

Bibliography

- [AdB94] P. America and F. de Boer. Reasoning about dynamically evolving process structures. *Formal Aspects of Computing*, 6:269–316, 1994.
- [AL97] M. Abadi and K. R. M. Leino. A logic of object-oriented programs. In M. Bidoit and M. Dauchet, editors, *TAPSOFT '97: Theory and Practice of Software Development, 7th International Joint Conference CAAP/FASE, Lille, France*, volume 1214 of *Lecture Notes in Computer Science*, pages 682–696. Springer-Verlag, 1997.
- [Alm97] P. S. Almeida. Balloon types: Controlling sharing of state in data types. In M. Akşit and S. Matsuoka, editors, *ECOOP '97: Object-Oriented Programming*, volume 1241 of *Lecture Notes in Computer Science*, pages 32–59. Springer-Verlag, 1997.
- [Ame83] American National Standards Institute, Inc. *Ada Programming Language*, ansi/mil-std-1815a edition, January 1983.
- [Ame87] P. America. Inheritance and subtyping in a parallel object-oriented language. In J. Bézivin, editor, *ECOOP '87, European Conference on Object-Oriented Programming, Paris, France*, volume 276 of *Lecture Notes in Computer Science*, pages 234–242. Springer-Verlag, 1987.
- [Ame89] P. America. A behavioural approach to subtyping in object-oriented programming languages. Technical Report 443, Philips Research Laboratories, Nederlandse Philips Bedrijven B. V., 1989.
- [Ame91] P. America. Designing an object-oriented programming language with behavioural subtyping. In J. W. de Bakker, W. P. de Roever, and G. Rozenberg, editors, *Foundations of Object-Oriented Languages*, volume 489 of *Lecture Notes in Computer Science*, pages 60–90. Springer-Verlag, 1991.
- [Apt81] K. R. Apt. Ten years of Hoare logic: A survey — part I. *ACM Trans. on Prog. Languages and Systems*, 3:431–483, 1981.
- [BA96] M. Ben-Ari. *Understanding Programming Languages*. John Wiley & Sons, 1996.
- [Bac88] R. J. R. Back. A calculus of refinement for program derivations. *Acta Informatica*, 25:593–624, 1988.
- [Ban95] G. S. Banavar. *An Application Framework for Compositional Modularity*. PhD thesis, The University of Utah, 1995.
- [Bar97] J. Barnes. *Ada 95 Rationale*, volume 1247 of *Lecture Notes in Computer Science*. Springer, 1997.
- [BC90] G. Bracha and W. Cook. Mixin-based inheritance. *ACM SIGPLAN Notices*, 25(10):303–311, October 1990. *OOPSLA ECOOP '90 Proceedings*, N. Meyrowitz (editor).

- [BG77] R. M. Burstall and J. A. Goguen. Putting theories together to make specifications. In *Proc. 5th International Joint Conference on Artificial Intelligence*, pages 1045–1058. Morgan Kaufmann Publishers, 1977.
- [BG94] P. Borba and J. A. Goguen. On refinement and FOOPS. Technical Report PRG-TR-17-94, Oxford University Computing Laboratory, 1994.
- [Bij89] A. Bijlsma. Calculating with pointers. *Science of Computer Programming*, 12:191–205, 1989.
- [BL91] G. Bracha and G. Lindstrom. Modularity meets inheritance. Technical Report UUCS-91-017, University of Utah, October 1991.
- [BLO94] G. S. Banavar, G. Lindstrom, and D. Orr. Type-safe composition of object modules. In *Computer Systems and Education*, pages 188–200. McGraw Hill, 1994. Also available as University of Utah Technical Report UUCS-94-001.
- [BMR95] A. Borgida, J. Mylopoulos, and R. Reiter. On the frame problem in procedure specifications. *IEEE Transactions on Software Engineering*, 21(10):785–798, October 1995.
