System LAV and Its Applications

Milena Vujošević-Janičić

Progress in Decision Procedures: From Formalizations to Applications

Belgrade,
March 30, 2013.
Overview

**Milena Vujošević-Janičić, Viktor Kuncak**


**Milena Vujošević-Janičić, Mladen Nikolić, Dušan Tošić, Viktor Kuncak**

People

prof. Dušan Tošić
Faculty of Mathematics, Belgrade

prof. Viktor Kuncak
EPFL Switzerland

Mladen Nikolić
Faculty of Mathematics, Belgrade
LAV is a tool for statically verifying program assertions and locating bugs such as buffer overflows, pointer errors and division by zero.

- LAV is publicly available and open-source
  http://argo.matf.bg.ac.rs/?content=lav

- Experimental evaluation shows that LAV is competitive with related tools based on symbolic execution and bounded model checking (KLEE, CBMC and ESBMC)
Application of LAV

Discussion

Input code

LLVM
Code transformation
input code
Symbolic execution
of blocks
loop-free code
Relationships between blocks
Correctness conditions
next function analysis

Output
LAV is primarily developed for programs in C.
Thanks to LLVM, LAV can be used for other procedural languages.
Input program is transformed to LLVM code.
LLVM is a rich compiler framework
Each LLVM function consists of blocks of instructions which are interconnected
LLVM uses simple RISC-like instructions
LLVM code is an input to LAV
LAV unrolls loops to get loop-free program
LAV can unroll loops in two ways, by using under-approximation or over-approximation
LAV symbolically executes instructions of each block.

For safety critical commands, LAV generates safety conditions.

LAV checks these conditions in different contexts.

LAV communicates with an SMT solver through an SMT interface.

- LAV symbolically executes instructions of each block.
- For safety critical commands, LAV generates safety conditions.
- LAV checks these conditions in different contexts.
- LAV communicates with an SMT solver through an SMT interface.
LAV Application of LAV Discussion

- Code transformation
  - Input code
  - LLVM
  - Loop-free code
- Symbolic execution of blocks
  - Condition
  - SMT interface
  - Output
LAV Application of LAV

Discussion

Input code

LLVM

Code transformation

LLVM

Input code

Boolector bv, arr

Symbolic execution of blocks

loop-free code

SMT interface

condition

Correctness conditions

next function analysis

Relationships between blocks

Milena Vujošević-Janičić

System LAV and Its Applications
LAV

Application of LAV
Discussion

- Code transformation
- LLVM
- Input code
- Symbolic execution of blocks
- loop-free code
- condition
- SMT interface
- Boolector bv,arr
- MathSAT la,bv,euf
- Relationships between blocks
- Correctness conditions
- Output

Milena Vujošević-Jančić
System LAV and Its Applications
LAV

Application of LAV

Discussion

Input code

LLVM

Code transformation

loop-free code

Symbolic execution of blocks

condition

SMT interface

Boolector

bv, arr

MathSAT

la, bv, euf

Yices

la, bv, euf

Relationships between blocks

Correctness conditions

Output

Milena Vujošević-Janičić

System LAV and Its Applications
LAV Application of LAV Discussion

LAV

Code transformation

LLVM

Input code

Boolector bv,arr

MathSAT la,bv,euf

Yices la,bv,euf

Z3 la, bv,euf,arr

Symbolic execution of blocks

loop-free code

condition

SMT interface

Relationships between blocks

Correctness conditions

next function analysis

Output

Milena Vujošević-Janičić System LAV and Its Applications
LAV

Application of LAV
Discussion

Code transformation

Input code

LLVM

Symbolic execution of blocks

loop-free code

SMT interface

sat/unsat

Boolector

MathSAT

Yices

Z3

la, bv, euf, arr

Input code

la, bv, euf

la, bv, euf

la, bv, euf, arr

Relationships between blocks

Correctness conditions

next function

analysis

Output
LAV

**Application of LAV**

**Discussion**

**Input code**

**LLVM**

**Code transformation**

**loop-free code**

**Symbolic execution of blocks**

**SMT interface**

**Boolector**

**bv,arr**

**MathSAT**

**la,bv,euf**

**Yices**

**la,bv,euf**

**Z3**

**la, bv,euf,arr**

**Relationships between blocks**

**Correctness conditions**

**next function analysis**

**Output**

Milena Vujošević-Janičić

System LAV and Its Applications
LAV

- Code transformation
- Loop-free code
- Symbolic execution of blocks

LLVM

Input code

- Boollector
  - bv, arr
- MathSAT
  - la, bv, euf
- Yices
  - la, bv, euf
- Z3
  - la, bv, euf, arr

Relationships between blocks

Correctness conditions

Output analysis

Milena Vujošević-Janičić

System LAV and Its Applications
LAV

Application of LAV
Discussion

Code transformation

LLVM

Input code

Boolector

bv, arr

MathSAT

la, bv, euf

Yices

la, bv, euf

Z3

la, bv, euf, arr

Symbolic execution of blocks

loop-free code

SMT interface

Relationships between blocks

Correctness conditions

Output

next function analysis

System LAV and Its Applications

Milena Vujošević-Janičić
**LAV**

- **Code transformation** → **LLVM** → **Input code**
- **loop-free code**

- **Symbolic execution of blocks**

- **Relationships between blocks**

- **Correctness conditions**

  - For each block LAV constructs its postcondition built from:
    - control flow information
    - transformation done by symbolic execution
For each unresolved safety critical command, LAV considers the function context.

At this stage, LAV uses SMT solver in an incremental manner.

