

# Technologies for finding errors in object-oriented software

K. Rustan M. Leino

Microsoft Research, Redmond, WA, USA  
<http://research.microsoft.com/~leino>  
leino@microsoft.com

Manuscript KRML 120, 25 August 2003.

**Abstract.** To know if a program is correct, one needs specifications that say what the program is intended to do or intended not to do. But how does one write such specifications for object-oriented software with subtypes and dynamically-dispatched methods, and how does one reason about the semantics of such programs? These lectures give the semantics of a language with modern object-oriented features and consider several issues in the specification of object-oriented programs, including some recently formalized methodology. The lectures also include a demo of a programming tool that reasons semantically about programs and uses an automatic theorem prover to find common errors, and go through some issues in designing such a checker.

This note contains a sample of references related to my lectures at the summer school on Formal Models of Software, 1–6 September 2003, École Polytechnique de Tunisie, Tunis, Tunisia.

## 0 Overview papers of Extended Static Checking tools and technology

0. David L. Detlefs, K. Rustan M. Leino, Greg Nelson, and James B. Saxe. Extended static checking. Research Report 159, Compaq Systems Research Center, December 1998.
1. Cormac Flanagan, K. Rustan M. Leino, Mark Lillibridge, Greg Nelson, James B. Saxe, and Raymie Stata. Extended static checking for Java. In *Proceedings of the 2002 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, volume 37, number 5 in *SIGPLAN Notices*, pages 234–245. ACM, May 2002.
2. K. Rustan M. Leino. Extended static checking: A ten-year perspective. In Reinhard Wilhelm, editor, *Informatics—10 Years Back, 10 Years Ahead*, volume 2000 of *Lecture Notes in Computer Science*. Springer, 2000.
3. K. Rustan M. Leino, Greg Nelson, and James B. Saxe. ESC/Java user’s manual. Technical Note 2000-002, Compaq Systems Research Center, October 2000.

## 1 ESC/Java checking technology

4. Edsger W. Dijkstra. *A Discipline of Programming*. Prentice Hall, Englewood Cliffs, NJ, 1976.

5. K. Rustan M. Leino, James B. Saxe, and Raymie Stata. Checking Java programs via guarded commands. In Bart Jacobs, Gary T. Leavens, Peter Müller, and Arnd Poetzsch-Heffter, editors, *Formal Techniques for Java Programs*, Technical Report 251. Fernuniversität Hagen, May 1999. Also available as Technical Note 1999-002, Compaq Systems Research Center.
6. Cormac Flanagan and James B. Saxe. Avoiding exponential explosion: Generating compact verification conditions. In *Conference Record of the 28th Annual ACM Symposium on Principles of Programming Languages*, pages 193–205. ACM, January 2001.
7. Charles Gregory Nelson. Techniques for program verification. Technical Report CSL-81-10, Xerox PARC, June 1981. The author’s PhD thesis.
8. David Detlefs, Greg Nelson, and James B. Saxe. Simplify: A theorem prover for program checking. Technical Report HPL-2003-148, HP Labs, July 2003.
9. Todd Millstein. Toward more informative ESC/Java warning messages. In James Mason, editor, *Selected 1999 SRC summer intern reports*, Technical Note 1999-003. Compaq Systems Research Center, 1999.
10. K. Rustan M. Leino, Todd Millstein, and James B. Saxe. Generating error traces from verification-condition counterexamples. Manuscript KRML 120, available from <http://research.microsoft.com/~leino>, 2003. Submitted.

