Formal Methods and Argument-Based Safety Cases

John Rushby
Computer Science Laboratory, SRI International, Menlo Park CA, USA

Safety-critical systems must be supplied with strong assurance that they are, indeed, safe. Traditionally, assurance has been based on adherence to standards, but recently the idea of a *safety case* has been gaining acceptance. A safety case provides an explicit *argument* that a system is safe to deploy; the notion of "safe" is made precise in suitable *claims* or *goals* for the system and its context of deployment, and the argument is intended to substantiate these claims, based on *evidence* concerning the system and its design and construction. The approach can be applied recursively, so that substantiated claims about a subsystem can be used as evidence in a parent case. Evaluators examine the case and may certify the system if they are persuaded that the claims are appropriate, the evidence is valid, and the argument is correct.

I will describe the safety-case approach and contrast it with use of standards and outline some of the advantages and disadvantages of each. I will then describe potentially useful applications of formal methods in the construction and evaluation of safety cases.

One application is use of infinite bounded model checking in automated analysis of abstract designs, both for verification of correctness and synthesis of assumptions. I will sketch how infinite bounded model checking and k-induction are automated, and the operation of the Satisfiability Modulo Theories (SMT) solvers on which they depend, and will illustrate their application to some small but interesting examples from fault-tolerant and real-time systems.

Another application is use of theorem proving to analyze the overall argument of a safety case. I will illustrate this and will also introduce the distinction between *formal proof* and *argument* and will discuss some of the issues and controversies around these, which extend from verification of computer systems into epistemology and the philosophy of science.

Finally, I will discuss the conundrum that the top-level goals for most critical systems are stated quantitatively – for example, "no catastrophic failure in the lifetime of all airplanes of one type" – and these translate into probabilistic requirements for subsystems and hence for software, for example, a probability of failure less than 10^{-9} per hour for flight-critical software, whereas the evidence produced for software is generally about correctness and does not seem to support quantitative reliability claims.

I will resolve this conundrum by arguing that what verification actually does is provide evidence for assessing a probability of "possible perfection". Possible perfection does relate to reliability and has other attractive properties that I will describe. I will explain how formal verification can allow assessment of a probability of perfection, and will discuss plausible values for this probability and consequences for correctness of verification systems themselves.

For safety cases, see papers by Tim Kelly, University of York, and by Adelard, City University [1]. Kelly's thesis is good place to start.

For modern formal methods, my paper [4] is a reasonable brief overview, and a recent paper from Rockwell Collins [3] describes their application in safety-critical avionics.

For possible perfection, see a technical report by Bev Littlewood and myself [2], or a shorter paper by me [5].

For assurance of verification systems themselves, see a very recent paper by Shankar [6]; I will make sure this is available online.

References

1. T. Kelly. *Arguing Safety – A Systematic Approach to Safety Case Management.* PhD thesis; Dept. of Computer Science, University of York, UK; 1998.
2. B. Littlewood, J. Rushby. *Reasoning about the Reliability of Fault-tolerant Systems in which one Component is "possibly perfect".* Technical Report SRI-CSL-09-02, Computer Science Laboratory, SRI International, 2009; Revised January 2010.
3. S. P. Miller, M. W. Whalen, D. D. Cofer. *Software Model Checking takes off.* Communications of the ACM, Vol. 53(2); pp. 58-64; 2010.
4. J. Rushby. *Automated Formal Methods enter the Mainstream.* Communications of the Computer Society of India, Vol. 31(2); pp. 28-32; 2007.
   Special Theme *Issue on Formal Methods* edited by R. Banach. Archived in Journal of Universal Computer Science, Vol. 13(5); pp. 650-660. `http://www.jucs.org/jucs_13_5`
5. J. Rushby. *Software Verification and System Assurance.* D. Van Hung, P. Krishnan (eds.), 7th International Conf. Software Engineering and Formal Methods (SEFM); IEEE Computer Society; pp. 3-10; 2009.
6. N. Shankar. *Rewriting, Inference, and Proof.* International Workshop on Rewriting Logic and its Applications, 2010. To appear.