Specification and Verification of Object-Oriented Software

K. Rustan M. Leino
Microsoft Research, Redmond, USA

The specification of object-oriented and other pointer-based programs must be able to describe the structure of the program's dynamically allocated data as well as some abstract view of what the code implements. The verification of such programs can be done by generating logical verification conditions from the program and its specifications and then analyzing the verification conditions by a mechanical theorem prover.

In these lectures, I present an object-based language, Dafny, whose specifica- tions use the style of dynamic frames. I show how to write and specify programs in Dafny, and show how to build a first-order automatic program verifier for Dafny programs, generating the verification conditions as input to an automatic satisfiability-modulo-theories solver.

Related Reading Dynamic frames refer to a specification technique where one defines a portion of memory – a set of memory locations, a *frame* – and then specifies the effect of methods on this frame. Frames can change over time, making them *dynamic*. Dynamic frames were introduced by Kassios [8] and were first implemented in an automatic program verifier by Smans *et al.* [18]. A prevalent architecture of such verifiers first translates the source language to a primitive intermediate verification language, and then generates theorem-prover input from the intermediate language. ESC/Modula-3 [5] and ESC/Java [7] used early forms of this architecture, which is now further developed in Boogie [1] and Why [6]. A pedagogical development of the architecture for a core object-oriented language is given in previous Marktoberdorf lecture notes [14]. The style of dynamic-frames specifications bears some resemblance to the valid/state specification idiom in ESC/Modula-3 [5, 12], to data groups [10, 13], and to separation logic with predicates [17]. Alternatives are explored in JML [9], which uses universe types [16], and Spec# [3], which uses the Boogie methodology [2, 11, 4, 15].

References

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

2. M. Barnett, R. DeLine, M. Fähndrich, K.R.M. Leino, and W. Schulte. *Verification of Object-Oriented Programs with Invariants.* Journal of Object Technology, 3(6); pp 27–56; 2004.

3. M. Barnett, K.R.M. Leino, and W. Schulte. *The Spec# Programming System: An Overview.* In Construction and Analysis of Safe, Secure and Interoperable Smart devices (CASSIS 2004); G. Barthe, L. Burdy, M. Huisman, J.-L. Lanet, and T. Muntean (eds); LNCS 3362; pp 49–69; Springer; 2005.

4. M. Barnett and D. A. Naumann. *Friends need a bit more: Maintaining Invariants over Shared State.* In Seventh International Conference on Mathematics of Program Construction (MPC 2004); D. Kozen, C. Shankland (eds.); LNCS 3125, pp 54–84; Springer; 2004.

5. D.L. Detlefs, K.R.M. Leino, G. Nelson, and J.B. Saxe. *Extended Static Checking.* Research Report 159, Compaq Systems Research Center, December 1998.

6. J.-C. Filliâtre. *Why: a Multi-Language Multi-Prover Verification Tool.* Research Report 1366, LRI, Université Paris Sud, March 2003.

7. C. Flanagan, K.R.M. Leino, M. Lillibridge, G. Nelson, J.B. Saxe, and R. Stata. *Extended Static Checking for Java.* In Proc of Programming Language Design and Implementation (PLDI'02); Vol. 37(5) in SIGPLAN Notices; pp 234–245; ACM; 2002.

8. I.T. Kassios. *Dynamic Frames: Support for Framing, Dependencies and Sharing without Restrictions.* In "Formal Methods" 14th International Symposium on Formal Methods (FM 2006); J. Misra, T. Nipkow, and E. Sekerinski (eds); LNCS 4085; pp 268–283; Springer; 2006.

9. G.T. Leavens, A.L. Baker, and C. Ruby. *JML: A Notation for Detailed Design.* In "Behavioral Specifications of Businesses and Systems"; H. Kilov, B. Rumpe, and I. Simmonds (eds); pp 175–188; Kluwer Academic Publishers; 1999.

10. K.R.M. Leino. *Data Groups: Specifying the Modification of Extended State.* In Proc of the 1998 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA'98), VOL 33 (10); SIGPLAN Notices, pp 144–153; ACM; 1998.

11. K.R.M. Leino and P. Müller. *Object Invariants in Dynamic Contexts.* In European Conference on Object-Oriented Programming (ECOOP'04); M. Odersky (ed); LNCS 3086; pp 491–516; Springer; 2004.

12. K.R.M. Leino and G. Nelson. *Data Abstraction and Information Hiding.* ACM Transactions on Programming Languages and Systems, 24(5); pp 491–553; 2002.

13. K.R.M. Leino, A. Poetzsch-Heffter, and Y. Zhou. *Using Data Groups to Specify and Check Side Effects.* In Proc of the Programming Language Design and Implementation (PLDI'02), Vol 37(5); SIGPLAN Notices; pp 246–257; ACM;2002.

14. K.R.M. Leino and W. Schulte. *A Verifying Compiler for a Multi-Threaded Object-Oriented Language.* In "Programming Methodology", Summer School Marktoberdorf 2006; NATO ASI Series; Springer; 2007. `http://research.microsoft.com/~leino/papers.html`

15. K.R.M. Leino and A. Wallenburg. *Class-Local Object Invariants.* In First India Software Engineering Conference (ISEC 2008); ACM; 2008.

16. P. Müller, A. Poetzsch-Heffter, and G.T. Leavens. *Modular Invariants for Layered Object Structures.* Science of Computer Programming, 62; pp 253–286; 2006.

17. M.J. Parkinson and G.M. Bierman. *Separation Logic and Abstraction.* In Proc of the Principles of Programming Languages (POPL'05); J. Palsberg and M. Abadi (eds); pp 247–258; ACM; 2005.

18. J. Smans, B. Jacobs, F. Piessens, and W. Schulte. *An Automatic Verifier for Java-like Programs based on Dynamic Frames.* In Fundamental Approaches to Software Engineering (FASE 2007); LNCS 4422; Springer; 2008.