Formal Verification

John Harrison
Intel Corporation, Hillsboro OR, USA

Formal verification (FV) involves the use of logical reasoning to establish properties of a computer program (or hardware element, protocol etc.) In the purest form of FV, rigorous logical deduction is used to prove with full mathematical rigor that the program satisfies a logical specification. This is often contrasted with the traditional approach of testing on a variety of specific inputs or situations, which may be a useful way of increasing confidence and catching bugs, but is usually not completely exhaustive or conclusive.

However, there is increasing interest in intermediate uses of formal methods, which establish non-trivial properties of the program (for example, that there will be no numeric overflow) without necessarily proving full functional correctness. Thus, it can be fruitful to think of FV as occupying a continuum of static analysis techniques with traditional approaches (basic type systems, lint) at one end and full functional correctness proofs at the other.

Modern FV is supported and made more practical by a wide variety of automatic or semi-automatic tools, including SAT (Boolean satisfiability) solvers, model checkers, automated theorem provers and general interactive proof assistants. In the lectures I will talk both about these tools themselves and how they can be applied.

Bradley and Manna [1] and Kroening and Strichman [4] describe various logical decision procedures and how they can be applied to program verification. Clarke, Grumberg, and Peled [2] discusses model checking in more detail, while Peled [7]) and Kropf [5]) survey various approaches to verification in, respectively, software and hardware applications. Harrison [3] gives a wider introductory survey of automated reasoning, while MacKenzie [6] is an engaging historical account of the intertwined development of theorem proving and program verification.

References

1. A.R. Bradley, Z. Manna. *The Calculus of Computation: Decision Procedures with Applications to Verification.* Springer; 2007.
2. E.M. Clarke, O. Grumberg, D. Peled. *Model Checking.* MIT Press; 1999.
3. J. Harrison. *Handbook of Practical Logic and Automated Reasoning.* Cambridge University Press; 2009.
4. D. Kroening, O. Strichman. *Decision Procedures: An Algorithmic Point of View.* Springer; 2008.
5. T. Kropf. *Introduction to Formal Hardware Verification.* Springer; 1999.
6. D. MacKenzie. *Mechanizing Proof: Computing, Risk and Trust.* MIT Press; 2001.
7. D. Peled. *Software Reliability Methods.* Springer; 2001.