- [Boe99] F. S. de Boer. A WP-calculus for OO. In W. Thomas, editor, *Foundations of Software Science and Computation Structures*, volume 1578 of *Lecture Notes in Computer Science*, pages 135–149. Springer-Verlag, 1999.
- [Bok99] B. Bokowski. Implementing “object ownership to order”. Presented at the Intercontinental Workshop on Aliasing in Object-Oriented Systems at ECOOP’99, 1999. Available from <http://cuiwww.unige.ch/~ecoopws/iwaoos/papers/index.html>.
- [Boo94] G. Booch. *Object oriented analysis and design with applications*. Addison-Wesley, 1994.
- [BPF97] K. B. Bruce, L. Petersen, and A. Fiech. Subtyping is not a good “match” for object-oriented languages. In M. Akşit and S. Matsuoka, editors, *ECOOP ’97: Object-Oriented Programming*, volume 1241 of *Lecture Notes in Computer Science*, pages 104–127. Springer-Verlag, 1997.
- [BPJ00] J. van den Berg, E. Poll, and B. Jacobs. First steps in formalising JML. In S. Drossopoulou, S. Eisenbach, B. Jacobs, G. T. Leavens, P. Müller, and A. Poetzsch-Heffter, editors, *Formal Techniques for Java Programs*. Technical Report 269, Fernuniversität Hagen, 2000. Available from www.informatik.fernuni-hagen.de/pi5/publications.html.
- [Bra92] G. Bracha. *The Programming Language Jigsaw: Mixins, Modularity and Multiple Inheritance*. PhD thesis, The University of Utah, 1992.
- [BRS99] M. Benedikt, T. Reps, and M. Sagiv. A decidable logic for describing linked data structures. In S. D. Swierstra, editor, *Programming Languages and Systems (ESOP ’99)*, volume 1576 of *Lecture Notes in Computer Science*, pages 2–19. Springer-Verlag, 1999.
- [Bud91] T. Budd, editor. *Object-Oriented Programming*. Addison-Wesley, 1991.
- [BV99] B. Bokowski and J. Vitek. Confined types. In *Proceedings of Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, ACM SIGPLAN Notices, 1999.
- [CDD⁺89] D. Carrington, D. Duke, R. Duck, P. King, and G. Rose. *Object-Z: an object oriented extension to Z*. North-Holland, 1989.

- [CFR93] T. R. Colburn, J. H. Fetzer, and T. L. Rankin. *Program Verification*. Kluwer Academic Publishers, 1993.
- [CGR96] P. Chalin, P. Grogono, and T. Radhakrishnan. Identification of and solutions to shortcomings of LCL, a larch/c interface specification language. In M.-C. Gaudel and J. Woodcock, editors, *FME '96: Industrial Benefit and Advances in Formal Methods*, volume 1051 of *Lecture Notes in Computer Science*, pages 385–404. Springer-Verlag, January 1996.
- [CH96] G. Cornell and C. S. Horstmann. *Java bis ins Detail*. Heise, 1996.
- [CL94] Y. Cheon and G. T. Leavens. The Larch/Smalltalk interface specification language. *ACM Transactions on Software Engineering and Methodology*, 3(3):221–253, July 1994.
- [Cla93] U. Claussen. *Objektorientiertes Programmieren*. Springer-Verlag, 1993.
- [COR⁺95] J. Crow, S. Owre, J. Rushby, N. Shankar, and M. Srivas. *A Tutorial Introduction to PVS*, April 1995.
- [Cou90] P. Cousot. Methods and logics for proving programs. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 15, pages 841–993. Elsevier Science Publishers, 1990.
- [CPN98] D. G. Clarke, J. M. Potter, and J. Noble. Ownership types for flexible alias protection. In *Proceedings of Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)*, volume 33(10) of *ACM SIGPLAN Notices*, October 1998.
- [Dav99] M. Davis. Immutables. *Java-Report*, 4(4):70–77, April 1999.
