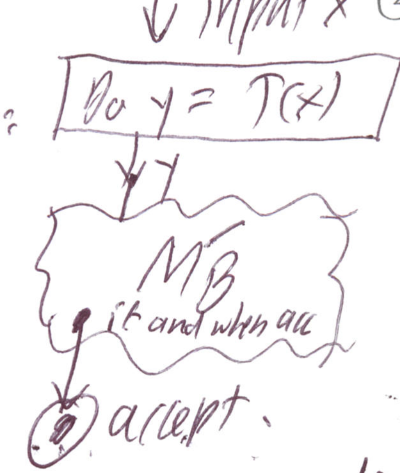
$\iff y \in B \iff x \in A$ by "$A \leq_m B$ via f."

$\therefore L(M_A) = A$

For (b), re-do diagram with M_B not a solid box.

Take a TM M'_B s.t. $L(M'_B) = B$ but since we only know $B \in RE$, we diagram M'_B as a "wavy box":

M'_A:



For all x, M'_A accepts $x \overset{\text{by code}}{\Leftrightarrow} M'_B$ accepts y

$\Leftrightarrow M_B$ accepts $f(x) \Leftrightarrow f(x) \in B \Leftrightarrow x \in A$ $\therefore L(M'_A) = A$

$y = f(x) = T(x)$ by $L(M'_B) = B$ by reduction assumption

(c) Note: $\forall x: x \in A \Leftrightarrow f(x) \in B$ $\therefore A \leq_m B$

$\Leftrightarrow \forall x: x \in \tilde{A} \Leftrightarrow f(x) \in \tilde{B}$ $\Leftrightarrow \tilde{A} \leq_m \tilde{B}$ ☒

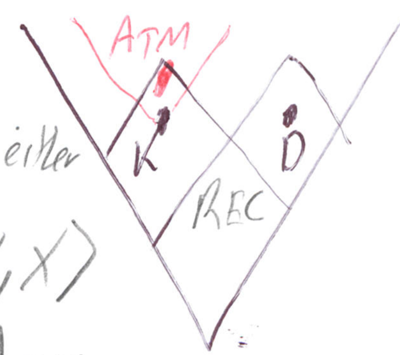
So $B \in coRE \wedge A \leq_m B \Rightarrow \tilde{B} \in RE \wedge \tilde{A} \leq_m \tilde{B} \overset{\text{by (b)}}{\Rightarrow} \tilde{A} \in RE \Rightarrow A \in coRE$.

<u>Contrapositive</u>: Suppose $A \leq_m B$ <u>Then</u>:

(a') If A is undecidable, then B is undecidable.

(b') If A is not Turing recognizable, then neither is B.

(c') If $A \notin coRE$ then $B \notin co\text{-}RE$.



Example reduction K is undecidable, not co-re either

"A" = K_{TM} "B" = A_{TM} $K \leq_m A_{TM}$ via $f(x) = \langle x, x \rangle$

$x \in K \Leftrightarrow x$ is a TM that accepts $x \Leftrightarrow \langle x, x \rangle \in A_{TM}$

The function $f(x) = \langle x, x \rangle$ is easily computable

Define <u>NE_{TM}</u> = INST: A TM M
Ques: Is $L(M) \neq \emptyset$?

<u>E_{TM}</u>: Inst: A TM M
Ques: Is $L(M) = \emptyset$?

Theorem: $A_{TM} \le_m NE_{TM}$, so NE_{TM} (and E_{TM}) are undecidable

Domain = $\{\langle M, w\rangle$: where M is a TM and $w \in \Sigma^*\}$ "X"

"A" = A_{TM} "B" = NE_{TM}

Range = TMs. Hence our goal is to build $M' = f(M, w)$ by a computable fn f, such that $\langle M, w\rangle \in A_{TM} \iff \langle M'\rangle \in NE_{TM}$ i.e. st. M accepts $w \iff L(M') \neq \emptyset$

M': ↓ input z
Simulate $M(w)$
if and when M accepts w
→ Accept z.

Then for all instances $\langle M, w\rangle$ of the A_{TM} problem,

$\langle M, w\rangle \in A_{TM} \equiv M$ accepts $w \Rightarrow$ for all z M' accepts $z \Rightarrow L(M') = \Sigma^*$
$\Rightarrow L(M') \neq \emptyset$
$\Rightarrow \langle M'\rangle \in NE_{TM}$

M does not acc. w i.e. $\langle M, w\rangle \notin A_{TM}$ $\Rightarrow$ for all z, M' does not accept z
$\Rightarrow L(M') = \emptyset \Rightarrow \langle M'\rangle \notin NE_{TM}$

And the code $\langle M'\rangle = f(M, w)$ is computable "by Hardcoding",

so $A_{TM} \le_m NE_{TM}$ via f because $\langle M, w\rangle \in A_{TM} \iff \langle M'\rangle \in NE_{TM}$.

⊗ **Added:**

This reduction embodies an important idea: The "All-or-Nothing Switch":

M accepts $w \Rightarrow L(M') = \Sigma^*$. So you have $\langle M, w\rangle \in A_{TM} \iff \langle M'\rangle \in ALL_{TM}$

M does not acc $w \Rightarrow L(M') = \emptyset$. and also $\langle M, w\rangle \in A_{TM} \iff \langle M'\rangle \in NE_{TM}$.

Thus the same reduction function f reduces A_{TM} to ALL_{TM} and to NE_{TM}. It follows that ALL_{TM} is likewise undecidable, indeed, not co-c.e. Whereas NE_{TM} is c.e., ALL_{TM} is not c.e. either. The way to show that is to show $D_{TM} \le_m ALL_{TM}$ too. This uses a different idea I call the "Delay Switch" — to come in Tuesday's lecture.