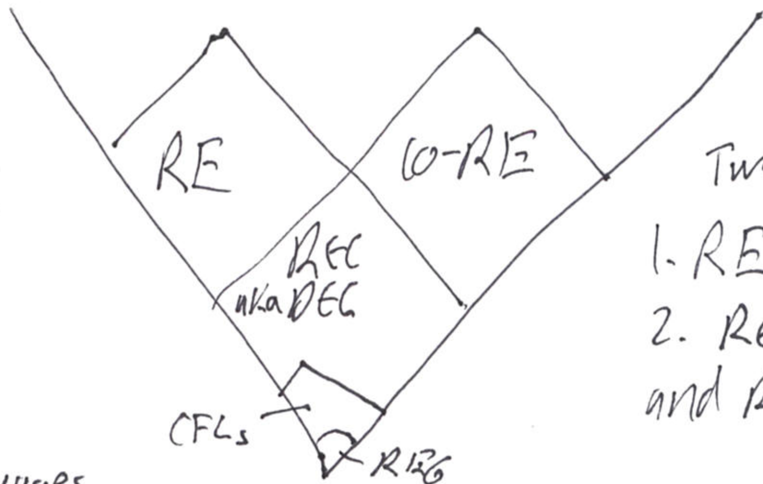
Lecture Thu 4/30 Undecidability & Reductions



Two main facts pictured:
1. RE $\cap$ co-RE = REC
2. Reductions — RE, co-RE, and REC are all closed under them

languages
RE = $\{L$: for some Turing machine M, $L = L(M)\}$.
$L \subseteq \Sigma^*$

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

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

"unpack"

RE $\cap$ co-RE = REC   level of classes

is a statement about languages: For all languages L, (L is recognizable & $\tilde{L}$ is recognizable) $\iff$ L is decidable.

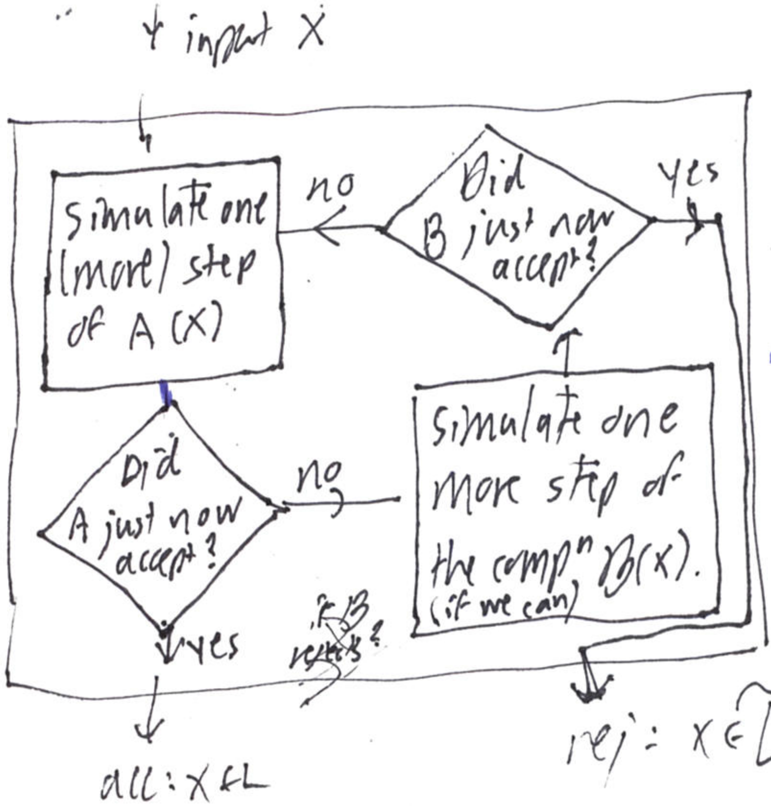
Theorem 4.22: A language is Turing recognizable & co-TM recognizable iff it is decidable.

(unpack as statement about machines & strings.)

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

For $\Rightarrow$ we need to show: 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.

Proof: We are given partial algorithms A & B:
partial ≡ when they dont accept an input x, they might not halt.



Fact: For all $X \in \Sigma^*$, either $X \in L$, whereupon A(X) eventually will halt and accept ... OR $X \in \tilde{L}$, whereupon B(X) will eventually halt and accept.

M

Goals: $L(M) = L$
M always halts.

$\boxed{= L(A) = \sim L(B)}$

(Either machine might halt & reject, but we dont need to rely on that.)

By the Fact, the big loop always eventually exits with the correct answer, so $L(M) = L$ and M is total ← ie halts for all inputs.

