Some techniques for Undecidability / Recognizability.

The general technique for showing a problem is "easy" (decidable, recognizable and, later, poly-time decidable) is to come up with an algorithm (TM) that solves it (decides, recognizes, or poly-time decides it).

The general technique for showing a problem is "hard" is to show that,

if ∃ a black box solver for the problem in question

then ∃ a "solver of the unsolvable"

(e.g. a decider for the undecidable).

Solver of the unsolvable = " on input ⟨...⟩ ...

Note: all the pink

call black box

ACCEPT.

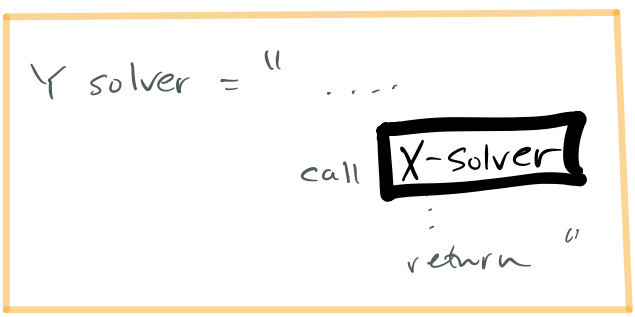
code must be computable so that,

else : "REJECT."

if black box works, the code undeniably "solves the unsolvable."

When we show that, if we have a solver for X then we can construct a solver for Y that is referred to as

"Reducing Y to X"



all this other stuff is doable.

Eg A_{TM} "reduces to" Halt "Turing-reduces to"

$A_{TM} \leq_{TM}$ Halt

Techniques for reducing one language to another.

1. Run machines in parallel to avoid waiting for a looping machine.

$$U = \{ \langle M_1, M_2, w \rangle \mid M_1, M_2 \text{ are TMs, and } w \text{ is accepted by } M_1 \text{ or } M_2 \text{ or both} \}$$

Claim: U is recognizable.

Proof: Here is an algorithm that recognizes U.

U-recognizer = "on input $\langle M_1, M_2, w \rangle$,

1. In parallel, run M_1 on w and run M_2 on w.
   - If either one accepts, ACCEPT
   - If both reject, REJECT."

Running in parallel means:
- take a step of M_1's computation on w
- take a step of M_2's computation on w,

and repeat, until one halts. If it

accepts, halt and ACCEPT.

if it rejects, continue running the other TM.
if it accepts, ACCEPT
if it rejects, REJECT.

This way you do not have to wait for a looping machine <u>if</u> the other one halts.

Another technique: **Rewiring a TM.**

$Rej = \{ \langle M, w \rangle \mid M \text{ is a TM that rejects } w \}$

Claim: Rej is undecidable.

Proof: We reduce A_{TM} to Rej.

Assume $\exists$ a Rej-decider X

We construct a A_{TM}-decider A as follows:

A = " on input $\langle M, w \rangle$ where M is a TM, w a string

1. in M, make each transition to h_r go instead to h_a
and make each transition to h_a go instead to h_r. Call the resulting TM M_{rej}

2. Run X on $\langle M_{rej}, w \rangle$
if X accepts, ACCEPT.
if X rejects, REJECT."

Since X is a decider for Rej, X will accept $\langle M_{rej}, w \rangle$ iff M, on input w, goes to h_a; and in that case, A accepts $\langle M, w \rangle$.

If M loops or rejects w, X will reject $\langle M_{rej}, w \rangle$, and A rejects $\langle M, w \rangle$ ... ie A decides A_{TM}

$\Rightarrow\Leftarrow$

$\therefore$ $\nexists$ a decider for Rej. ▨

And don't forget our other technique,
Enumerable parallelism:

$NotEmpty_{TM} = \{ \langle M \rangle \mid M \text{ is a TM that accepts some string} \}$

Claim: $NotEmpty_{TM}$ is recognizable.

Proof: NE is a TM that recognizes $NotEmpty_{TM}$, where

NE = " on input $\langle M \rangle$ where M is a TM:

1. Let $i = 1$.
   1.1 Let S_i = the first i strings of Σ^* in shortlex order
      1.1.1 Run M on each string in S_i for i steps (each).
      If M ever accepts, ACCEPT.
      otherwise, $i = i+1$, go to 1.1 "

Argue here that NE recognizes $NotEmpty_{TM}$.

Another technique:

TMs constructing special-purpose TMs which they can feed as input to other TMs!

$$LoopsOnOne = \{ \langle M_1, M_2, w \rangle \mid \text{exactly one of } M_1 \text{ and } M_2 \text{ loops on } w \}$$

Claim: LoopsOnOne is undecidable.

Proof: We reduce A_{TM} to LoopsOnOne.

Suppose $\exists$ a TM LOO that decides LoopsOnOne.

Then we can construct a TM A that decides ATM as follows:

A = "on input $\langle M, w \rangle$, where M is a TM and w a string,

1. Construct a TM Looper that loops on all inputs
2. Run LOO on $\langle M, Looper, w \rangle$
   if LOO rejects, REJECT.

else run M on w

if M accepts, ACCEPT

else if M rejects, REJECT. "

Argue here why A decides A_{TM}.

Note added after class:

$NotE_{TM} = \{ \langle M \rangle \mid M \text{ accepts at least one string} \}$

Claim: $NotE_{TM}$ is not decidable.

Proof: We reduce A_{TM} to $NotE_{TM}$. S ∃ a TM N that decides $NotE_{TM}$. Then we can construct a TM A that decides A_{TM} as follows:

A = " on input $\langle M, w \rangle$, where M is a TM, w a string:

1. Construct a TM M_w that does the following:

M_w = "1. Ignore the input.

2. Run M on w.

If M accepts, ACCEPT.

If M rejects, REJECT. "

2. Run N on $\langle M_\omega \rangle$.
   if N accepts, ACCEPT.
   if N rejects, REJECT."

We argue that A decides A_{TM}:

The language $L(M_\omega)$ is either Σ^* (if M accepts ω) or $L(M_\omega) = \emptyset$ (if M does not accept ω).

Hence

M does not accept $\omega \Rightarrow L(M_\omega) = \emptyset \Rightarrow N$ rejects $\langle M_\omega \rangle \Rightarrow A$ REJECTS $\langle M, \omega \rangle$

M accepts $\omega \Rightarrow L(M_\omega) \neq \emptyset \Rightarrow N$ accepts $\langle M_\omega \rangle \Rightarrow A$ ACCEPTS $\langle M, \omega \rangle$.

$\therefore$ A decides A_{TM}.

$\Rightarrow\Leftarrow$ A_{TM} is undecidable.

$\therefore$ N does not exist, and $\overline{E_{TM}}$ is undecidable. ▨