What computers just cannot do. (Part II)

Adam Finkelstein
Administrivia

- Midterm - take home during week of 3/12
  - Review session Fri 3/9 3pm Friend 005
  - Last year’s exam linked under “extras” on web
- Back-off from pseudocode
  - H2, Q6: now optional
  - On midterm: be able to read, but *not* write it
Epimenides Paradox

- Κρήτες ἁεί ψεύσται
- “Cretans, always liars!”
- But Epimenides was a Cretan!’
  (can be resolved…)

- More troubling: “This sentence is false.”
Barber Paradox

- Town with one male barber
- Each man either shaves or sees the barber
- Barber shaves the men who do not shave themselves
- Does barber shave himself? Contradiction either way!

Bertrand Russell (1872 –1970)
Recap from last time

- Turing-Post computational model:
  - Greatly simplified model
  - Infinite tape, each square either 0/1
  - Program = finite sequence of instructions (only 6 types!)
  - Unlike pseudocode, no conditionals or loops, only “GOTO”
  - \text{code}(P) = \text{binary representation of program } P
Motivation

Simplify!

(Get to the heart of the matter)
Doubling program

1. PRINT 0
2. GO LEFT
3. GO TO STEP 2 IF 1 SCANNED
4. PRINT 1
5. GO RIGHT
6. GO TO STEP 5 IF 1 SCANNED
7. PRINT 1
8. GO RIGHT
9. GO TO STEP 1 IF 1 SCANNED
10. STOP
Halting

Program

1. PRINT 0
2. GO LEFT
3. GO TO STEP 2 IF 1 SCANNED
4. PRINT 1
5. GO RIGHT
6. GO TO STEP 5 IF 1 SCANNED
7. PRINT 1
8. GO RIGHT
9. GO TO STEP 1 IF 1 SCANNED
10. STOP

Program halts on this input data if STOP is executed in a finite number of steps.
Some facts

- **Fact 1:** Every pseudocode program can be written as a T-P program, and vice versa.

- **Fact 2:** There is a universal T-P program.

![Diagram](image-url)

\[ \begin{array}{cccccccccccccccc}
  & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & \ldots \\
\end{array} \]

- \( U \) simulates \( P \)'s computation on \( V \)

\( \text{code}(P) \)
Discussion

Is there a universal pseudocode program?

How would you write it?
Halting Problem

- Decide whether $P$ halts on $V$ or not

- **Cannot be solved!** Turing proved that *no Turing-Post program can solve Halting Problem*
Short detour

- Proof by contradiction…
- Feeding a program to itself…
Proof By Contradiction

- Suppose statement S is true
- Make series of logical deductions from S
- Arrive at deduction that is clearly false
- ... therefore S must be false
Feeding a program to itself

- A python program to count lines:
  ```python
  import sys
  count = 0
  for line in sys.stdin.readlines():
      count = count + 1
  print count
  ```

- Run this program using itself as input:
  ```bash
  % python count_lines.py < count_lines.py
  5
  ```
Proof for halting problem

Suppose we had a solution:

```python
would_it_stop( program, data ):  
    if( something terribly clever ) {  
        report TRUE;
    } else {  
        report FALSE;
    }
```

This version due to Craig Kaplan, U of Waterloo
http://www.cgl.uwaterloo.ca/~csk/halt/
Proof for halting problem

- Feed a program to itself:

```python
stops_on_self(program):
    report would_it_stop(program, program);
```
Proof for halting problem

Now let’s mix things up:

```java
bobs_yer_uncle( program ):  
    if( stops_on_self( program ) ) {  
        while( TRUE ) { do nothing }  (loop forever)  
    } else {  
        report TRUE;  
    }
```
Proof for halting problem

- Finally, run bobs_yer_uncle on itself

- Two possible outcomes:
  - Never halts, or
  - Halts and reports TRUE
Proof for halting problem

- Consider case of infinite loop
  - \texttt{stops\_on\_self} (bobs\_yer\_uncle) reports TRUE
  - \texttt{would\_it\_stop} (bobs\_yer\_uncle, bobs\_yer\_uncle) reports TRUE
  - … but then bobs\_yer\_uncle would stop when fed itself
  - … contradiction!
Proof for halting problem

Consider case where it reports TRUE

- stops_on_self(bobs_yer_uncle) reports FALSE
- would_it_stop(bobs_yer_uncle, bobs_yer_uncle) reports FALSE
- … but then bobs_yer_uncle would run forever
- … contradiction!
Lessons to take away

- Computation is a very simple process (can arise in unexpected places)

- Universal Program

- No real boundary between hardware, software, and data

- No program that decides whether or not mathematical statements are theorems.
Age-old mystery: Self-reproduction.

How does the seed encode the whole?
Fact: for every program $P$, there exists a program $P'$ that has the exact same functionality except at the end it also prints code($P'$) on the tape.
Next time

Graphics…