∴ L is decidable. ☒

2. Definition: A language B is mapping reducible to a language C, written $B \leq_m C$, if there is a computable function $f: \Sigma^* \to \Sigma^*$ s.t.

$$\text{for all } x, \quad x \in B \iff f(x) \in C.$$

f is computable ≡ there is a TM T with output st. $\forall x \in \Sigma^*$: T(x) halts and outputs f(x).

Note: Because $dom(f) = \Sigma^*$, T(x) must always halt & output something, so T is total.

Note 2: A language L is decidable ⟺ the fn $\underset{\equiv L(x)}{f_L(x)} = \begin{cases} 1 & \text{if } x \in L \\ 0 & \text{ow.} \end{cases}$ is computable.

Example: Define $K_{TM} = \{ e(M) : M \text{ does accept } e(M) \}$. K_{TM} is the language of the "Self-Acceptance Problem."
Basically $K_{TM} = \sim D_{TM}$, so it is undecidable.

A_{TM} : Inst: A machine M and a string w QUES: Does M accept w?
type: Machine & a string.

K_{TM} : Inst: Just a machine M.   Ques: Does M accept the string $e(M)$?

Example: $K_{TM} \leq_m A_{TM}$ via the function $f(M) = \langle M, e(M) \rangle$

Referencing the Def^n: "B" $\equiv K_{TM}$ "C" $\equiv A_{TM}$ "x" $= M$ "$f(x)$" is a pair $\langle M, e(M) \rangle$

Then via the Def^n, we need $M \in K_{TM} \Leftrightarrow f(M) \in A_{TM}$
and f is computable.

$M \in K_{TM}$ ||| M accepts $e(M)$

$f(M) \in A_{TM}$ ||| since $f(M) = \langle M, e(M) \rangle$ and $\langle M, w \rangle \in A_{TM} \Leftrightarrow M$ accepts w,
these are $\Leftrightarrow$ to each other.

f is computable ||| f calls $e(M)$ to encode M and then f doubles up to make a pair.

Text notation w/o $e(M)$: $f(\langle M \rangle) = \langle M, \langle M \rangle \rangle$.
Notation w/o source/code distinction: $f(M) = M \# M$.

Technote: What we really have is

$$f(u) = \begin{cases} \text{if } u \text{ is the code of a TM } M, \text{ then output } \langle u, u \rangle \text{ or } \langle u, e(M) \rangle, \langle M, e(M) \rangle \text{ (all the same) } u \equiv e(M) \equiv \langle M \rangle \\ \text{if } u \text{ is not a valid code, i.e. } u \notin Range(e) \text{ or } u \notin Range(\langle \cdot \rangle) \text{ then just output } \varepsilon \text{ since } \varepsilon \notin A_{TM}. \end{cases}$$

must be defined for all $u \in \Sigma^*$.

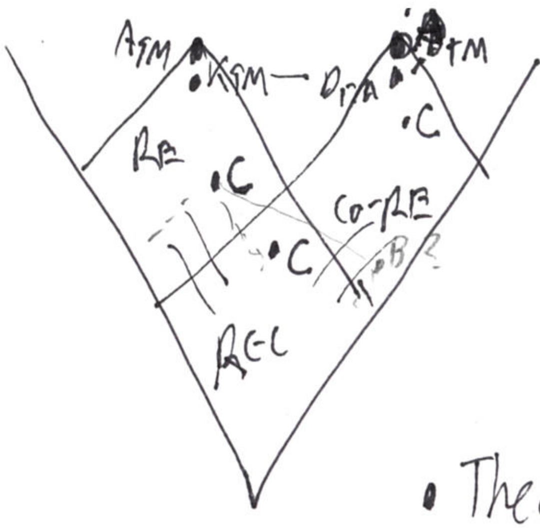
We need Ran(e) to be decidable.

WE MAY ALWAYS ASSUME RANGES OF ENCODINGS ARE NICE
ARE NICE.

Basically $f(M) = M, M$.

$M \in K_{TM} \Leftrightarrow f(M) \in A_{TM}$.

- Since we initially showed D_{TM} and hence K_{TM} are undecidable, it follows from this that A_{TM} is undecidable.
- Since A_{TM} is c.e., this shows so is K_{TM}.



- $\bar{L}$ is a mirror image of L
- If $B \le_m C$, then this is indicated by a steeper than 45° angle from B up to C.
- Theorem: If $B \le_m C$, then:

a) If C is decidable, then B is decidable.