- [DGLM95] M. Day, R. Gruber, B. Liskov, and A. C. Myers. Subtypes vs. where clauses: Constraining parametric polymorphism. In *Proceedings of the 10th Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '95)*, volume 30 of *ACM SIGPLAN Notices*, pages 156–168, 1995.
- [Dha97] K. K. Dhara. Behavioral subtyping in object-oriented languages. Technical Report 97-09, Iowa State University, May 1997.
- [Dij76] E. W. Dijkstra. *A Discipline of Programming*. Prentice-Hall, 1976.
- [Dip98] P. Dippold. Logische Grundlagen einer Teilsprache von Java. Master's thesis, Fernuniversität Hagen, 1998. (in German).
- [DK92] E. H. Durr and J. van Katwijk. VDM++: A formal specification language for object-oriented design. In *TOOLS Europe '92*, pages 63–77, 1992.
- [DL96] K. K. Dhara and G. T. Leavens. Forcing behavioral subtyping through specification inheritance. In *Proceedings of the 18th International Conference on Software Engineering*, pages 258–267. IEEE Computer Society Press, 1996.
- [DLN98] D. L. Detlefs, K. R. M. Leino, and G. Nelson. Wrestling with rep exposure. Research Report 156, Digital Systems Research Center, 1998.
- [DLNS98] D. L. Detlefs, K. R. M. Leino, G. Nelson, and J. B. Saxe. Extended static checking. Research Report 159, Digital Systems Research Center, 1998. see also www.research.digital.com/SRC/esc/Esc.html.
- [ES90] M. A. Ellis and B. Stroustrup. *The Annotated C++ Reference Manual*. Addison-Wesley, 1990.
- [Fai85] R. E. Fairley. *Software Engineering Concepts*. McGraw-Hill, 1985.
- [FGJM85] K. Futatsugi, J. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of OBJ2. In *Principles of Programming Languages*, pages 52–66. ACM, 1985.

- [Flo67] R. W. Floyd. Assigning meanings to programs. In *Mathematical Aspects of Computer Science*, volume 19 of *Proceedings of Symposia in Applied Mathematics*, pages 19–32. American Mathematical Society, 1967.
- [FM96] J. Feiler and A. Meadow. *Essential OpenDoc*. Addison-Wesley, 1996.
- [FM98] C. Fischer and D. Meemken. JaWa: Java with assertions. In C. H. Cap, editor, *JIT '98 Java-Informationen-Tage 1998*. Springer-Verlag, 1998.
- [Gea97] D. M. Geary. *Graphic Java 1.1: Mastering the AWT*. Sun Microsystems Press, 1997.
- [GG91] S. J. Garland and J. V. Guttag. A guide to LP, the Larch Prover. Technical Report 82, Digital Systems Research Center, 1991.
- [GH93] J. V. Guttag and J. J. Horning. *Larch: Languages and Tools for Formal Specification*. Springer-Verlag, 1993.
- [GJS96] J. Gosling, B. Joy, and G. Steele. *The Java Language Specification*. Addison-Wesley, Reading, MA, 1996.
- [GMP90] D. Guaspari, C. Marceau, and W. Polak. Formal verification of Ada programs. *IEEE Transactions on Software Engineering*, 16(9):1058–1075, September 1990.
- [GP82] W. D. Gillett and S. V. Pollack. *An Introduction to Engineered Software*. Holt, Rinehart and Winston, 1982.
- [GTZ98] D. Genius, M. Trapp, and W. Zimmermann. An approach to improve locality using sandwich types. In X. Leroy and A. Ohori, editors, *Proceedings of the 2nd Types in Compilation Workshop*, volume 1473 of *Lecture Notes in Computer Science*, pages 194–214. Springer-Verlag, 1998.
- [GWM⁺00] J. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouannaud. Introducing OBJ. In G. Malcolm, editor, *Software Engineering with OBJ: algebraic specification in action*. Kluwer, 2000.
