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://argo.matf.bg.ac.rs/?content=lav

*Joint work with Viktor Kuncak, EPFL and Filip Maric
*L*LVM based *A*utomated *V*erifier

Verification: Experiments

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

Problem	# Solutions	Avg. Lines	Avg. Reported Bugs	Avg. False Alarms
calculations	60	30	0.82	0.05
arrays and matrices	71	46	4.20	0
strings and structures	26	60	2.92	1.11
Summary	157	42	2.69	0.20

Verification: Analysis of Results

	calculations & arrays and matrices	strings and structures
Most frequent bug	buffer overflow	null pointer dereferencing
# programs with the above bug # bugs	81 225	15 46
Second most frequent bug	devision by zero	buffer overflow
# programs with the abouve bug # bugs	22 22	15 30

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

1.	`if(a<b) n = a;`
	`else n = b;`
	`if(c<d) m = c;`
	`else m = d;`
		`n = min(a, b);`
		
		`m = min(c, d);`

2.	`for(i=0; i<n; i++)`
	`      for(j=0; j<n; j++)`
	`            if(i==j)`
	`                  m[i][j] = 1;`
		`for(i=0; i<n; i++)`
		`      m[i][i] = 1;`

3.	`for(i=0; i<strlen(s); i++)`	`for(i=0; s[i]; i++)`

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

Bibliography

Milena Vujosevic Janicic, Viktor Kuncak, 2012 — Development and Evaluation of LAV: An SMT-Based Error Finding Platform. *Verified Software, Theories, Tools, and Experiments* 2012:98-113

Mladen Nikolic, 2013 — Measuring Similarity of Graph Nodes by Neighbor Matching, *Intelligent Data Analysis*, 2013.

Verification: One Simplified Student's Code

```
1:  #include<stdio.h>
2:  #include<stdlib.h>
3:  int power(int n)
4:  {
5:  int i, pow;
6:  for(i=0, pow=1; i<n; i++, pow*=10);
7:  return pow;
8:  }
9:
10: int get_digit(int n, int d)
11: {
12: return (n/power(d))%10;
13: }
14:
15: int main(int argc, char** argv)
16: {
17: int n, d;
18: n = atoi(argv[1]);
19: d = atoi(argv[2]);
20: printf("%d\n", get_digit(n, d));
21: }
```

```
line 12: UNSAFE
line 18: UNSAFE
line 19: UNSAFE
line 20: 12: UNSAFE

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
```