

The Verified Software Roadmap

N. Shankar

shankar@csl.sri.com

URL: <http://www.csl.sri.com/~shankar/>

Computer Science Laboratory
SRI International
Menlo Park, CA

Overview

Professor Tony Hoare has proposed the goal of automatically verified software as a *grand scientific challenge* for computing.

A series of meetings (funded by NCO HCSS through NSF) have been organized to explore the nature of this challenge.

A preliminary workshop was held at SRI International in Washington DC in April 2004 and was attended by about 50 participants.

A larger workshop was held during Feb 21–23, 2005 at SRI International in Menlo Park (<http://www.csl.sri.com/users/shankar/VGC05/>)

An even larger working conference was held in Zurich, Switzerland during the week of Oct. 10, 2005 (<http://vstte.inf.ethz.ch>).

Outline

- A brief history of verification
- Hoare's verification grand challenge
- The Road Ahead
- Agenda
- Discussion

A Brief History of Verification

1947-49: Goldstine/von Neumann and Turing: Assertional hand proofs of program correctness.

1961-63: McCarthy's *A Basis for a Mathematical Theory of Computation*

1966/67: Floyd introduces assertional reasoning on flowcharts for proving partial and total correctness.

1969: Hoare introduces axiomatic semantics for programming constructs. King writes a dissertation on automatic program proving through verification condition generation.

Fast Forward . . .

1970s: Predicate transformer semantics, LCF/ML, Boyer-Moore prover, VDM, abstract interpretation, algebraic data types, temporal logic, combination decision procedures.

de Millo, Lipton, and Perlis on *Social Processes and Proof*.

1980s: Model checking, hardware verification, HOL/Nuprl/Coq/Isabelle/EHDM, UNITY, TLA, I/O automata, Z specification language, OBJ3, KIDS.

1990s: Symbolic model checking, timed/hybrid model checking, predicate abstraction, bounded model checking, B Method, proof carrying code, typed assembly language.

Intel FDIV and aborted Ariane-5 launch.

... To the Present

Industrial use of hardware verification (AMD, Intel, Synopsys, Cadence, Mentor Graphics).

Microsoft's SLAM project for device driver verification (uses theorem proving, predicate abstraction, and model checking).

Large-scale program analysis: A380, Ariane-5, Linux/OpenBSD kernel.

A diverse suite of experimental tools with competitions, intermediate formats, API standardization.

Several large-scale verification systems/projects (ESC/Java, Spec#, Verisoft, LOOP, JML).

Many interesting applications: hardware, security, cryptographic protocols, communication protocols, AI planning.

Hoare's Verification Grand Challenge

A mature scientific discipline should set its own agenda and pursue ideals of purity, generality, and accuracy far beyond current needs.

Science explains why things work in full generality by means of calculation and experiment.

An engineering discipline exploits scientific principles to the study of the specification, design, construction, and production of working artifacts, and improvements to both process and design.

The verification challenge is to achieve a significant body of verified programs that have precise external specifications, complete internal specifications, machine-checked proofs of correctness with respect to a sound theory of programming.

The Deliverables

A *comprehensive theory* of programming that covers the features needed to build practical and reliable programs.

A *coherent toolset* that automates the theory and scales up to the analysis of large codes.

A *collection of verified programs* that replace existing unverified ones, and continue to evolve in a verified state.

“You can’t say anymore it can’t be done! Here, we have done it. ”

Hoare and Misra's Timetable

The long-term goal is to establish that the science and practice of computing do converge. The principles of specification, design, architecture, language, and semantics are employed in practice and taught to students.

In 20 years, we will have a well-developed theory, a comprehensive and powerful suite of tools, and compelling body of experimental evidence that reliable software can be produced employing formal verification techniques.

In the next 5 years, the foundation will be laid for the work ahead through the development of mature tools and standards.

Verified software \neq Verifying the code.

First Step: A Roadmap for Research

The roadmap document should spell out a fifteen-year coordinated program of research in incremental steps.

The key points of the roadmap are

1. Accelerated scaling of the performance, robustness, and functionality of basic verification technology.
2. Integration and embedding of this technology in program development and verification methodologies and environments, and in teaching.
3. Pilot projects and challenges that convincingly demonstrate the feasibility of rigorous, large-scale verification.

Panels

- Correct-by-construction (Leavens)
- Integrated verification environments (Leino)
- Theory (Naumann)
- System reliability (Rushby)
- Interoperable tools (Shankar)
- Verified Software Repository (Woodcock)
- Pilot Projects (Joshi)

Agenda

Sat. April 1, 2006

9-10AM: Overview (Shankar), Repository (Woodcock)

Coffee 10.30: Preliminary panel presentations

McCarthy, Leavens, Leino, Naumann, Joshi. Shankar, Rushby.

12.30-1.30PM: Lunch

1.30-5.30PM: Split into subgroups with writing assignments

Sun April 2, 2006 (Clocks spring forward)

9AM- 12 noon: Panel presentations

Lunch

1-5PM: Elaboration of outlines in subgroups.