EVALUATING CODE-BASED TEST INPUT GENERATOR TOOLS

Presented by Zoltan Micskei
# Code-based test generation

```c
int fun1(int a, int b){
    if (a == 0){
        printf(ERROR_MSG);
        return -1;
    }
    if (b > a)
        return b*a + 5;
    else
        return (a+b) / 2;
}
```

<table>
<thead>
<tr>
<th>#</th>
<th>a</th>
<th>b</th>
<th>stm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>*</td>
<td>1, 2</td>
</tr>
<tr>
<td>2</td>
<td>a!=0</td>
<td>b &gt; a</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>a!=0</td>
<td>b &lt; a</td>
<td>4</td>
</tr>
</tbody>
</table>
Code-based test generator tools

Detailed list of tools: [http://mit.bme.hu/~micskeiz/pages/cbtg.html](http://mit.bme.hu/~micskeiz/pages/cbtg.html)
Motivation

How can the different test input generator tools be compared and evaluated?
APPROACH
Core features

- Types, Operators
- Conditionals, Loops
- Arrays, Functions

Basic

- Structure usage
- Nested structures
- ...

Structures

- Object usage
- Inheritance, interfaces
- ...

Objects

- Generic functions
- Generic objects
- ...

Generics

- Arithmetic, Strings
- Collections
- ...

Library
Extra features

- Stdin, Properties
- Files, Sockets

- Threads, Locks
- Indeterminacy

- Classes, Methods

- Native functions

Environment
Multi-threading
Reflection
Native code
Snippets

```java
public static int guessObject(SimpleObject obj) {
    if (obj == null) {
        return -1;
    }

    if (obj.getOperationCount() == 2 && obj.getResult() == 3) {
        return 1;
    } else {
        return 0;
    }
}
```

300 core snippets
63 extra snippets
Manual sample inputs
Execution (SETTE framework)

1. Code snippets
2. Generate Runner Project for a Tool
   - Tool-specific Files (e.g. test-drivers)
3. Execute Code Snippets on a Tool
   - Raw Results
4. Parse Results
   - Results in XML
5. Generate Test Suite
   - JUnit Test Cases
6. Analyse Coverage
   - Evaluation for a Tool

Working Phase

Artifact
RESULTS
## Experiments

<table>
<thead>
<tr>
<th>Tools</th>
<th>Status</th>
<th>Measures</th>
<th>Setup</th>
</tr>
</thead>
<tbody>
<tr>
<td>CATG</td>
<td>N/A</td>
<td>Coverage, Size, Duration, Mutation score</td>
<td>30s limit, Repeat 10x, Variable time</td>
</tr>
<tr>
<td>EvoSuite</td>
<td>EX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IntelliTest</td>
<td>T/M</td>
<td></td>
<td></td>
</tr>
<tr>
<td>jPET</td>
<td>NC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPF</td>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Randoop</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overview of capabilities (Status)
Summary of insights

- **Duration**: 20 min – 200 min
  - Randoop, EvoSuite use all available time
- **Size**:
  - 700 (manual) < 1000 (IntelliTest) << 270 000 (Randoop)
- **Extra snippets**
  - EvoSuite: custom sandbox
  - Otherwise significant challenge
Summary

All code and results are available: http://sette-testing.github.io