Some challenges to logic

Limits on
- truth

Liar's Paradox: 'I am lying'

Russell set $R = \{ x | x \not\in x \}$

Cantor: Power($\{0, 1, 2, \ldots\}$)

Sorites: heap (minus one grain)

Turing: Halting Problem
Some challenges to logic

Limits on
- truth
  Liar’s Paradox: ‘I am lying’
Some challenges to logic

Limits on

- truth
  Liar’s Paradox: ‘I am lying’

- sets/membership ∈
  Russell set $R = \{x \mid \text{not } x \in x\}$
Some challenges to logic

Limits on

- truth
  - Liar’s Paradox: ‘I am lying’

- sets/membership $\in$
  - Russell set $R = \{ x \mid \text{not } x \in x\}$

- countability
  - Cantor: Power($\{0, 1, 2, \ldots\}$)
Some challenges to logic

Limits on
- truth
  Liar’s Paradox: ‘I am lying’
- sets/membership ∈
  Russell set \( R = \{ x \mid \text{not } x \in x \} \)
- countability
  Cantor: \( \text{Power}(\{0, 1, 2, \ldots\}) \)
- change
  Sorites: heap (minus one grain)
Some challenges to logic

Limits on
  - truth
    Liar’s Paradox: ‘I am lying’
  - sets/membership $\in$
    Russell set $R = \{ x \mid \text{not } x \in x \}$
  - countability
    Cantor: $\text{Power}(\{0, 1, 2, \ldots\})$
  - change
    Sorites: heap (minus one grain)
  - computability
    Turing: Halting Problem
Tolerance and Sorites chains

A unary relation $P$ is tolerant up to $\text{near}_P$ if

$$P(x)$$ whenever $\text{near}_P(x, y)$ and $P(y)$. 

**Example 1.** $P(x)$ is $\text{heap}(x)$, $\text{near}_P(x, y)$ is $|x - y| \leq 1$ grain
Tolerance and Sorites chains

A unary relation $P$ is tolerant up to near$_P$ if

\[ P(x) \text{ whenever } \text{near}_P(x, y) \text{ and } P(y). \]

**Example 1.** $P(x)$ is heap($x$),

\[ \text{near}_P(x, y) \text{ is } |x - y| \leq 1 \text{ grain} \]

**Example 2.** $P(x)$ is walking-distance($x$),

\[ \text{near}_P(x, y) \text{ is } |x - y| \leq 1 \text{ foot} \]
Tolerance and Sorites chains

A unary relation $P$ is *tolerant up to near$_P$* if

$$P(x) \text{ whenever near}_P(x, y) \text{ and } P(y).$$

**Example 1.** $P(x)$ is $heap(x)$,

near$_P(x, y)$ is $|x - y| \leq 1$ grain

**Example 2.** $P(x)$ is $walking-distance(x)$,

near$_P(x, y)$ is $|x - y| \leq 1$ foot

**Example 3.** $P(x)$ is $young(x)$, $sunny(x)$,

near$_P(x, y)$ is $|x - y| \leq 1$ picosec
Tolerance and Sorites chains

A unary relation $P$ is *tolerant up to near$_P$* if

$$P(x) \text{ whenever near}_P(x, y) \text{ and } P(y).$$

**Example 1.** $P(x)$ is $heap(x)$,

$$\text{near}_P(x, y) \text{ is } |x - y| \leq 1 \text{ grain}$$

**Example 2.** $P(x)$ is $walking-distance(x)$,

$$\text{near}_P(x, y) \text{ is } |x - y| \leq 1 \text{ foot}$$

**Example 3.** $P(x)$ is $young(x)$, $sunny(x)$,

$$\text{near}_P(x, y) \text{ is } |x - y| \leq 1 \text{ picosec}$$

A *Sorites chain* is a sequence $y_1, \ldots, y_n$ such that $P$ holds of $y_1$ but not $y_n$, even though $\text{near}_P(y_i, y_{i+1})$ for $1 \leq i < n$. 
The Halting Problem

Given a program $P$ and data $D$, return either 0 or 1 (as output), with 1 indicating that $P$ halts on input $D$

$$\text{HP}(P, D) := \begin{cases} 1 & \text{if } P \text{ halts on } D \\ 0 & \text{otherwise} \end{cases}$$
The Halting Problem

Given a program $P$ and data $D$, return either 0 or 1 (as output), with 1 indicating that $P$ halts on input $D$

$$
HP(P, D) := \begin{cases} 
1 & \text{if } P \text{ halts on } D \\
0 & \text{otherwise}
\end{cases}
$$

**Theorem (Turing)** *No TM computes HP.*

The proof is similar to the Liar’s Paradox distributed as follows

- **H:** ‘L speaks the truth’
- **L:** ‘H lies’

with a spoiler L (exposing H as a fraud).
Proof of uncomputability

Given a TM $P$ that takes two arguments, we show $P$ does not compute HP by defining a TM $\overline{P}$ such that

$$P(\overline{P}, \overline{P}) \neq \text{HP}(\overline{P}, \overline{P}).$$
Proof of uncomputability

Given a TM \( P \) that takes two arguments, we show \( P \) does not compute HP by defining a TM \( \overline{P} \) such that

\[
P(\overline{P}, \overline{P}) \neq \text{HP}(\overline{P}, \overline{P}) .
\]

Let

\[
\overline{P}(D) :\overset{\sim}{=} \begin{cases} 
1 & \text{if } P(D, D) = 0 \\
\text{loop} & \text{otherwise.}
\end{cases}
\]

and notice

\[
\text{HP}(\overline{P}, \overline{P}) = \begin{cases} 
1 & \text{if } \overline{P} \text{ halts on } \overline{P} \\
0 & \text{otherwise} \quad \text{(def of HP)}
\end{cases}
\]

\[
= \begin{cases} 
1 & \text{if } P(\overline{P}, \overline{P}) = 0 \\
0 & \text{otherwise} \quad \text{(def of } \overline{P})
\end{cases}
\]
There is a TM that meets the positive part of HP (looping exactly when HP asks for 0), in view of the existence of a

**Universal Turing Machine**: a TM $U$ that runs $P$ on $D$

$$U(P, D) \simeq P(D)$$

for any given TM $P$ and data $D$. 
Semi-solvability of HP

There is a TM that meets the positive part of HP (looping exactly when HP asks for 0), in view of the existence of a

**Universal Turing Machine**: a TM $U$ that runs $P$ on $D$

$$U(P, D) \sim P(D)$$

for any given TM $P$ and data $D$.

Your first assessed assignment will be to encode $U$ as a Prolog program. (Details later.)

N.B. A key idea behind Prolog is that a program is essentially a knowledge representation.