Challenges
In Deductive Software Verification

Reiner Hähnle (with Marieke Huisman, U Twente)
www.se.tu-darmstadt.de

TU Darmstadt, Software Engineering Group
Reasoning about Correctness of Programs
As automatic and as precise as possible

→ Subfield of AD since early days (Bledsoe, Manna & Waldinger,...)
→ A main consumer of AD technology
→ Driver of AD research: theory reasoning, SMT, tactics
Deductive Software Verification

Reasoning about Correctness of Programs
As automatic and as precise as possible

→ Subfield of AD since early days (Bledsoe, Manna & Waldinger, ...)
→ A main consumer of AD technology
→ Driver of AD research: theory reasoning, SMT, tactics

Many “challenges” apply to AD in general
“Specification is the New Bottleneck”
— Beckert et al, Systems Software Verification, pp18–32, 2012

→ Programming languages more concise than specification languages
→ Specifications larger, more complex than code
→ Modular verification requires contracts
“Specification is the New Bottleneck”

— Beckert et al, Systems Software Verification, pp18–32, 2012

→ Programming languages more concise than specification languages
→ Specifications larger, more complex than code
→ Modular verification requires contracts

Challenge
Specify program boundaries: library, system, GUI

Challenge
Invest in debugging & understanding of specs

Challenge
Automate specification generation
Integration

- Integration at **tool** level
- **Method integration** (Model checking, symb. ex. abstraction, ...)
- Integration into **production environment**
Integration

→ Integration at tool level
→ Method integration (Model checking, symb. ex. abstraction, ...)
→ Integration into production environment

Challenge
“Universal” intermediate language with formal semantics

Challenge
Tool integration, API writing: no scientific reward

Challenge
Integrate proof management into GitHub
Coverage

- Mainstream languages not designed with **analyzability** in mind
- Problematic: concurrency, floating point, reflection
- Non-functional properties
Mainstream languages not designed with analyzability in mind

Problematic: concurrency, floating point, reflection

Non-functional properties

Challenge
Create a widely-used programming language designed to be analyzable

Challenge
Keep up with rapid evolution of mainstream industrial programming languages
(huge challenge for any academic tool)
Usability

→ Our research is **method- and tool-driven**
→ Benchmarks, case studies: no feedback on **usability**
→ “Winning CASC irrelevant for industrial stakeholder”
→ Need to demonstrate that tool **saves time, money**
Usability

→ Our research is method- and tool-driven
→ Benchmarks, case studies: no feedback on usability
→ “Winning CASC irrelevant for industrial stakeholder”
→ Need to demonstrate that tool saves time, money

**Challenge**
Back up claims on increased effectiveness or productivity by experimental user studies

**Challenge**
Establish paper category experimental user study

**Challenge**
Usability as driver for research investment
Funding

→ Mature deduction tools require sustained effort
→ Some research challenges require decades to address
→ ... but each <<your favorite funding agency>>-project must be a “breakthrough” and “disruptive”
Mature deduction tools require **sustained** effort

Some research challenges require **decades** to address

... but each **<<your favorite funding agency>>**-project must be a “breakthrough” and “disruptive”

**Challenge**
Academic reward system must provide incentives

**Challenge**
Computer Science must be re-classified as engineering or experimental science with according infrastructure
Industrial and Societal Context

- Digitalization of everything is huge opportunity for formal methods and, hence, AD technology
- Tool-based software analysis also applicable to CPS
- Certification goes from HW to SW
- But, we must find out what industry needs
Industrial and Societal Context

→ Digitalization of everything is huge opportunity for formal methods and, hence, AD technology
→ Tool-based software analysis also applicable to CPS
→ Certification goes from HW to SW
→ But, we must find out what industry needs

**Challenge**
Get involved in standardization efforts (languages, certification)

**Challenge**
Quality control for deduction/verification tools (robustness, usability, learnability, ...)