- [Ham97] G. Hamilton. *JavaBeans*. Sun Microsystems, Inc., 1997. Available from <http://java.sun.com/beans/docs/spec.html>
- [Har92] S. P. Harbison, editor. *Modula-3*. Prentice Hall, 1992.
- [HC97] C. S. Horstmann and G. Cornell. *Core Java*. Sunsoft Press, 1997.
- [Heh93] E. C. R. Hehner. *A Practical Theory of Programming*. Texts and Monographs in Computer Science. Springer-Verlag, 1993.
- [HJ99] M. Huisman and B. Jacobs. Java program verification via a Hoare logic with abrupt termination. Technical Report CSI-R9912, Computing Science Institute, Univ. Nijmegen, 1999.
- [HJ00] M. Huisman and B. Jacobs. Java program verification via a Hoare logic with abrupt termination. In E. Maibaum, editor, *Fundamental Approaches to Software Engineering*, volume 1783 of *Lecture Notes in Computer Science*, pages 284–303. Springer-Verlag, 2000.
- [HK00] K. Huizing and R. Kuiper. Verification of object-oriented programs using class invariants. In E. Maibaum, editor, *Fundamental Approaches to Software Engineering*, volume 1783 of *Lecture Notes in Computer Science*, pages 208–221. Springer-Verlag, 2000.
- [HLW⁺92] J. Hogg, D. Lea, A. Wills, D. de Champeaux, and R. Holt. Report on ECOOP'91 workshop W3: The Geneva convention on the treatment of object aliasing. *OOPS Messenger*, 3(2):11–16, 1992.
- [Hoa69] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 583, 1969.
- [Hoa72] C. A. R. Hoare. Proofs of correctness of data representation. *Acta Informatica*, 1:271–281, 1972.

- [Hog91] J. Hogg. Islands: Aliasing protection in object-oriented languages. In A. Paepcke, editor, *OOPSLA '91 Conference Proceedings*, pages 271–285, October 1991. SIGPLAN Notices, 26 (11).
- [Hol91] I. J. Holyer. *Functional Programming with Miranda*. Pitman, 1991.
- [HU79] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
- [HW73] C. A. R. Hoare and N. Wirth. An axiomatic definition of the programming language PASCAL. *Acta Informatica*, pages 335–355, 1973.
- [IP00] A. Igarashi and B. C. Pierce. On inner classes. In E. Bertino, editor, *ECOOP 2000: Object-Oriented Programming*, volume 1850 of *Lecture Notes in Computer Science*, pages 129–153. Springer-Verlag, 2000.
- [JBH⁺98] B. Jacobs, J. van den Berg, M. Huisman, M. van Berkum, U. Hensel, and H. Tews. Reasoning about Java classes. In *Proceedings of Object-Oriented Programming Systems, Languages and Applications (OOPSLA)*, 1998. Also available as TR CSI-R9812, University of Nijmegen.
- [JLMPH99] B. Jacobs, G. T. Leavens, P. Müller, and A. Poetzsch-Heffter. Formal techniques for Java programs. In A. Moreira and D. Demeyer, editors, *Object-Oriented Technology. ECOOP'99 Workshop Reader*, volume 1743 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999. Available from www.informatik.fernuni-hagen.de/pi5/publications.html.
- [JML] JML interest list. Archive at www.cs.iastate.edu/~leavens/JML.html.
- [Jon90] C. B. Jones. *Systematic Software Development using VDM*. Prentice Hall, 1990.
- [Jon91a] K. D. Jones. LM3: A Larch interface language for Modula-3: A definition and introduction. Technical Report 72, Digital Equipment Corporation, Systems Research Center, 1991.
- [Jon91b] H. B. M. Jonkers. Upgrading the pre- and postcondition technique. In S. Prehn and W. J. Toetenel, editors, *VDM '91: Formal Software Development Methods*, volume 551 of *Lecture Notes in Computer Science*, pages 428–456. Springer-Verlag, 1991.
