Example projects

UW CSE 403
March 30, 2016
Michael Ernst
Outline

• Concurrency: name protection vs. value protection
• Stack Overflow parsing
• Minimizing bug fixes
• Prevent index-out-of-bounds errors
• Purity or side effect analysis
• Generating tests from documentation
Concurrency: name protection vs. value protection

Thread 1
balance = balance + deposit;

Thread 2
balance = balance + deposit;

Suppose:
• I start with a balance of $100
• I deposit $50 at ATM 1
• My spouse deposits $25 at ATM 2
What is the final value of balance?
Solution: locking

Object myLock;
@GuardedBy("myLock") int balance;

// legal
synchronized(myLock) {
    balance = balance + deposit;
}

// illegal
balance = balance + deposit;
Standard semantics of @GuardedBy is unsound: protects names, not values

Object myLock;
@GuardedBy("myLock") List<String> words;

// legal
synchronized(myLock) {
    words.add("hello");
}

// illegal
words.add("hello");

// Permitted by name protection!
List<String> otherList;
synchronized(myLock) {
    otherList = words;  // OK, because myLock is currently held
}
otherList.add("hello");  // PROBLEM: may occur in parallel with other operations
Object myLock;
@GuardedBy("myLock") List<String> words;

@GuardedBy("myLock") List<String> otherList;
synchronized(myLock){
    otherList = words; // OK!
}
otherList.add("hello"); // forbidden
synchronized(myLock) {
    otherList.add("hello"); // OK
}
A value-protection implementation exists

• Evaluate it

• Current practice
  • What do programmers think @GuardedBy means?
  • Do programmers use it as documented?
  • Do programs have latent concurrency bugs?

• Usability
  • Is the value-protection semantics easy to understand and use?
  • Use it on some real programs, make suggestions and enhancements
Stack Overflow parsing

• Stack Overflow helps programmers
• How can it help tools?

Examples:
• Summarize source code
• Autocomplete or code snippet suggestions
• Code generation from English text

Problem: naïve use of Stack Overflow
• Text = title of the question
• Code = first code snippet in the accepted answer
Example Stack Overflow question and answer

“How can I merge two Python dictionaries?”
Stack Overflow parsing

Problems with standard techniques:

• Question titles are often short or non-descriptive
• Text in the answer often serves an important explanatory purpose
• Answers often have multiple code snippets.
  • It may be necessary to concatenate two snippets in order to achieve a particular goal.
  • An answer may give two different ways to solve a problem, in which case the two snippets should *not* be merged.

Goal: better parsing, or at least segmentation into distinct parts
Evaluation: re-run previous experiments; improvements?
Minimizing bug fixes

diff --git a/java/src/plume/MathMDE.java.jpp
b/java/src/plume/MathMDE.java.jpp
index b6dcf96..cbcaf9c 100644
--- a/java/src/plume/MathMDE.java.jpp
+++ b/java/src/plume/MathMDE.java.jpp
@@ -353,19 +353,19 @@ public final class MathMDE {
    return pow_fast(base, expt);
 }

- private static int pow_fast(int base, int expt) throws ArithmeticException {
-     if (expt < 0) {
-        throw new ArithmeticException("Negative exponent passed to pow");
-     }
-     int this_square_pow = base;
-     int result = 1;
-     while (expt>0) {
-         if ((expt & 1) != 0) {
-             result *= this_square_pow;
-         }
-         expt >>= 1;
-         this_square_pow *= this_square_pow;
-     }
-     return result;
- }

+ private static int pow_fast(int base, int expt) throws ArithmeticException {
+     if (expt < 0) {
+         throw new ArithmeticException("Negative exponent passed to pow");
+     }
+     int this_square_pow = base;
+     int result = 1;
+     while (expt>0) {
+         if ((expt & 1) != 0) {
+             result *= this_square_pow;
+         }
+         expt >>= 1;
+         this_square_pow *= this_square_pow;
+     }
+     return result;
+ }
Minimizing bug fixes

Every commit should have a single purpose:
• add a feature
• fix a bug
• refactor

In practice each commit mixes multiple distinct changes
• harder for programmers and tools to interpret.

Goal: minimize a patch
• Example: find the smallest part of the patch that fixes the bug
• Leave out documentation changes, variable renaming, refactorings, ...
Prevent index-out-of-bounds errors

int i = -1;
... a[i] ...  // run-time error

int j = myList.size();
... myList.get(j) ...  // run-time error

It’s better to prevent an error at compile time than to have a user discover it at run time
Compile-time checking via type systems

The Java compiler already gives warnings about certain types of errors:

```java
String s = "hello";
... a[s] ... // compile-time error
```

Goal: compiler also warns about index-out-of-bounds errors

CSE 331 showed how to prove that all array/list dereferences are within bounds
• It was a manual, tedious process
Idea: extend Java’s type system

• If program type-checks, every index is within bounds
• Tool called the Checker Framework ([http://CheckerFramework.org/](http://CheckerFramework.org/)) makes it easy to write a type system
• User can write annotations within Java 8 syntax

Evaluate with case studies
Purity or side effect analysis

if (this.myField != null) {
    int x = this.computeValue();
    ... this.myField.toString() ...  // can this line suffer a null pointer exception?
}

It can! The reason is that the computeValue method might set myField to null.

A "pure" procedure:
• performs no visible side effects, and
• returns the same value when it is called twice on the same values.

There are many other uses for purity besides this analysis
Purity implementation and evaluation

• Idea: re-implement the analysis in Salcianu and Rinard's paper "Purity and Side Effect Analysis for Java Programs"
  • jppa tool was widely used, but has not been maintained
  • Java tools are much better now, so re-implementation should be straightforward

• Evaluation:
  • Against other tools for purity analysis
  • Plugged into downstream tools (nullness analysis, test generation, etc.)
  • May find ways to improve the purity analysis, too.
/**
 * Checks to see whether the comparator
 * is now locked against further changes.
 * @throws UnsupportedOperationException
 * if the comparator is locked
 */
protected void checkLocked() {...}

void test() {
  FixedOrderComparator c = new FixedOrderComparator(...);
  ...
  c.compare(...);
  ...
  if (c.isLocked()) {
    try {
      c.checkLocked();
      fail();
    } catch (UnsupportedOperationException e) {
      // Expected Exception!
    }
  } else {
    c.checkLocked();
  }
}
Goal: generate tests from English documentation

- Parse descriptions such as "throws NullPointerException if any element of the array is null"
- Assume that the programmer already has test inputs
  - the only question is whether the code's behavior is correct

A prototype tool exists

Challenges:
- Better natural language processing and pattern-matching to recognize documentation that programmers write
- Evaluating the tool: given English documentation and the tool's output, are its assertions correct and sufficient?
  - Idea: Pay programmers to produce goal files, via a crowdsourcing platform.
  - Experimental design: can you trust the programmers?
More sources of ideas

Mike’s ideas:

- [http://homes.cs.washington.edu/~mernst/uw-only/research/potential-research-projects.html](http://homes.cs.washington.edu/~mernst/uw-only/research/potential-research-projects.html)
- [https://raw.githubusercontent.com/codespecs/daikon/master/doc/to do.txt](https://raw.githubusercontent.com/codespecs/daikon/master/doc/to do.txt)

A better source of ideas:

- Your experience, and your frustrations when developing software