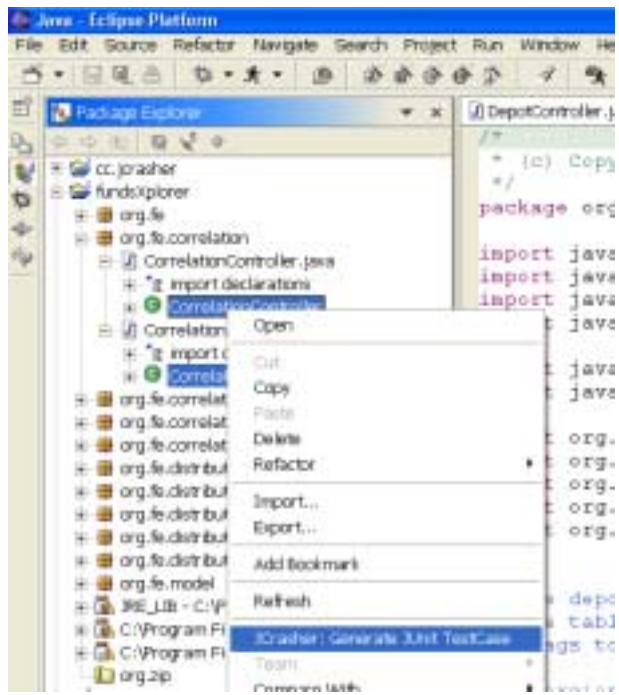
CS 7001, Mo. 17. November 2003

Mini Project Three: Exception Grouping for JCrasher— an Automatic Robustness Tester for Java

Christoph Csallner
csallner@cc.gatech.edu

Yannis Smaragdakis
yannis@cc.gatech.edu



Problem of Manually Searching through Many Exception Reports

- Assume you run 100,000 random test-cases
- You get 10,000 problem reports
- Many of them may be equivalent, redundant
- Automatic aggregation based on some clustering algorithm would help
  - Should be easy to implement

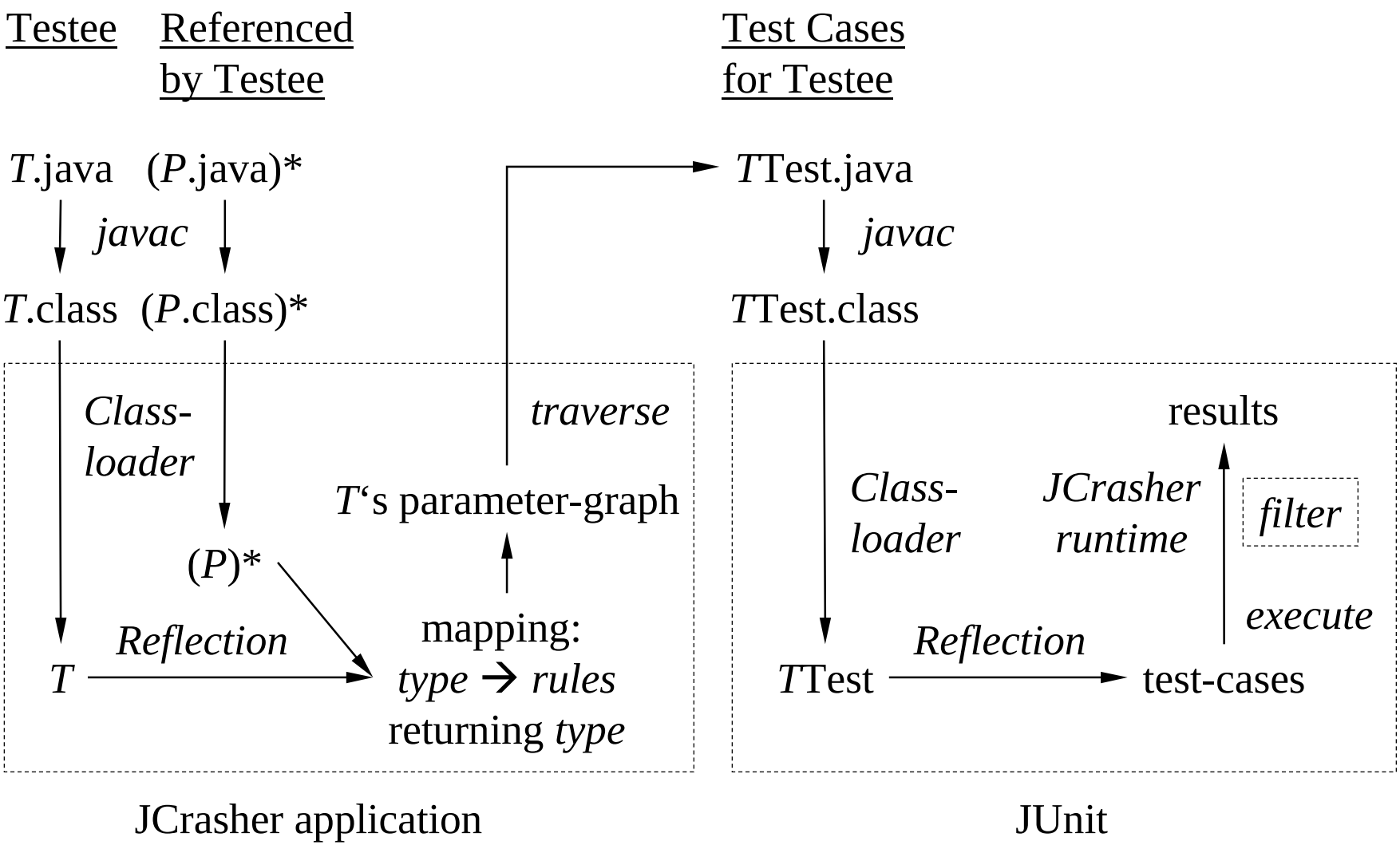
Why is this an interesting problem?

