Undecidable Problems

What can go wrong tape memory and the ability to move around?
Decidable problems: problems we write programs to solve.
Programs as opposed to Machines.
An unsolvable problem.
Last Time

1. Stack memory helps to solve $L = \{0^n \# 1^n\}$ but isn’t enough for $L = \{0^n \# 1^n \# 0^n\}$.

2. Tape memory: read, write and move around capability.

3. Using tape memory to solve $L = \{0^n \# 1^n \# 0^n\}$.
   
   High-level description of a Turing Machine (TM) — *pseudo-code*.

4. Low-level detailed description of a TM to solve $L = \{0^n \# 1^n \# 0^n\}$ — *machine-code*.

5. High-level description of a TM for multiplication, $L = \{0^i \# 1^j \# 0^{i \times j} | i, j \geq 1\}$.
   
   What about the detailed low-level description?

6. A TM is more powerful than an actual computer which has finite memory.
Today: Decidable and Undecidable Problems

1. The infinite loop.

2. Recognizable languages (allowing the infinite loop for “reject”).

3. Decidable languages (no-infinite loops allowed).

4. Programs versus computing machines.
   - The string (text/mathematical) description of a TM.
   - The Universal Turing Machine (UTM).

5. Feeding the string $< TM >$ as input to TM (TM is a Turing machine).

6. The language corresponding to “program verification”
   - A simpler language: TMs that reject “themselves”.

7. The simpler language is undecidable.

8. We cannot automate program verification!

9. Challenge problem #2, though it looks harmless, is UNSOLVABLE!