The language A_{TM}

- Consider the language:
  A_{TM} = {<M,w> | M is a TM and M accepts w}
  - NOTE: <A,B,…> is just a string encoding the objects A, B, …
  - In particular, <M,w> is a string listing the components of TM M followed by the string w
  - Given input <M,w>, it should be easy to extract the info about M and to simulate M on w (try writing a TM to do this!)
- What can we say about A_{TM}?

A_{TM} is Turing-recognizable

- A_{TM} is Turing-recognizable: Recognizer TM U for A_{TM}:

  On input string <M,w>:
  Simulate M on w.
  ACCEPT <M,w> if M halts & accepts w;
  REJECT <M,w> if M halts & rejects
  (Loop (& thus reject <M,w>) if M ends up looping).
  U accepts <M,w> iff M accepts w, i.e. L(U) = A_{TM}

"Universal" TM
(can simulate any TM)

Yeah, but is it decidable?!!

Is A_{TM} decidable?

- No! A_{TM} = {<M,w> | M is a TM and M accepts w} is undecidable! 1-slide Proof (by Contradiction):
  1. Assume A_{TM} is decidable ⇒ there's a decider H, L(H) = A_{TM}
  2. H on <M,w> = ACC if M accepts w
     REJ if M rejects w (by halting in q_{REJ} or looping)
  3. Construct new TM D: On input <M>:
     Simulate H on <M,<M>> (here, w = <M>)
     If H accepts, then REJ input <M>
     If H rejects, then ACC input <M>
  4. What happens when D gets <D> as input?
     D rejects <D> if H accepts <D,<D>> if D accepts <D>
     D accepts <D> if H rejects <D,<D>> if D rejects <D>

  Either way: Contradiction! D cannot exist ⇒ H cannot exist
  Therefore, A_{TM} is not a decidable language.

Undecidability Proof uses Diagonalization

Input strings

List of TMs

	<M_1>	<M_2>	<M_3>	…
M_1	ACC	REJ	*loop*	…
M_2	REJ	*loop*	ACC	…
M_3	ACC	ACC	REJ	…
:	:	:	:	

If H exists →

D outputs opposite of diagonal

	<M_1>	<M_2>	<M_3>	…	<D>
M_1	ACC	REJ	**REJ**	…	ACC
M_2	REJ	**REJ**	ACC	…	ACC
M_3	ACC	ACC	REJ	…	REJ
:	:	:	:	:	:
D	REJ	ACC	ACC	…	??

D on <M_i> accepts if and only if M_i on <M_i> rejects.
So, D on <D> will accept if and only if D on <D> rejects!
A contradiction ⇒ H cannot exist!
Therefore, A_{TM} is not a decidable language.

One Last Concept: Reducibility

- How do we show a new problem B is undecidable?
- Idea: Show that A_{TM} is reducible to the new problem B
  - What does this mean and how do we show this?
- Show that if B was decidable, then you can use the decider for B as a *subroutine* to decide A_{TM}
  - Contradiction, therefore B must also be undecidable

The Halting Problem is Undecidable (Turing, 1936)

- Example: Halting Problem: Does TM M halt on input w?
  - Equivalent language: $A_H = \{ \langle M,w \rangle \mid \text{TM M halts on input w}\}$
  - Need to show A_H is undecidable
  - We know $A_{TM} = \{\langle M,w \rangle \mid \text{TM M accepts w}\}$ is undecidable
- Show A_{TM} is reducible to A_H (Theorem 5.1 in text)
  - Suppose A_H is decidable $\Rightarrow$ there's a decider M_H for A_H
  - Then, we can construct a decider D_{TM} for A_{TM}:

    On input <M,w>, run M_H on <M,w>.
    - If M_H rejects, then REJ (this takes care of M looping on w)
    - If M_H accepts, then simulate M on w until M halts
    - If M accepts, then ACC input <M,w>; else REJ

    $L(D_{TM}) = A_{TM} \Rightarrow A_{TM}$ is decidable! Contradiction $\Rightarrow A_H$ is undecidable
- E.g. 2: Show $E_{TM} = \{\langle M \rangle \mid \text{M is a TM and } L(M) = \varnothing\}$ is undecidable (Theorem 5.2 in text)