- *Robustness* is the ability of a program unit to handle gracefully at run time any inputs that are statically permissible by the unit's interface.
- Robustness is important for many programs as there may be no way to statically ensure that all functionality is called in a way that respects the necessary preconditions.
  - For example a future program extension might call the program with some unexpected parameters. Then a non-robust program would terminate abnormally or crash as it cannot handle these parameters.

Robustness Testing

- *Robustness Testing* seeks such unexpected parameters that the program cannot handle
- Analyze the program under test.
- Provide well-formed but random data as inputs.
  - Many different parameter combinations possible, e.g. m(int, int) has $2^{64} = 2^{8*4} * 2^{8*4}$ parameter combinations
  - Produces many test-cases, some of them are redundant
- Check—typically with limited human help—whether the results are correct.

JCrasher: An Automatic Robustness Tester for Java



Solution

- Modified JUnit
- Check for each exception, whether a similar one has occurred before
  - Similar = same type and same stack-trace
- Also useful for regular testing
- Alternative Approach
  - Use static analysis to only generate non-similar test cases

Demo

- Testee `P1` from freshmen programming course homework:

```
public static int[] getSquaresArray(int length) {
    int[] emptyArray = new int [length];  // [..]
```

  - `NegativeArraySizeException` iff `length == -1`
- JCrasher automatically generates 95 test cases: `P1Test*`
  - Calling testee's methods with different parameter combinations
- JUnit reports 22 exceptions or errors
- Our Grouping-JUnit reports two exceptions or errors
  - Suppresses eight duplicate reports of `NegativeArraySizeException` caused by `P1.getSquaresArray`

Conclusions and Future Work

- Easy way to suppress duplicate exception reports
- Using exception grouping to evaluate JCrasher's bug-finding effectiveness
- Will add a tree-based visualization to JUnit GUI
  - Suppressed exceptions shown on demand

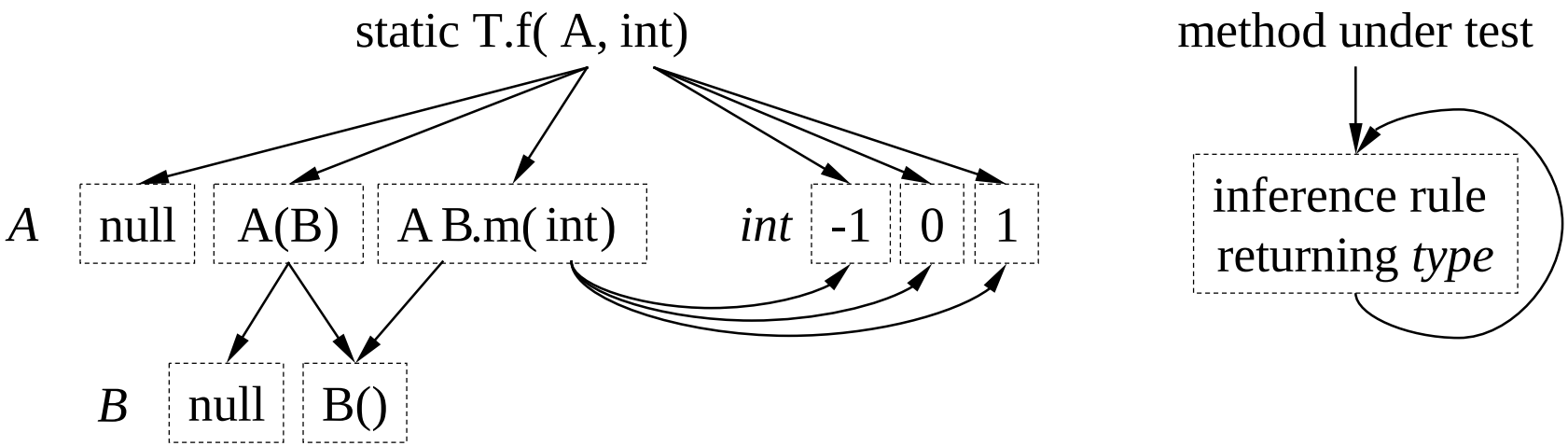
Backup

Collect Type Inference Rules

- Search class under test for inference rules
- Transitively search referenced types
- Inference rules
  - Method $T.m(P_1, P_2, ..., P_n)$ returns X:
    $X \leftarrow T, P_1, P_2, ..., P_n$
  - Sub-type Y {extends | implements} X:
    $X \leftarrow Y$
  - Constructors and preset values are implicitly known
- Add each discovered inference rule to mapping:
  $X \rightarrow$ inference rules returning X

Generate Test Cases For a Method

Parameter Graph for Method T.f(A, int)



Test Cases:

f(null, -1), f(null, 0), f(null, 1),
f(A(null), -1), ...,

Test Case Execution and Exception Filtering

- JCrasher generated test cases look like:

```
public void test1() throws Throwable {
   try { /* test case */ }
   catch (Exception e) {
       dispatchException(e);   /* JCrasher runtime */
   }
}
```

- An exception indicates one of the following
  - As a part of the method's contract, the method under test signals a violated precondition—no bug.
  - The method under test has run into an unforeseen problem and is terminated—bug.