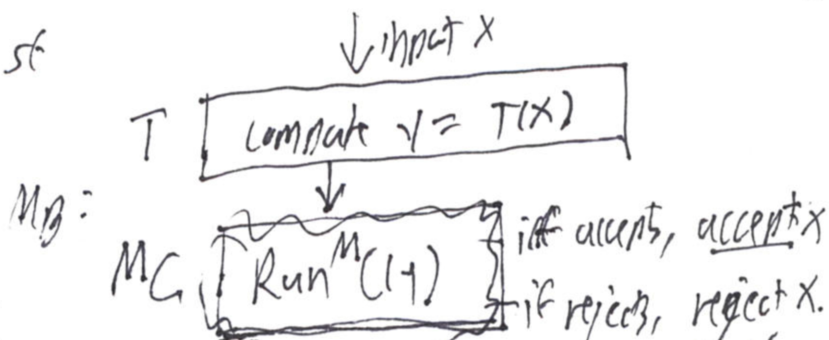
b) If C is recognizable, then B is recognizable (re. or c.e.) Synonyms

c) If C is co-c.e., then B is co-c.e.

Proof: (a) Given C is decidable, this means we have a total TM M_C such that $L(M_C) = C$.

- Given $B \le_m C$, 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 $x \in B \iff y \in C$ so M_B accepts $x \iff y \in C \iff x \in B$ so $L(M_B) = B$. Thus B is decidable.

(b) Since we copy back the same answer, this still recognizes B if M_C merely recognizes C.

(c) follows because $x \in B \iff T(x) \in C \equiv x \in \tilde{B} \iff T(x) \in \tilde{C} \equiv \tilde{B} \le_m \tilde{C}$.

HaltTM: Inst: A TM M and an input w.
Ques. Does M(w) halt?

Show HaltTM is undecidable.

Two styles. §5.1 "vs" §5.3

§5.1: suppose we had a decider R for the HaltTM problem. Then we could build a decider S for ATM as follows (...) But S cannot exist since ATM is undecidable. So HaltTM is undec.

5

Text Algm: Input $\langle M, w\rangle$. instance of A_{TM}.

S :: 1. Use the hypothetical decider $R(M, w)$

- If it says no, $M(w)$ doesn't halt, then M can't possibly accept, so $\langle M, w\rangle \notin A_{TM}$.
- If it says yes, then it is safe to run $M(w)$ to see if it accepts, and copy back the answer.

Given R deciding $Halt_{TM}$, S would be a decider for A_{TM}, but it can't be.
$\therefore$ R cannot exist
$\therefore$ $Halt_{TM}$ is undecidable.

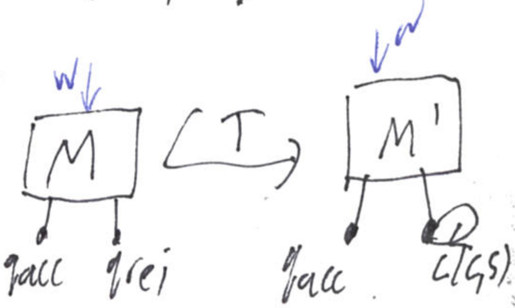
Show: $A_{TM} \leq_m Halt_{TM}$.

§5.3 "style" Build a Mapping $T(M, w) = (M', w)$ s.t.

$\langle M, w\rangle \in A_{TM} \iff \langle M', w\rangle \in Halt_{TM}$

$|||$

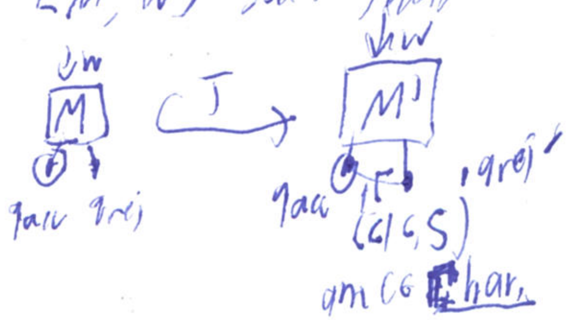
M accepts w $\iff$ $M'(w)$ halts.



Since rejecting by M causes an infinite loop in M',
$M(w)$ accepts $\iff$ $M'(w)$ halts. And T just adds an arc, so computable.

Extra: Conversely, we can show $Halt_{TM} \leq A_{TM}$ as follows: We need a mapping T that given "an M and a w" makes $T(M, w) = \langle M', w\rangle$ such that

M halts on w $\iff$ M' accepts w.



Using the text's "normal form" in which all halting computations end either at q_{acc} or at q_{rej}, we build M' by adding arcs on every character from q_{rej} to q_{acc}. Then to be pro-forma, we no longer consider $q = q_{rej}$ to be "the" reject state but just an ordinary state q, and we make a new reject state q'_{rej} that goes unused. ⊠