## 2 Related work on full functional specification and verification

11. JML homepage. On the web at <http://www.cs.iastate.edu/~leavens/JML.html>.
12. Gary T. Leavens, Albert L. Baker, and Clyde Ruby. Preliminary design of JML: A behavioral interface specification language for Java. Technical Report 98-06f, Iowa State University, Department of Computer Science, July 1999.
13. Lilian Burdy, Yoonsik Cheon, David Cok, Michael D. Ernst, Joe Kiniry, Gary T. Leavens, K. Rustan M. Leino, and Erik Poll. An overview of JML tools and applications. In Thomas Arts and Wan Fokkink, editors, *Eighth International Workshop on Formal Methods for Industrial Critical Systems (FMICS '03)*, volume 80 of *Electronic Notes In Theoretical Computer Science*, pages 73–89. Elsevier, June 2003. Also available as Technical Report NIII-R0309, Department of Computer Science, University of Nijmegen.
14. Bart Jacobs, Joachim van den Berg, Marieke Huisman, Martijn van Berkum, Ulrich Hensel, and Hendrik Tews. Reasoning about Java classes (preliminary report). In *Proceedings of the 1998 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '98)*, volume 33, number 10 in *SIGPLAN Notices*, pages 329–340. ACM, October 1998.
15. Tiziana Margaria and Wang Yi, editors. *The LOOP compiler for Java and JML*, volume 2031 of *Lecture Notes in Computer Science*. Springer, April 2001.
16. J.-R. Abrial. *The B-Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.
17. David Guaspari, Carla Marceau, and Wolfgang Polak. Formal verification of Ada programs. *IEEE Transactions on Software Engineering*, 16(9):1058–1075, September 1990.

## 3 Related work on languages and language features

18. Butler W. Lampson, James J. Horning, Ralph L. London, James G. Mitchell, and Gerald J. Popek. Report on the programming language Euclid. Technical Report CSL-81-12, Xerox PARC, October 1981. An earlier version of this report appeared in *ACM SIGPLAN Notices*, 12(2), February 1977.

19. Bertrand Meyer. *Object-oriented Software Construction*. Series in Computer Science. Prentice-Hall International, New York, 1988.
20. Escher Technologies, Inc. Getting started with Perfect. Available from <http://www.eschertech.com>, 2001.
21. David Evans, John V. Guttag, James J. Horning, and Yang Meng Tan. LCLint: A tool for using specifications to check code. In David S. Wile, editor, *SIGSOFT '94, Proceedings of the Second ACM SIGSOFT Symposium on Foundations of Software Engineering*, ACM SIGSOFT Software Engineering Notes 19(5), pages 87–96, December 1994.
22. Robert DeLine and Manuel Fähndrich. Enforcing high-level protocols in low-level software. In *Proceedings of the 2001 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, volume 36, number 5 in *SIGPLAN Notices*, pages 59–69. ACM, May 2001.
23. Jeffrey S. Foster, Tachio Terauchi, and Alexander Aiken. Flow-sensitive type qualifiers. In *Proceedings of the 2002 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, volume 37, number 5 in *SIGPLAN Notices*, pages 1–12. ACM, May 2002.
24. Hongwei Xi and Frank Pfenning. Dependent types in practical programming. In *Conference Record of POPL '99: The 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 214–227, January 1999.
25. Cormac Flanagan and Steven N. Freund. Type-based race detection for Java. In *Proceedings of the 2000 ACM SIGPLAN conference on Programming Design and Implementation (PLDI)*, pages 219–232, 2000.
26. David F. Bacon, Robert E. Strom, and Ashis Tarafdar. Guava: A dialect of Java without data races. In *Proceedings of the 2000 ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages & Applications (OOPSLA 2000)*, volume 35, number 10 in *SIGPLAN Notices*, pages 382–400. ACM, October 2000.