- [Jos97] R. Joshi. Extended static checking of programs with cyclic dependencies. Technical Note 1997-028, Digital Systems Research Center, 1997. In J. Mason, editor, *1997 SRC Intern Projects*.
- [Kas90] U. Kastens. *Übersetzerbau*. Oldenburg, 1990.
- [Kee89] S. E. Keene. *Object-Oriented Programming in Common Lisp*. Addison-Wesley, 1989.
- [KMMPN83] B. Bruun Kristensen, O. Lehrmann Madsen, B. Møller-Pedersen, and K. Nygaard. Abstraction mechanisms in the BETA programming language. In *Tenth ACM Symposium on Principles of Programming Languages*, pages 285–298, Austin, Texas, 1983.
- [KR88] B. W. Kernighan and D. M. Ritchie. *The C Programming Language*. Prentice Hall, 1988.
- [Kra98] R. Kramer. iContract—the Java Design by Contract tool. In R. Ege, M. Singh, and B. Meyer, editors, *Technology of Object-Oriented Languages Tools 26*. IEEE Computer Society, 1998.
- [Kru92] C. W. Krueger. Software reuse. *ACM Computing Surveys*, 24(2):131–183, June 1992.

- [KT90] D. Kozen and J. Tiuryn. Logics of programs. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 14, pages 789–840. Elsevier Science Publishers, 1990.
- [KT99] G. Kniesel and D. Theissen. JAC — Java with transitive readonly access control. Presented at the Intercontinental Workshop on Aliasing in Object-Oriented Systems at ECOOP’99, 1999. Available from <http://cuiwww.unige.ch/~ecoopws/iwaoos/papers/index.html>.
- [Kuh70] T. S. Kuhn. *The Structure of Scientific Revolutions*, volume 2 of *International Encyclopedia of Unified Science*. The University of Chicago Press, 2nd edition, 1970.
- [Lak96] J. Lakos. *Large-Scale C++ Software Design*. Addison-Wesley, 1996.
- [Lam86] L. Lamport. *TEX: A document preparation system*. Addison-Wesley, 1986.
- [Lar] Larch frequently asked questions. Available from www.cs.iastate.edu/~leavens/larch-faq.html.
- [LB99] G. T. Leavens and A. L. Baker. Enhancing the pre- and postcondition technique for more expressive specifications. In J. M. Wing, J. Woodcock, and J. Davies, editors, *FM’99 – Formal Methods: World Congress on Formal Methods in Development of Computer Systems*, volume 1709 of *Lecture Notes in Computer Science*, pages 1087–1106. Springer-Verlag, 1999.
- [LBR99a] G. T. Leavens, A. L. Baker, and C. Ruby. JML: A notation for detailed design. In H. Kilov, B. Rumpe, and I. Simmonds, editors, *Behavioral Specifications of Businesses and Systems*, pages 175–188. Kluwer Academic Publishers, 1999.
- [LBR99b] G. T. Leavens, A. L. Baker, and C. Ruby. Preliminary design of JML: A behavioral interface specification language for Java. Technical Report 98-06c, Iowa State University, Department of Computer Science, January 1999.
- [LCD⁺94] B. Liskov, D. Curtis, M. Day, S. Ghemawhat, R. Gruber, P. Johnson, and A. C. Myers. *Theta Reference Manual*. MIT Laboratory for Computer Science, Cambridge, MA, February 1994. Programming Methodology Group Memo 88, available from www.pmg.lcs.mit.edu/papers/thetaref/.
- [LD00] G. T. Leavens and K. K. Dhara. Concepts of behavioral subtyping and a sketch of their extension to component-based systems. In G. T. Leavens and M. Sitaraman, editors, *Foundations of Component-Based Systems*. Cambridge University Press, 2000.
- [Lea88] G. T. Leavens. *Verifying Object-Oriented Programs that use Subtypes*. PhD thesis, Massachusetts Institute of Technology, 1988. Published as MIT/LCS/TR-439 in February 1989.
