Using and Building an Automatic Program Verifier

Rustan Leino
Microsoft Research, Redmond, USA

These lectures give a taste of what current technology can do for program correctness.

As computer scientists, we sometimes find the need to reason rigorously about the correctness of our programs. Such reasoning can be assisted by an automatic program verifier. One of the aims in these lectures is to teach concepts needed to use a program verifier and to give students some experience with a program verifier (namely, with the language and verifier Dafny). This will start with basic ideas like procedure specifications and loop invariants, and cover the more advanced topic of how to specify data-structure invariants in heap-manipulating programs. The take-home message is for students to remember verification and verification tools the next time a program is not entirely obvious.

As researchers in field of software correctness, we often find the need to build some program-analysis tool, be it a simple checker or a full-blown program verifier. Satisfiability-modulo-theories (SMT) solvers provide automated reasoning power for many common domains, and therefore make a useful component of many program analyses. To build a tool that uses an SMT solver also requires a lot of plumbing, plumbing that is shared among many similar analysis tools. The second aim of these lectures is to teach theoretical foundations of this shared infrastructure, described in terms of an intermediate verification language (namely, the language and verification engine Boogie). The lectures will also teach the main ideas behind the translation of an analysis problems into Boogie. The take-home message is: don't build your plumbing from scratch, leverage an intermediate verification language.

References

[1] M. Barnett, B-Y. E. Chang, R. DeLine, B. Jacobs, K.R.M. Leino. *Boogie: A Modular Reusable Verifier for Object-Oriented Programs.* In: F. S. de Boer, M. M. Bonsangue, S. Graf, W-P. de Roever (eds), Formal Methods for Components and Objects: 4th International Symposium, FMCO 2005; LNCS 4111, pp. 364-387, Springer, 2006.

[2] E. W. Dijkstra, W. H. J. Feijen. *A Method of Programming.* Addison-Wesley, 1988.

[3] D. Gries. *The Science of Programming.* Texts and Monographs in Computer Science, Springer 1981.

[4] A. Kaldewaij. *Programming: The Derivation of Algorithms.* International Series in Computer Science, Prentice-Hall, 1990.

[5] K.R.M. Leino. *Specification and Verification of Object-Oriented Software.* In: M. Broy, W. Sitou, T. Hoare (eds.), Engineering Methods and Tools for Software Safety and Security; Vol. 22 of NATO Science for Peace and Security Series D: Information and Communication Security, Summer School Marktoberdorf 2008 lecture notes, pp. 231-266, IOS Press, 2009.

[6] K.R.M. Leino. *Dafny: An Automatic Program Verifier for Functional Correctness.* In: LPAR-16, LNCS 6355, pp. 348-370, Springer, 2010.