#### 4 Related work on other checking techniques

27. Patrick Cousot and Rhadia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM Symposium on Principles of Programming Languages*, pages 238–252. ACM, January 1977.
28. Patrick Cousot. Semantic foundations of program analysis. In Steven S. Muchnick and Neil D. Jones, editors, *Program Flow Analysis: Theory and Applications*, chapter 10, pages 303–346. Prentice-Hall, Englewood Cliffs, NJ, 1981.
29. William R. Bush, Jonathan D. Pincus, and David J. Sielaff. A static analyzer for finding dynamic programming errors. *Software—Practice & Experience*, 30:775–802, 2000.
30. Thomas Ball and Sriram K. Rajamani. Automatically validating temporal safety properties of interfaces. In Matthew B. Dwyer, editor, *Model Checking Software, 8th International SPIN Workshop*, volume 2057 of *Lecture Notes in Computer Science*, pages 103–122. Springer, May 2001.
31. James C. Corbett, Matthew B. Dwyer, John Hatcliff, Shawn Laubach, Corina S. Pasareanu, Robby, and Hongjun Zheng. Bandera: Extracting finite-state models from Java source code. In *ICSE 2000, Proceedings of the 22nd International Conference on Software Engineering*, pages 439–448. ACM, June 2000.
32. Willem Visser, Klaus Havelund, Guillaume P. Brat, and Seungjoon Park. Model checking programs. In *The Fifteenth IEEE International Conference on Automated Software Engineering (ASE '00)*, pages 3–12. IEEE Computer Society, September 2000.

33. Manuvir Das, Sorin Lerner, and Mark Seigle. ESP: Path-sensitive program verification in polynomial time. In *Proceedings of the 2002 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, volume 37, number 5 in *SIGPLAN Notices*, pages 57–68. ACM, May 2002.
34. G. Ramalingam, Alex Warshavsky, John Field, Deepak Goyal, and Mooly Sagiv. Deriving specialized program analyses for certifying component-client conformance. In *Proceedings of the 2002 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, volume 37, number 5 in *SIGPLAN Notices*, pages 83–94. ACM, May 2002.
35. Seth Hallem, Benjamin Chelf, Yichen Xie, and Dawson R. Engler. A system and language for building system-specific, static analyses. In *Proceedings of the 2002 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, volume 37, number 5 in *SIGPLAN Notices*, pages 69–82. ACM, May 2002.

## 5 Work related to my presentation of program semantics

36. Edsger W. Dijkstra. *A Discipline of Programming*. Prentice Hall, Englewood Cliffs, NJ, 1976.
37. Greg Nelson. A generalization of Dijkstra’s calculus. *ACM Transactions on Programming Languages and Systems*, 11(4):517–561, 1989.
38. Ralph-Johan Back and Joakim von Wright. *Refinement Calculus: A Systematic Introduction*. Graduate Texts in Computer Science. Springer-Verlag, 1998.
39. C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580,583, October 1969.
40. Carroll Morgan. The specification statement. *ACM Transactions on Programming Languages and Systems*, 10(3):403–419, July 1988.
41. John McCarthy and James Painter. Correctness of a compiler for arithmetic expressions. In J.-T. Schwartz, editor, *Proc. Symposia in Applied Mathematics*. American Mathematical Society, 1967.
42. O.-J. Dahl, E. W. Dijkstra, and C. A. R. Hoare, editors. *Structured Programming*. Academic Press, 1972.
43. C. A. R. Hoare and N. Wirth. An axiomatic definition of the programming language PASCAL. *Acta Informatica*, 2(4):335–355, 1973.
44. G. Birkhoff. *Lattice Theory*, volume 25 of *Colloquium Publications*. American Mathematical Society, 1967.
45. Carroll Morgan. Data refinement using miracles. *Information Processing Letters*, 26(5):243–246, January 1988.
46. D. L. Parnas. A technique for software module specification with examples. *Communications of the ACM*, 15(5):330–336, May 1972.
47. K. Rustan M. Leino, James B. Saxe, and Raymie Stata. Checking Java programs via guarded commands. In Bart Jacobs, Gary T. Leavens, Peter Müller, and Arnd Poetzsch-Heffter, editors, *Formal Techniques for Java Programs*, Technical Report 251. Fernuniversität Hagen, May 1999. Also available as Technical Note 1999-002, Compaq Systems Research Center.

