Program Verification: 
Next Steps For Usability

Usable Verification 2010
Peter H. Schmitt and Mattias Ulbrich | November 16, 2010
Overview

Who we are

- Deductive Java source code verification
- Dynamic logic, symbolic execution
- JML
- Combine interaction and automation

What we did

1. Our view of the field
2. next steps for more usable verification systems
Program verification

- extended debugging
- increase confidence
- full functional verification

Target
Program verification

- SBMC
- Abs. Int.
- SLAM

- ESC/Java
- HAVOC

- RAC
  test case gen.

- KeY
- VCC
- L4.verified

Target

- extended debugging
- increase confidence
- full functional verification

Effort:
- decrease effort ⇒ increase usability
  - enhance analyses
  - make things less difficult
  - push the limits
Program verification

Effort

- extensive annot. + hardly automatic
- annotations + mostly automatic
- little annot. + fully automatic
- no annotations

Target

- extended debugging
- increase confidence
- full functional verification

Effort decrease effort ⇒ increase usability

Peter H. Schmitt and Mattias Ulbrich – Program Verification-Next Steps
Program verification

- **Effort**
  - extensive annot. + hardly automatic
  - annotations + mostly automatic
  - little annot. + fully automatic
  - no annotations

- **Target**
  - extended debugging
  - increase confidence
  - full functional verification

- **Effort** → **Target**

- Peter H. Schmitt and Mattias Ulbrich – Program Verification-Next Steps

- November 16, 2010 3/9
Program verification

- extensive annot. + hardly automatic
- annotations + mostly automatic
- little annot. + fully automatic
- no annotations

Effort

Target

- increase confidence
- full functional verification
- extended debugging

decrease effort ⇒ increase usability

decrease effort =⇒ increase usability

effort decrease =⇒ increase usability
Program verification

Effort

- extensive annot. + hardly automatic
- annotations + mostly automatic
- little annot. + fully automatic
- no annotations

Target

- extended debugging
- increased evidence
- full functional verification

Effort: decrease effort ⇒ increase usability

Enhance analyses

Peter H. Schmitt and Mattias Ulbrich – Program Verification-Next Steps

November 16, 2010 3/9
Program verification

- Extend debugging
- Full functional verification
- Increase confidence

- Target
  - SBMC
  - Abs. Int.
  - SLAM
  - RAC
  - Test case gen.
  - ESC/Java
  - HAVOC
  - KeY
  - VCC
  - L4.verified

- No annotations
- Little annotations + fully automatic
- Annotations + mostly automatic
- Extensive annotations + hardly automatic

- Effort
- Decrease effort \(\Rightarrow\) increase usability
- Make things less difficult
- Enhance analyses

Peter H. Schmitt and Mattias Ulbrich – Program Verification-Next Steps
Program verification

No annotations

little annotation + mostly automatic

Annotations + hardly automatic

Extended annotation + fully automatic

Effort

Target

decrease effort \Rightarrow increase usability

push the limits

make things less difficult

enhance analyses

Peter H. Schmitt and Mattias Ulbrich – Program Verification-Next Steps
Next Steps (1)

Mature Specification Languages

1. Compare with programming languages: $C \overset{\sim}{\rightarrow}^{20\text{yrs}} \text{Java}$: types, semantics, portability, ...

2. Integrative spec language for light and heavy weight specification, common platform
   (RAC, deduction, testing, documentation, ...)

3. good data abstraction concept (framing problem)

4. abstract data types

5. candidates: JML, ACSL, CodeContracts, ...
   (however: Many tools, many syntaxes, many semantics)
Specified and Verified Libraries

1. needed for wide-spread use of verification
2. a large task
3. open research questions
4. full functional and/or special purpose?
5. which libraries?
6. should be a community effort
   (see JML specathlon)
Next Steps (3)

Domain Specific Specifications

1. model driven software development
2. conciser, shorter, easier to understand
3. broader audience
4. code and specification generation
5. examples: security flow properties, algorithmic properties
Next Steps (4)

Bridge the Gap between Model and Program Verification

MODEL
specification/verification

PROGRAM
spec/verification
Next Steps (4)

Bridge the Gap between Model and Program Verification

MODEL
specification/verification

PROGRAM
spec/verification
Next Steps (4)

Bridge the Gap between Model and Program Verification

1. established modelling methodologies like B, Z, ASM, ...
   Have a concept of refinement

2. integrate into software design process

3. code generation / specification generation
Next Steps (4)

Bridge the Gap between Model and Program Verification

1. established modelling methodologies like B, Z, ASM, ...
2. Have a concept of refinement
3. integrate into software design process
4. code generation / specification generation
Integrate Automation and Interaction

There will always be interaction if the problem is sufficiently difficult like loop invariants, quantifier instantiations, lemmata.

1. reduce interactions (inference, powerful decision procedures)
2. help verifying person find these auxiliary information
3. provide good feedback to where and what failed, and how to proceed.
Next steps (5)

Integrate Automation and Interaction

There will always be interaction if the problem is sufficiently difficult like loop invariants, quantifier instantiations, lemmata.

1. reduce interactions (inference, powerful decision procedures)
2. help verifying person find these auxiliary information
3. provide good feedback to where and what failed, and how to proceed.

How to proceed

- “Help me by providing an upper bound for int-variable x.”
- “Help me by providing evidence that $x > 5$ is part of the loop invariant in line...”
- “Adjust your post condition because $x > 5$ does not hold for input $y = 10$. ”
5 Next Steps for Usability

1. Mature Specification Languages
2. Specified and Verified Libraries
3. Domain Specific Specifications
4. Bridge the Gap between Model and Program Verification
5. Integrate Automation and Interaction

Thank you