Programming Languages and Techniques (CIS120)

Bonus Lecture

April 12, 2013

Consequences of “Code is Data”
Code is Data

Note: most images have been removed from this version of the presentation.
Code is Data

- A Java source file is just a sequence of characters.
- We can represent programs with Strings!

```java
String p_0 = "class C { public static void
    main(String args[]) {...}";
String p_1 = "class D { public static void
    main(String args[]) {...}";
...
String p_123123 = // solution to HW09
...
String p_93919113414 = // code for Eclipse
...
```
Consequence 1: Tools and Compilers
Tools and Compilers

• We can create programs that manipulate programs.

• Example 1: Eclipse
  – Note that Eclipse manipulates a representation of Java programs
  – Eclipse is written in Java itself
  – You could use Eclipse to edit the code for Eclipse... ?!

• Example 2: Compiler
  – The Java compiler takes a representation of a Java program
  – It outputs a “low-level” representation of the program as a .class file (i.e. Java byte code)
  – Could also compile to other representations, e.g. x86 “machine” code
Example Compilation: Java to X86

class Point {
    int x;
    int y;
    Point move(int dx, int dy) {
        x = x + dx;
        y = y + dy;
        return this;
    }
}

.globl __fun__Point.move
__fun__Point.move:
pushl %ebp
    movl %esp, %ebp
    subl $4, %esp
__5:
    movl 8(%ebp), %eax
    movl 4(%eax), %eax
    movl %eax, -4(%ebp)
    addl %ecx, -4(%ebp)
    movl -4(%ebp), %ecx
    movl 8(%ebp), %eax
    movl %ecx, 4(%eax)
    movl 8(%ebp), %eax
    movl 0(%eax), %eax
    movl %eax, -4(%ebp)
    movl 16(%ebp), %ecx
    addl %ecx, -4(%ebp)
    movl -4(%ebp), %ecx
    movl 8(%ebp), %eax
    movl %ecx, 0(%eax)
    movl 8(%ebp), %eax
    movl %ebp, %esp
    popl %ebp
    ret
Example 3: Interpreters
Consequence 2: Malware
Consequence 2: Malware

• Why does Java do array bounds checking?
• Unsafe language like C and C++ don’t do that checking;
  – They will happily let you write a program that “writes past”
    the end of an array.

• Result:
  – viruses, worms, “jailbreaking”
    mobile phones, Spam, botnets, ...

• Fundamental issue:
  – Code is data.
  – Why?
Consider this C Program

```c
void m() {
    char[10] buffer;

    char c = read();
    int i = 0;
    while (c != -1) {
        buffer[i] = c;
        c = read();
        i++;
    }
    process(buffer);
}

void main() {
    m();
    // do some more stuff
}
```

Notes:
- C doesn’t check array bounds
- Unlike Java, it stores arrays directly on the stack
- What could possibly go wrong?
Abstract Stack Machine

“Stack Smashing Attack”
void registerStudent() {
    print("Welcome to student registration.");
    print("Please enter your name:");
    String name = readLine();
    evalSQL("insert into Students(‘’ + name + ‘’)’");
}
Consequence 3: Undecidability
Undecidability Theorem

It is impossible to write a method

```java
boolean halts(String prog)
```

such that for any valid Java program \( P \) represented as a string \( p_P \)

```java
halts(p_P)
```

returns true exactly when the program \( P \) halts on all inputs, and false otherwise.
Suppose we could write such a program:

```java
class HaltDetector {
    public static boolean halts(String javaProgram) {
        // ...do something wonderful...
        // return true if javaProgram halts on all inputs
        // return false if javaProgram does not
    }
}
```

- `HaltDetector.halts(p) ⇒ true` means “p halts on all inputs”
- `HaltDetector.halts(p) ⇒ false` means “p does not halt on some input”
- `HaltDetector.halts` never raises an exception or loops
Do these methods always halt?

“boolean m(String x){ return false; }”
=> YES

“boolean m(String x){ return m(x); }”
=> NO

“boolean m(String x){
    if (x.length() == 3) return true;
    else return false;
}”
=> YES

“boolean m(String x){
    if (x.length() == 3) return m(x);
    else return false;
}”
=> NO

“boolean m(String x){
    if (x.length() == 3) return m(x + \"a\");
    else return false;
}”
=> YES

CIS120 / Spring 2013
Consider this Program called Q:

class HaltDetector {
    public static boolean halts(String javaProgram) {
        // ...do something wonderful...
        // return true if javaProgram halts on all inputs
        // return false if javaProgram does not
    }
}

class Q {
    String p_Q = ??? // string for program Q itself

    public static void main(String[] args) {
        if (HaltDetector.halts(p_Q)) {
            while (true) { /* infinite loop! */ }
        }
    }
}
What happens when we run it?

HaltDetector.halts(p_Q) ⇒ true then Q ⇒ infinite loop
HaltDetector.halts(p_Q) ⇒ false then Q ⇒ always halts

Contradiction!

- Russell’s Paradox (1901)
- Gödel’s Incompleteness Theorem (1931)
- Both rely on self reference.
Potential Hole in the Proof

- What about the ??? in the program Q?
- It is supposed to be a String representing the program Q itself.
- How can that be possible?
- Answer: code is data!

- See Quine.java
Profound Consequences

• Halts is *undecidable*
  – *There are problems that cannot be solved by a computer program!*

• Rice’s Theorem:
  – Any “sufficiently interesting” property about computer programs is undecidable!

• You can’t write a perfect virus detector!
  – Three possibilities:
  – (1) virus detector might go into an infinite loop
  – (2) it gives you false positives (i.e. says something is a virus when it isn’t)
  – (3) it gives you false negatives (i.e. it says a program is not a virus when it is)

• Corollary: You can’t write a perfect autograder!
Recommended Reading

• Logicomix: An Epic Search for Truth. Apostolos Doxiadis and Christos Papadimitriou
• Secure Coding in C and C++. Robert C. Seacord
• Goedel, Escher, Bach: An eternal golden braid. Douglas Hofstadter
• I am a Strange Loop. Douglas Hofstadter