Coming up

- Final projects:
  - final project presentations: Tue Dec 10, in this room
  - final submission due: Tue Dec 10, 11:55 PM

Project Final Presentations

- December 10, 10AM-11:15AM
- Think of this as a science fair.
- Each team will get space to set up whatever you want! Demo, or poster, or presentation on a laptop...
- You will have 4 minutes with the prof. Think about how you want to use it!
- Describe and discuss the solution, and demo the implementation.
- Will see 2 separate judges.
- Chance to see other projects too!
- Practice, practice, practice!

Today’s plan

- Evaluations
- Power of computing

Evaluations

http://owl.oit.umass.edu/partners/courseEvalSurvey/uma/

- If we get 80% participation by tomorrow:
  - Everyone gets 0.5 points of extra credit.
  - Everyone gets a chance to submit an optional extra credit assignment.

Power of Software

Can you write any program I describe to you?

Can you write:

A program HALTS?
INPUT: the source code of a method
OUTPUT: false if the method enters an infinite loop, true if it does not.
What’s HALTS? (method)?

```java
method() {
    print "hello, world";
}
```

What’s HALTS? (method)?

```java
method() {
    for (int x=0; x<5; x++)
        print "hello, world";
}
```

What’s HALTS? (method)?

```java
method() {
    for (int x=0; x<-1; x++)
        print "hello, world";
}
```

What’s HALTS? (method)?

```java
method() {
    while (true);
}
```

What’s HALTS? (method)?

```java
method() {
    int x = 785\textsuperscript{th} digit of \pi;
    if (x == 7)
        while(true);
}
```

What’s HALTS? (method)?

```java
method() {
    int x = 785\textsuperscript{th} digit of \pi;
    int y = x^x^x^x^x+1;
    int z = y^h digit of \pi;
    if (z == 0)
        while(true);
}
```
What’s HALTS?(method)?
method() {  
  int x = 785\textsuperscript{th} digit of \pi;  
  int y = x \times x \times x \times x + 1;  
  int[] z[] = the \ y\textsuperscript{th} through (x+y)\textsuperscript{th}  
  digits of \pi;  
  if (z ever repeats in \pi again)  
    while(true):  
}  

How about the general case?
• Let’s count programs. How many programs are there?

Specifications
• And how many specification are there?
  — let’s limit ourselves to simple specifications:
    • given a set of numbers, e.g., \{2, 4, 6\}
    • on input \(i\), return 1 if \(i\) is in the set, and 0 otherwise

First 64 programs
• How many of our specifications can I solve with 64 programs?
  (a) 64  
  (b) 32  
  (c) 8  
  (d) 6  
  (e) 2

First 64 programs
• With 64 programs, how large can my specification sets get (if I am being compact)
  (a) 64  
  (b) 32  
  (c) 8  
  (d) 6  
  (e) 2

  \{\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\},
  \{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\}, \{1, 6\}, \{2, 3\}...
  \{1, 2, 3\}, \{1, 3, 4\}, \ldots, \{1, 2, 3, 4\}, \ldots, \{1, 2, 3, 4, 5\} \ldots

set size -> number of specs
• Suppose I can only write 4 programs.
• I start with the smallest set specification:
  {}
• that’s 1 program. (return false on all inputs)
• With 4 programs, I can do
  \{\}, \{1\}, \{2\}, \{1, 2\}
Scalability Problem

- To cover subsets of a set of \( n \) numbers, I need \( 2^n \) programs.
- But I only have as many programs as there are natural numbers.
- That's exponentially smaller than the number of specifications there are.

Can't do it for all subsets!

---

Can HALTS? exist?

- Imagine that you wrote HALTS?
- I will write a new program NALTS?:

  ```
  NALTS?(Method p) {
    if (HALTS?(p)==false) return 1;
    else while (true);
  }
  ```

  Key: run the program on itself
  What is the value of NALTS?(NALTS?)

---

What is the value of NALTS? (NALTS?)

- Two cases:
  1. If NALTS?(NALTS?) goes into an infinite loop, then HALTS?(NALTS?)==true, which means that NALTS? terminates. So case 1 is impossible.
  2. If NALTS?(NALTS?) does not go into an infinite loop, then HALTS?(NALTS?)==false, which means that NALTS? does not terminate. So case 2 is impossible.

---

Conclusion

- The program HALTS cannot exist!
- Many programs cannot exist!

- Learn more in CS 401 or CS 601

---

Zero-Knowledge Proofs

How can I prove to you I know X without telling you anything about X?