Automatic Guidance for the Formal Verification of High Integrity Ada

Andrew Ireland
Department of Computing & Electrical Engineering
Heriot-Watt University
Edinburgh, Scotland
E-mail: A.Ireland@hw.ac.uk

Abstract

While the value of formal methods within the development of critical systems is recognized, scalability and usability remain key challenges to their wider adoption. Automation and high-level interaction will play an important role in meeting these challenges. Our aim is to investigate[1] techniques for automating the formal verification of software for critical systems. In particular we will focus upon SPARK [1], a subset of the Ada programming language developed by Praxis Critical Systems[2]. SPARK has a proven track record in the production of software for critical systems within the finance, aerospace, rail and telecommunications markets. The value of formal verification is recognized by Praxis as being crucial to the production of critical systems. This is reflected within the SPARK toolkit that supports formal verification through the SPADE Proof Checker. The SPADE Proof Checker, however, is interactive and requires highly skilled users with development times typically measured in person-months. The need for greater efficiency within the formal verification process is recognized by Praxis. Proof planning, as implemented within CLAM [3], has been investigated extensively within the context of proof by mathematical induction [2]. What distinguishes proof planning from other approaches to theorem proving is the flexibility it offers by separating the issues of search (method level) and soundness (tactic level). This means that the planning of a proof need not follow a strict backward or forward style of construction. This flexibility gave rise to *proof critics* [4, 6], an extension that supports the automatic analysis and patching of failed proof planning attempts. The most striking proof planning successes have been achieved where proofs require the discovery of auxiliary inductive lemmas or generalizations [5, 6, 7]. Such proof discovery capabilities outperform the conventional inductive theorem provers. Loop invariants are by their nature inductive so the proof plans for induction easily transferred to the problem of reasoning about imperative programs [10, 11, 12, 9]. In terms of high-level interaction, proof failures are also typically where experienced users make significant interventions. Proof critics, therefore, provide a very natural mechanism for managing user interaction as well as automatic proof patching. This observation gave rise to *interactive proof critics* [8], a CLAM interface that gives the user the opportunity of assisting with the proof patching process. Consequently, interactive proof critics extend the scope of the proof patching capabilities of CLAM by facilitating a collaborative style of interaction.

The time is now ripe to test the proof discovery and high-level interaction capabilities of proof planning on "industrial strength" problems. SPARK and the associated SPADE Proof Checker provide an the ideal vehicle given its commercial success within the production of

[1] The research proposal outlined here has been funded under the EPSRC Critical Systems Programme – EPSRC grant GR/R24081.

[2] `http://www.praxis-cs.co.uk`

critical systems. We believe that proof planning techniques can make a significant impact on the SPARK approach to producing high integrity software. We also believe that working on "industrial strength" problems will be beneficial to our basic research agenda.

References

[1] J. Barnes. *High Integrity Ada: The Spark Approach.* Addison-Wesley, 1997.

[2] A. Bundy, A. Stevens, F. van Harmelen, A. Ireland, and A. Smaill. Rippling: A heuristic for guiding inductive proofs. *Artificial Intelligence*, 62:185–253, 1993. Also available from Edinburgh as DAI Research Paper No. 567.

[3] A. Bundy, F. van Harmelen, C. Horn, and A. Smaill. The Oyster-Clam system. In M. E. Stickel, editor, *10th International Conference on Automated Deduction*, pages 647–648. Springer-Verlag, 1990. Lecture Notes in Artificial Intelligence No. 449. Also available from Edinburgh as DAI Research Paper 507.

[4] A. Ireland. The Use of Planning Critics in Mechanizing Inductive Proofs. In A. Voronkov, editor, *International Conference on Logic Programming and Automated Reasoning – LPAR 92, St. Petersburg*, Lecture Notes in Artificial Intelligence No. 624, pages 178–189. Springer-Verlag, 1992. Also available from Edinburgh as DAI Research Paper 592.

[5] A. Ireland and A. Bundy. Extensions to a Generalization Critic for Inductive Proof. In M. A. McRobbie and J. K. Slaney, editors, *13th International Conference on Automated Deduction*, pages 47–61. Springer-Verlag, 1996. Springer Lecture Notes in Artificial Intelligence No. 1104. Also available from Edinburgh as DAI Research Paper 786.

[6] A. Ireland and A. Bundy. Productive use of failure in inductive proof. *Journal of Automated Reasoning*, 16(1–2):79–111, 1996. Also available as DAI Research Paper No 716, Dept. of Artificial Intelligence, Edinburgh.

[7] A. Ireland and A. Bundy. Automatic Verification of Functions with Accumulating Parameters. *Journal of Functional Programming: Special Issue on Theorem Proving & Functional Programming*, 9(2):225–245, March 1999. A longer version is available from Dept. of Computing and Electrical Engineering, Heriot-Watt University, Research Memo RM/97/11.

[8] A. Ireland, M. Jackson, and G. Reid. Interactive Proof Critics. *Formal Aspects of Computing: The International Journal of Formal Methods*, 11(3):302–325, 1999. A longer version is available from Dept. of Computing and Electrical Engineering, Heriot-Watt University, Research Memo RM/98/15.

[9] A. Ireland and J. Spark. Proof planning for strategy development. Research Memo RM/00/3, Dept. of Computing and Electrical Engineering, Heriot-Watt University, 2000. To appear in a Special Issue of the Annals of Mathematics and Artificial Intelligence on Strategies in Automated Deduction.

[10] A. Ireland and J. Stark. On the Automatic Discovery of Loop Invariants. In *Proceedings of the Fourth NASA Langley Formal Methods Workshop – NASA Conference Publication 3356*, 1997. Also available from Dept. of Computing and Electrical Engineering, Heriot-Watt University, Research Memo RM/97/1.

[11] J. Stark and A. Ireland. Invariant discovery via failed proof attempts. In P. Flener, editor, *Logic-Based Program Synthesis and Transformation*, number 1559 in LNCS, pages 271–288. Springer-Verlag, 1998. An earlier version is available from the Dept. of Computing and Electrical Engineering, Heriot-Watt University, Research Memo RM/98/2.

[12] J. Stark and A. Ireland. Towards automatic imperative program synthesis through proof planning. In *The* 14^{th} *IEEE International Conference on Automated Software Engineering*, pages 44–51. IEEE Computer Society, 1999.