Lecture Thu 4/30  

Undecidability & Reductions

Two main facts pictured:
1. $\text{RE} \cap \text{co-RE} = \text{REC}$
2. Reductions - $\text{RE}$, $\text{co-RE}$, and $\text{REC}$ are all closed under their

languages

$\text{RE} = \{ L : \text{for some Turing Machine } M, L = L(M) \}$

$\text{co-RE} = \{ L : \sim L \in \text{RE} \} = \{ L : \text{for some TM } M, L = \sim L(M) \}$

$\text{REC} = \{ L : \text{for some TM } M, L = L(M) \text{ and } M \text{ halts for all inputs} \}$

For all languages $L$, $(L$ is recognizable $\& \bar{L}$ is recognizable $) \iff L$ is decidable.

Theorem 4.22: A language is Turing recognizable $\& \text{co-TM recognizable iff it is decidable.}$

Note: The $\iff$ direction is immediate staying at high level if $L$ is decidable it is ipso facto recognizable, ditto for $\bar{L}$.

For $\Rightarrow$ we need $h$ such: If there are TMs $A$ and $B$ such that $L(A) = L$ and $L(B) = \sim L$, then $L$ is decidable, which means we can build a TM $M$ such that $L(M) = L$ and $M$ halts for all inputs.
1. **Proof:** We are given partial algorithms $A$ and $B$:

- **Partial:** when they don't accept an input $x$, they might not halt.

**Input $x$:**

1. Simulate one (more) step of $A(x)$.
2. Did $A$ just now accept?
   - Yes: Simulate one more step of the comp. $B(x)$.
   - No: $x \in \Sigma^*$.
3. Did $B$ just now accept?
   - Yes: $x \in \Sigma^*$.
   - No: $x \in \Sigma^*$.

**Claim:** $L(M) = L$ for all $x \in \Sigma^*$.

**Proof:**

- For all $x \in \Sigma^*$, if $x \in L$, then $A(x)$ will eventually accept, and $B(x)$ will halt and accept.
- Conversely, if $x \notin L$, then $A(x)$ will eventually halt without accepting, and $B(x)$ will reject.

By the Fact, the bit flip algorithm always halts with the correct answer, so $L(M) = L$ and $M$ is robust for all $x \in \Sigma^*$.

2. **Definition:** A language $B$ is mapping reducible to a language $C$, written $B \leq_m C$, if there is a compatible function $f : \Sigma^* \rightarrow \Sigma^*$ such that for all $x$, $x \in B \Rightarrow f(x) \in C$.

**Note:** $f$ is compatible implies there is a TM $T$ with output $f(x)$ for $x \in \Sigma^*$.

**Note 2:** A language $L$ is decidable if the fn $f_L(x) = 1$ in $\Sigma^*$.

**Example:** Define $KTM = \{ e(M) : M \text{ does accept } e(M) \}$. $KTM$ is the language of the

- **Basically:** $KTM = \neg \text{Dom}_M$, so it is undecidable. "Self-Acceptance Problem."
ATM = Inst: A machine \( M \) and a string \( W \)  
Type: Machine \& a string. 

\( \text{TM} = \text{Inst: Just a machine } M. \)  
\( \text{Ques: Does } M \text{ accept the string } e(M)? \)

Example: \( K_m \leq_m \text{ATM} \text{ via the function } f(M) = \langle M, e(M) \rangle \)

Referencing the Def: " \( \emptyset \equiv \text{Km} \) \) \( C \equiv \text{ATM} \) " \( x \equiv M \)  
\( f(x) \) is a pair \( \langle M, e(M) \rangle \)

Then via the Def, we need \( M \in \text{Km} \equiv f(M) \in \text{ATM} \)  
and \( f \) is computable. \( \boxed{\text{Ill}} \)

\( M \) accepts \( e(M) \) since \( f(M) = \langle M, e(M) \rangle \) and \( \langle M, W \rangle \in \text{ATM} \equiv M \) accepts \( W \).