A command gets its status: safe, unsafe, flawed, unreachable.
LAV

Application of LAV

Discussion

LAV

Input code

LLVM

Code transformation

loop-free code

Symbolic execution of blocks

Relationships between blocks

Correctness conditions

next function analysis

Milena Vujošević-Jančić
System LAV and Its Applications
LAV

Application of LAV
Discussion

Input code

LLVM

Code transformation
loop-free code

Symbolic execution of blocks

Relationships between blocks

Correctness conditions

SMT interface

Boolector
bv, arr

MathSAT
la, bv, euf

Yices
la, bv, euf

Z3
la, bv, euf, arr

Output

next function analysis

System LAV and Its Applications

Milena Vujošević-Janičić
LAV

Application of LAV

Discussion

LAV

Code transformation

Loop-free code

Symbolic execution of blocks

Relationships between blocks

Correctness conditions

LLVM

Input code

Boolector

MathSAT

Yices

Z3

SMT interface

Output

Input code

Boolector

MathSAT

Yices

Z3

Relationships between blocks

Correctness conditions

Symbolic execution of blocks

Loop-free code

Code transformation

LLVM

Output

SMT interface

MathSAT

Yices

Z3

Boolector

La, bv, euf

La, bv, euf

La, bv, euf

La, bv, arr

System LAV and Its Applications
Output format

For a command that is not safe, LAV outputs the line number, the kind of the error, a program trace that introduces the error, and values of variables along this trace.

```c
#include <stdio.h>
#define MAX 10
int main()
{
  int arr[MAX], d, n, n_copy, max;
  get_array(arr, MAX);
  scanf("%d", &n);
  n_copy = n;
  max = n%10;
  while(n){
    d = n%10;
    if(max<d)
      max=d;
    n = n/10;
  }
  if(arr[max]>n_copy)
    printf("it is bigger\n");
  else printf("it is not bigger\n");
}
```
LAV in Education

- Verification tools are usually used for safety critical programs
- New domain: Education
- LAV analyzed corpus of 157 students’ programs (tests written at introductory programming course)
- LAV found 423 errors!
Verification tools are usually used for safety critical programs
New domain: Education
LAV analyzed corpus of 157 students’ programs (tests written at introductory programming course)
LAV found 423 errors!

Verification in education
Can a verification tool help in automated evaluation of students’ programs?
LAV in Education

- Verification tools are usually used for safety critical programs
- New domain: Education
- LAV analyzed corpus of 157 students’ programs (tests written at introductory programming course)
- LAV found 423 errors!

**Verification in education**

Can a verification tool help in automated evaluation of students’ programs?
Automated evaluation

- Beneficial for both teachers (automated grading) and students (immediate feedback)
  - Can a verification tool help teachers in grading assignments?
  - Can a verification tool help students to learn programming?

- Important for introductory programming courses – the biggest number of students

- With a growing number of on-line courses, this automation becomes even more significant
Automated evaluation

- Beneficial for both teachers (automated grading) and students (immediate feedback)
  - Can a verification tool help teachers in grading assignments?
  - Can a verification tool help students to learn programming?
- Important for introductory programming courses – the biggest number of students
- With a growing number of on-line courses, this automation becomes even more significant
Testing vs. verification

- Automated evaluation based on automated testing — does the program give the desired outputs for specific inputs?
- Standard problem: Testing cannot reveal all bugs
- Are there bugs that can be discovered by a verification tool in a program that successfully passed testing?
- 266 programs (solutions of 15 different problems) successfully passed testing
Testing vs. verification

- Automated evaluation based on automated testing — does the program give the desired outputs for specific inputs?
- Standard problem: Testing cannot reveal all bugs
- Are there bugs that can be discovered by a verification tool in a program that successfully passed testing?
- 266 programs (solutions of 15 different problems) successfully passed testing — LAV found 35 errors
Automated evaluation based on automated testing — does the program give the desired outputs for specific inputs?

Standard problem: Testing cannot reveal all bugs

Are there bugs that can be discovered by a verification tool in a program that successfully passed testing?

266 programs (solutions of 15 different problems) successfully passed testing — LAV found 35 errors
Testing vs. verification

- Randomly generated test-cases (fuzzing) discovered only 12 of these bugs and took significantly more time.
- Important issue (concerning feedback to students):
  - If a program crashes, testing cannot give an explanation why it crashed.
  - Verification tools can give explanations.
Similarity measure

- Functional correctness is not the only important aspect in automated evaluation.
- Requirements concerning the design of a solution cannot be addressed by testing and verification.
- To address the design, we used similarity measure between CFGs of expected solution and student’s solution:

  *Mladen Nikolić, Measuring Similarity of Graph Nodes by Neighbor Matching, Intelligent Data Analysis, (2013).*

- Open-source, publicly available from LAV’s page.
Similarity measure

- Functional correctness is not the only important aspect in automated evaluation
- Requirements concerning the design of a solution cannot be addressed by testing and verification
- To address the design, we used similarity measure between CFGs of expected solution and student’s solution:


- Open-source, publicly available from LAV’s page
Predicting a grade

Linear model based on testing ($t$), verification ($v$) and CFG similarity ($s$)

$$\text{grade} = c_1 \cdot t + c_2 \cdot v + c_3 \cdot s$$

- Coefficients $c_i$ are determined based on a set of instructor graded solutions
- The model obtained high correlation between predicted and instructor provided grades (84%)
- Better results compared to the model without verification component and to the model without similarity component
- All obtained results statistically significant
Conclusions

LAV

LAV is an SMT-Based error finding platform
  - Uses symbolic execution and elements of BMC
  - Competitive to related tools

Verification in education

Verification can add new quality to automated evaluation of students’ assignments:
  - Feedback for students
  - Automated grading for teachers
Future work

LAV
Further improvements of LAV

Application of LAV
Integrating automated evaluation into a web-based system
Thank you for your attention!