Automated evaluation of students’ programs:
Testing, Verification and Similarity

Milena Vujošević-Janičić, Mladen Nikolić and Dušan Tošić

— early stage work —
Fifth Workshop on Formal and Automated Theorem Proving and Applications
Belgrade,
February 3-4, 2012.
Agenda

- Motivation
- Testing
- Verification
- Program Similarity
- Conclusions and Further work
Agenda

• Motivation

• Testing

• Verification

• Program Similarity

• Conclusions and Further work
Motivation

- An automated quality evaluation tool

- Benefits for students: evaluation and guidance in absence of a teacher

- Benefits for teachers: automated marking of exams and error detection
Motivation

- **Starting point:** problem & teacher’s solution

- **Input:** student’s solution

- **Output:** evaluation of student’s solution
Motivation

• The approach integrates three features:
  
  – **Testing** — functional correctness
  
  – **Verification** — buffer overflows, null pointer dereferencing, division by zero ...
  
  – **Similarity** — modularity and structural simplicity
Agenda

• Motivation

• Testing

• Verification

• Program Similarity

• Conclusions and Further work
Testing

- Successful testing indicates functional correctness
- Test cases — given by a teacher or automatically generated
- Problems with comparing outputs
- Definition of a problem — precise and accurate
Agenda

- Motivation
- Testing
- Verification
- Program Similarity
- Conclusions and Further work
Verification

• LAV* is a bug-finding tool, it is open source

• LAV combines symbolic execution, SAT encoding of program’s behavior and bounded model checking

• LAV generates correctness conditions that are passed to a suitable SMT solver

• More details on LAV can be found in our VSTTE’12 paper or at http://argom.matf.bg.ac.rs/?content=lav

*Joint work with Viktor Kuncak, EPFL and Filip Maric

LLVM based Automated Verifier
Verification: Experiments

• 157 programs written by students at exams during an introductory course in programming analyzed

<table>
<thead>
<tr>
<th>Problem</th>
<th># Solutions</th>
<th>Avg. Lines</th>
<th>Avg. Reported Bugs</th>
<th>Avg. False Alarms</th>
</tr>
</thead>
<tbody>
<tr>
<td>calculations</td>
<td>60</td>
<td>30</td>
<td>0.82</td>
<td>0.05</td>
</tr>
<tr>
<td>arrays and matrices</td>
<td>71</td>
<td>46</td>
<td>4.20</td>
<td>0</td>
</tr>
<tr>
<td>strings and structures</td>
<td>26</td>
<td>60</td>
<td>2.92</td>
<td>1.11</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td><strong>157</strong></td>
<td><strong>42</strong></td>
<td><strong>2.69</strong></td>
<td><strong>0.20</strong></td>
</tr>
</tbody>
</table>
**Verification: Analysis of Results**

<table>
<thead>
<tr>
<th></th>
<th>calculations &amp; arrays and matrices</th>
<th>strings and structures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most frequent bug</td>
<td>buffer overflow</td>
<td>null pointer dereferencing</td>
</tr>
<tr>
<td># programs with the above bug</td>
<td>81</td>
<td>15</td>
</tr>
<tr>
<td># bugs</td>
<td>225</td>
<td>46</td>
</tr>
<tr>
<td>Second most frequent bug</td>
<td>division by zero</td>
<td>buffer overflow</td>
</tr>
<tr>
<td># programs with the above bug</td>
<td>22</td>
<td>15</td>
</tr>
<tr>
<td># bugs</td>
<td>22</td>
<td>30</td>
</tr>
</tbody>
</table>
Verification: Analysis of Results

- The vast majority of bugs due to wrong expectations e.g., that input parameters of programs will meet certain constraints

- This explains the large number of bugs in the corpus — adding only one check in a program would typically eliminate several bugs

- LAV could help students to remember to put these checks
Agenda

- Motivation
- Testing
- Verification
- Program Similarity
- Conclusions and Further work
Program Similarity

• Testing and verification — functional correctness and bugs

• Modularity

• Structural simplicity
## Program Similarity

<table>
<thead>
<tr>
<th></th>
<th>If Statement</th>
<th>Loop</th>
<th>Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>if(a&lt;b) n = a;</code></td>
<td><code>for(i=0; i&lt;n; i++)</code></td>
<td><code>n = min(a, b);</code></td>
</tr>
<tr>
<td></td>
<td><code>else n = b;</code></td>
<td><code>for(j=0; j&lt;n; j++)</code></td>
<td><code>m = min(c, d);</code></td>
</tr>
<tr>
<td></td>
<td><code>if(c&lt;d) m = c;</code></td>
<td><code>if(i==j)</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>else m = d;</code></td>
<td><code>m[i][j] = 1;</code></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><code>for(i=0; i&lt;strlen(s); i++)</code></td>
<td><code>for(i=0; s[i]; i++)</code></td>
<td></td>
</tr>
</tbody>
</table>
Program Similarity

- Control flow graph represents the structure of a program

- Program similarity — similarity of CFGs

- CFG similarity measure should reflect intuitive similarity of programs

- CFG similarities are computed as described in (Mladen Nikolic, 2013).

- First experimental results are encouraging
Agenda

• Motivation

• Testing

• Verification

• Program Similarity

• Conclusions and Further work
Conclusions and Further work

- What we have:
  - Some experience in automated testing
  - Software verification tool LAV
  - Program similarity measure

- What we need to do:
  - Define a framework for testing
  - Elimination of false alarms
  - Improvement of program similarity measure
  - Integration of all three parts into a web tool
Thank you

Verification: One Simplified Student's Code

```
#include<stdio.h>
#include<stdlib.h>
int power(int n)
{
    int i, pow;
    for(i=0, pow=1; i<n; i++, pow*=10);
    return pow;
}
int get_digit(int n, int d)
{
    return (n/power(d))%10;
}
int main(int argc, char** argv)
{
    int n, d;
    n = atoi(argv[1]);
    d = atoi(argv[2]);
    printf("%d\n", get_digit(n, d));
}
```

- Line 12: SAFE
- Line 18: SAFE
- Line 19: SAFE
- Line 20: 12: SAFE

Function: get_digit
- Error: division_by_zero
- Line 12: d == 1073741824,

Function: main
- Error: buffer_overflow
- Line 18: argc == 1, argv == 1

Function: main
- Error: buffer_overflow
- Line 19: argc == 2, argv == 1

Function: main
- Error: division_by_zero
- Line 20: 12: argc == 512, argv == 1, d == 1073741824, n == 0