\( \text{Note: } e(M) \text{ to encode and then } f \) doubles up to make a pair.

Text notation w/o \( e(M) \): \( f(<M>) = \langle M, <M> \rangle \).

Notation w/o source/code distinction: \( f(M) = M \# M \).

We need \( e(M) \) to be decidable.

\[ f(u) = \begin{cases}  \text{if } u \text{ is the code of a TM } M, \text{ then output } <M,M> \\  \text{if } u \text{ is not a valid code, i.e., } u \notin \text{Range}(e) \text{ or } <u,e(M)> \in \text{ATM} \text{ (all the same)} \text{ then just output } e \text{ since } e \notin \text{ATM}. \end{cases} \]

We may always assume ranges of encoding algorithm are nice.

\[ \forall u \in \mathbb{Z} \text{, } f(u) = M, M. \]

ME \( \text{Km} \equiv f(M) \in \text{ATM} \).

Since we initially showed \( \text{Dm} \) and hence \( \text{Km} \) are undecidable, it follows from this that \( \text{ATM} \) is undecidable.

Since \( \text{ATM} \) is i.e., this shows so is \( \text{Km} \).
C is a mirror image of L

- If \( B \leq_m C \), then this is indicated by a steeper than 45° angle from 13 to 16.

Theorem: If \( B \leq_m C \), then:

1. If \( C \) is decidable, then \( B \) is decidable.
2. If \( C \) is recognizable, then \( B \) is recognizable (re: or c.e.)
3. If \( C \) is c.o.-c.e., then \( B \) is c.o.-c.e.

Proof: (1) Given \( C \) is decidable, this means we have a total TM \( M_C \).

Given \( B \leq_m C \), we have a total TM \( T \) s.t.: \( \forall x: x \in B \iff T(x) \in C \).

Goal: build a total TM \( M_B \) s.t.

\[ L(M_B) = B. \]

As a simple combo of total boxes, \( M_B \) is total. And \( x \in B \iff x \in C \)

so \( M_B \) accepts \( x \iff x \in C \iff x \in B \)

(2) Since we copy back the same answer,

(3) follows because \( x \in B \iff T(x) \in C \iff x \in B \),

Halting: Inst.: A TM \( M \) and an input \( w \).

Ques. Does \( M(w) \) halt?

Show Halting is undecidable.

Two styles. 65.1 "vs" 65.3

§5.1: Suppose we had a decider \( D \) for the Halting problem. Then we could build a decider \( S \) for APM as follows (---) But \( S \) cannot exist while APM is undecidable. So Halting is indec.
Text Algorithm:

Input \( \langle M, w \rangle \) instance of \( \text{ATM} \).

1. Use the hypothetical decider \( R(M, w) \).
   - If it says no, \( M(w) \) doesn't halt, then \( M \) can't possibly accept, so \( \langle M, w \rangle \notin \text{ATM} \).
   - If it says yes, then it is safe to run \( M(w) \) to see if it accepts, and copy back the answer.

Given \( R \) decider

\( \text{HALT} \) is undecidable.

Show: \( \text{ATM} \leq_m \text{HALT} \).

\( M \) is Turing machine

\[ T(M, w) = (M', w) \text{ st.} \]

\[ \langle M, w \rangle \in \text{ATM} \Leftrightarrow \langle M', w \rangle \in \text{HALT} \]

\[ M \text{ accepts } w \Leftrightarrow M'(w) \text{ halts.} \]

Since rejecting by \( M \) causes an infinite loop in \( M' \),
\[ M(w) \text{ accepts } \Leftrightarrow M'(w) \text{ halts.} \]

And \( T \) just adds an arc, so compatible.

Extra: Conversely, we can show \( \text{HALT} \leq_m \text{ATM} \) as follows.

We need a mapping \( T \) that given \( M \) and \( w \) makes \( T(M, w) = (M', w) \) such that

\[ M \text{ halts on } w \Leftrightarrow M' \text{ accepts } w. \]