Today’s Topics

Computer Science
Noncomputability

Upcoming
Special Topic

Reading
*Great Ideas*, Chapter 15

On the Limits of Computing

- **Noncomputability**
  - Certain Problems *Not* Amenable to Computer Solution
  - Examples given here may seem strained and artificial.
  - However, computers have *very real* limitations

- Will Use Two Approaches to *Prove* Noncomputability
  1. Show *Existence* of Noncomputable Functions
  2. Prove That Certain Programs *Can Not Exist*

Existence of Noncomputable Functions

- **Approach**
  - Matching up Programs and Functions
  - E.g., assume 3 functions, only 2 programs
  - Without details, conclude one function has no program

- **Have:** *Uncountable Infinity* of Functions Mapping int to int
  - How can we show that is true?
  - Functions can be seen as columns in tables
  - Put all functions into a huge (*infinite*) table
  - Show that even that cannot hold them all
  - *Can you identify the functions in the following table?*

Table of *All* Integer to Integer Functions

|    | 1 | 1 | 2 | 6 | 0 | 0 | 8 | 2 | 1 | 4 |...
|----|---|---|---|---|---|---|---|---|---|---|---
| 2  | 4 | 4 | 7 | 0 | 1 | 8 | 4 | 1 | 7 |...
| 3  | 9 | 6 | 8 | 0 | 0 | 8 | 6 | 2 | 10 |...
| 4  | 16| 8 | 9 | 1 | 1 | 8 | 16| 3 | 13 |...
| 5  | 25| 10| 10| 1 | 0 | 8 | 10| 5 | 16 |...
| 6  | 36| 12| 11| 1 | 1 | 8 | 36| 8 | 19 |...
| 7  | 49| 14| 12| 1 | 0 | 8 | 14| 13| 22 |...
| 8  | 64| 16| 13| 1 | 1 | 8 | 64| 21| 25 |...
| 9  | 81| 18| 14| 1 | 0 | 8 | 18| 34| 28 |...
|    |   |   |   |   |   |   |   |   |   |   |...
A Function NOT in this (inclusive!) Table

| 1+1 | 1 | 2 | 6 | 0 | 0 | 8 | 2 | 1 | 4 |
| 2   | 4+1 | 4 | 7 | 0 | 1 | 8 | 4 | 1 | 7 |
| 3   | 9 | 6+1 | 8 | 0 | 0 | 8 | 6 | 2 | 10 |
| 4   | 16 | 8 | 9+1 | 1 | 1 | 8 | 16 | 3 | 13 |
| 5   | 25 | 10 | 10 | 1+1 | 0 | 8 | 10 | 5 | 16 |
| 6   | 36 | 12 | 11 | 1 | 1+1 | 8 | 36 | 8 | 19 |
| 7   | 49 | 14 | 12 | 1 | 0 | 8+114 | 13 | 22 |
| 8   | 64 | 16 | 13 | 1 | 1 | 8 | 64+121 | 25 |
| 9   | 81 | 18 | 14 | 1 | 0 | 8 | 18 | 34+128 |
| 10  | 100 | 20 | 15 | 1 | 1 | 8 | 100 | 55 | 31+1 |

Existance of Noncomputable Functions

- All Programs Can be Ordered (Thus Countable)
  - By size, shortest program first
  - Just use alphabetical order
- Try to Draw Lines Between Functions and Programs
  - Could draw lines from every program to every function
  - But, have proved functions uncountable...
  - Thus, There Must be Functions With NO Programs!
- Hard to come up with function that computer can't produce
  - Possible example: random generator
    (No algorithm can produce truly random number sequence)
  - Use Table
  - Program must be of finite size; Requires infinite table

Noncomputable Programs

- Programs that Read Programs
  - What programs have we used that read in programs?
  - Express programs as a single string (formatting messed up)
  - Therefore, could write program to see if there is an if statement in the program: answers YES or NO
  - How about, Does program halt?
  - Lack of while (and functions) guarantees a halt
  - Not very sophisticated
  - Not Halting for All Inputs is usually considered a Bug
- Solving the Halting Problem
  - Write specific code to check out more complicated cases
  - Gets more and more involved...

Existance of Noncomputable Functions

- Consider Following Program: Does it halt for all input?
  ```
  // input an integer value for k
  while ( (k > 1) )
  {
    if ( (k/2) * 2 == k ) // is k even?
    k = k / 2;
  else
    k = 3 * k + 1;
  }
  ```
- Try It!
  - e.g. 17: 52 26 13, 40 20 10 5, 16 8 4 2 1
  - For a long time, no one knew whether this quit for all inputs.
Proving Noncomputability

- Mathematicians have proven that no one, finite program can check this property for all possible programs
- Examples of non-computable problems
  - Equivalence: Define by same input > same output
  - Use variation of above program; not sure it ends
  - Cannot generally prove equivalence
- Use Proof by Contradiction (Indirect Proof)
- Proving non-computability
  - Sketch of proof
  - Find more details in book

Noncomputability Proof

- Assume Existence of Function `halt`
  
  ```
  string halt(string p, string x);
  ```
  
  - Inputs: `p = program, x = input data`
  - Returns: "Halts" or "Does not halt"

- Can now write:
  
  ```
  string selfhalt(string p);
  ```
  
  - Inputs: `p = program`
  - Returns: "Halts on self" or "Does not halt on self"
  - Uses: `halt(p, p)`
    - i.e.: asking if halts when program `p` uses itself as data

Noncomputability Proof.2

- Now write function `contrary`:
  ```
  void contrary();
  {
    TextField program = new TextField(1000);
    string p, answer;
    p = program.getText();
    answer = selfhalt(p);
    if (answer == "Halts on self"
    {
      while (true) // infinite loop
        answer = "x";
    }
    else
      return; // i.e., halts
  }

  "Feed it" this program.

Noncomputability Proof.3

- Paradox!
  - If `halt` program decides it halts, it goes into infinite loop and goes on forever
  - If `halt` program decides it doesn't halt, it quits immediately
- Therefore `halt` cannot exist!

- Whole classes of programs on program behavior are non-computable
  - Equivalence
  - Many other programs that deal with the behavior of a program
Living with Noncomputability

- What Does It All Mean?
  - Not necessarily a very tough constraint unless you get too greedy.
  - Programs can't do everything.
    - Beware of people who say they can!
  - Programs probably can't do things we don't know how to do...