What computers just cannot do

COS 116
3/7/2006
Instructor: Sanjeev Arora
Epimenides Paradox

- \textit{Κρητες έες ψεῦσται}
- “Cretians, always liars!”
- But Epimenides was a Cretian!
  (can be resolved…)

- More troubling: “This sentence is false.”
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.
Discussion

Is there a universal pseudocode program?

How would you write it?
Composing programs $P_1, P_2$

Desired: A T-P program that, on input $V$:

- First runs $P_1$ on $V$
- If the previous step halts, runs $P_2$ on the new tape contents

Ideas??
Halting Problem

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

- **Cannot be solved!** Turing proved that no Turing-Post program can solve the Halting Problem
Proof (by contradiction)

Suppose $H$ is a Turing-Post program solving the Halting problem

Use it to write a new program $P_0$:
1. On input $V$, $P_0$ checks if $V$ is the code to a Turing-Post program
2. If not, HALT
3. Else, use a Doubling Program to get $V, V$
4. Run $H$ on $V, V$
5. If $H$ says “doesn’t halt”, then HALT immediately
6. Otherwise, go into an infinite loop
Proof (cont’d)

1. On input \( V \), check if \( V \) is the code to a Turing-Post program
2. If not, HALT
3. Else, use a Doubling Program to get \( V, V \)
4. Run \( H \) on \( V, V \)
5. If \( H \) says “doesn’t halt”, then HALT immediately
6. Otherwise, go into an infinite loop

- But does \( P_0 \) halt on code(\( P_0 \))?
- If it doesn’t, then at step 5 it should halt
- If it does, then it should reach step 6 and go into an infinite loop (i.e. not halt!)
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.
Discussion next time: How to write a self-reproducing T-P program

Hint: The idea is the following:

Write the following twice, the second time in quotes “Write the following twice, the second time in quotes”