The Halting Problem

\[ A_{TM} = \{ <M,w> \mid M \text{ is a TM that accepts string } w \} \]

- Turing-recognizable?
  YES \quad \text{bec:} \quad \text{"Algo":} \quad 1) \text{ run the TM } M \text{ on } w
  2) \text{ if } M \text{ accepts, return true (accepted)}
  3) \text{ if } M \text{ rejects, return false (not accepted)}

- Turing-decidable?
  NO \quad \text{see the next slide}

in quotes bec. might never finish \( \text{(if M goes through infinite steps on } w) \)
The Halting Problem

Thm 4.11: $A_{TM}$ is not decidable.
Recall: $A_{TM} = \{ <M, w> \mid M \text{ is a TM that accepts string } w \}$

By contradiction, assume $A_{TM}$ is decidable.

Let acceptance $(\text{File } M, \text{File } w)$ be a function that decides $A_{TM}$.

We will create our own weird func, using the acceptance func.

```plaintext
func weird (File T)
  run acceptance (T, T)
  return the opposite of 5
```

What happens if weird (weird)?

- It returns false if acceptance (weird, weird) = true
- It returns true if weird does not accept weird (i.e., weird(weird) = false)

IMPOSSIBLE!

hence acceptance func cannot exist!
Thm 4.22: A language $L$ is decidable iff $L$ is Turing-recognizable and $\overline{L}$ is Turing-recognizable (we say that $L$ is co-Turing-recognizable).

$\Rightarrow$ if $L$ is decidable is $L$ also T-recognizable? $\checkmark$

$\Rightarrow$ T-recognizable? $\checkmark$ YES, just switch accept & reject states $\checkmark$

$\Leftarrow$ have a TM $M_1$ for $L$

$\Leftarrow$ have a TM $M_2$ for $\overline{L}$

how to create a TM-decider $M'$ for $L$

sketch (works):

1. alternate running one step of $M_1$ with one step of $M_2$
2. if $M_1$ accepts, $M'$ accepts
3. if $M_2$ accepts, $M'$ rejects

Note: exactly one of $M_1, M_2$ will accept bec. recognize $L$ and $\overline{L}$

Cor 4.23: $A_{TM}$ is not Turing-recognizable.

bec. $A_{TM}$ is T-recognizable & if $\overline{A_{TM}}$ were T-recognizable, $A_{TM}$ would be decidable