## 6 Object-oriented program semantics

48. K. Rustan M. Leino. *Toward Reliable Modular Programs*. PhD thesis, California Institute of Technology, 1995. Technical Report Caltech-CS-TR-95-03.

49. K. Rustan M. Leino. Ecstatic: An object-oriented programming language with an axiomatic semantics. In *The Fourth International Workshop on Foundations of Object-Oriented Languages*, January 1997. Proceedings available from <http://www.cis.upenn.edu/~bcpierce/FOOL/>.
50. K. Rustan M. Leino and Greg Nelson. Data abstraction and information hiding. *ACM Transactions on Programming Languages and Systems*, 24(5):491–553, September 2002.
51. K. Rustan M. Leino. Data groups: Specifying the modification of extended state. In *Proceedings of the 1998 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '98)*, volume 33, number 10 in *SIGPLAN Notices*, pages 144–153. ACM, October 1998.

## 7 Related work on data abstraction

52. C. A. R. Hoare. Proof of correctness of data representations. *Acta Informatica*, 1(4):271–81, 1972.
53. K. Rustan M. Leino. *Toward Reliable Modular Programs*. PhD thesis, California Institute of Technology, 1995. Technical Report Caltech-CS-TR-95-03.
54. K. Rustan M. Leino and Greg Nelson. Data abstraction and information hiding. *ACM Transactions on Programming Languages and Systems*, 24(5):491–553, September 2002.
55. K. Rustan M. Leino and Greg Nelson. Abstraction dependencies. In Annabelle McIver and Carroll Morgan, editors, *Programming Methodology*, Monographs in Computer Science, chapter 13, pages 269–289. Springer, 2002.
56. Peter Müller. *Modular Specification and Verification of Object-Oriented Programs*, volume 2262 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002. The author's PhD thesis, FernUniversität Hagen.
57. K. Rustan M. Leino. Data groups: Specifying the modification of extended state. In *Proceedings of the 1998 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '98)*, volume 33, number 10 in *SIGPLAN Notices*, pages 144–153. ACM, October 1998.
58. K. Rustan M. Leino, Arnd Poetzsch-Heffter, and Yunhong Zhou. Using data groups to specify and check side effects. In *Proceedings of the 2002 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, volume 37, number 5 in *SIGPLAN Notices*, pages 246–257. ACM, May 2002.

## 8 Related work on programming methodology

59. David L. Detlefs, K. Rustan M. Leino, Greg Nelson, and James B. Saxe. Extended static checking. Research Report 159, Compaq Systems Research Center, December 1998.
60. K. Rustan M. Leino and Greg Nelson. Data abstraction and information hiding. *ACM Transactions on Programming Languages and Systems*, 24(5):491–553, September 2002.
61. Bertrand Meyer. *Object-oriented Software Construction*. Series in Computer Science. Prentice-Hall International, New York, 1988.
62. K. Rustan M. Leino and Raymie Stata. Checking object invariants. Technical Note 1997-007, Digital Equipment Corporation Systems Research Center, January 1997.
63. K. Rustan M. Leino, Greg Nelson, and James B. Saxe. ESC/Java user's manual. Technical Note 2000-002, Compaq Systems Research Center, October 2000.

64. Cormac Flanagan, K. Rustan M. Leino, Mark Lillibridge, Greg Nelson, James B. Saxe, and Raymie Stata. Extended static checking for Java. In *Proceedings of the 2002 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, volume 37, number 5 in *SIGPLAN Notices*, pages 234–245. ACM, May 2002.
65. Mike Barnett, Robert DeLine, Manuel Fähndrich, K. Rustan M. Leino, and Wolfram Schulte. Verification of object-oriented programs with invariants. In Susan Eisenbach, Gary T. Leavens, Peter Müller, Arnd Poetsch-Heffter, and Erik Poll, editors, *Formal Techniques for Java-like Programs 2003*, July 2003. Available as Technical Report 408, Department of Computer Science, ETH Zurich.