- [Lea96] G. T. Leavens. An overview of Larch/C++: Behavioral specifications for C++ modules. In H. Kilov and W. Harvey, editors, *Specification of Behavioral Semantics in Object-Oriented Information Modeling*, chapter 8, pages 121–142. Kluwer Academic Publishers, Boston, 1996.
- [Lea97] G. T. Leavens. Larch/C++ reference manual. HTML version available from www.cs.iastate.edu/~leavens/larchc++manual/lcpp_toc.html, July 1997.
- [Lei95a] K. R. M. Leino. A myth in the modular specification of programs. Note KRML 63-0, 1995.

- [Lei95b] K. R. M. Leino. *Toward Reliable Modular Programs*. PhD thesis, California Institute of Technology, 1995.
- [Lei97] K. R. M. Leino. Ecstatic: An object-oriented programming language with an axiomatic semantics. In B. Pierce, editor, *Proceedings of the Fourth International Workshop on Foundations of Object-Oriented Languages*, 1997. Available from: www.cs.indiana.edu/hyplan/pierce/fool/.
- [Lei98] K. R. 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(10) of *ACM SIGPLAN Notices*, pages 144–153, October 1998.
- [LG86] B. Liskov and J. Guttag. *Abstraction and Specification in Program Development*. MIT Press, 1986.
- [LH92] K. Lano and H. Haughton. Reasoning and refinement in object-oriented specification languages. In O. L. Madsen, editor, *ECOOP '92 European Conference on Object-Oriented Programming*, volume 615 of *Lecture Notes in Computer Science*, pages 78–97. Springer-Verlag, 1992.
- [LMMPH00] M. Labeth, J. Meyer, P. Müller, and A. Poetzsch-Heffter. Formal verification of a doubly linked list implementation: A case study using the JIVE system. Technical Report 270, Fernuniversität Hagen, 2000.
- [LN97] K. R. M. Leino and G. Nelson. Abstraction and specification revisited. A revised version of this manuscript is available as technical report [LN00], 1997.
- [LN00] K. R. M. Leino and G. Nelson. Data abstraction and information hiding. Technical Report 160, Compaq Systems Research Center, 2000.
- [LS97a] K. R. M. Leino and R. Stata. Checking object invariants. Technical Report 1997-007, Digital Systems Research Center, January 1997.
- [LS97b] K. R. M. Leino and R. Stata. Direct dependencies and the pivot visibility rule. Note KRML 69-0, 1997.
- [LS99] K. R. M. Leino and R. Stata. Virginité: A contribution to the specification of object-oriented software. *Information Processing Letters*, 70(2):99–105, April 1999.
- [LSS99] K. R. M. Leino, J. B. Saxe, and R. Stata. Checking Java programs via guarded commands. In B. Jacobs, G. T. Leavens, P. Müller, and A. Poetzsch-Heffter, editors, *Formal Techniques for Java Programs*. Technical Report 251, Fernuniversität Hagen, 1999. Available from www.informatik.fernuni-hagen.de/pi5/publications.html.
- [Luc90] D. C. Luckham. *Programming with Specifications: An Introduction to Anna. A Language for Specifying Ada Programs*. Springer-Verlag, 1990.
- [LW90] G. T. Leavens and W. E. Weihl. Reasoning about object-oriented programs that use subtypes (extended abstract). In N. Meyrowitz, editor, *OOPSLA ECOOP '90 Proceedings*, volume 25(10) of *ACM SIGPLAN Notices*, pages 212–223. ACM, October 1990.
- [LW93] B. Liskov and J. M. Wing. Specifications and their use in defining subtypes. In A. Paepcke, editor, *Proceedings of the 1998 ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '93)*, volume 28 of *ACM SIGPLAN Notices*, pages 16–28. ACM Press, 1993.

- [LW94] B. Liskov and J. M. Wing. A behavioral notion of subtyping. *ACM Transactions on Programming Languages and Systems*, 16(6), 1994.
- [LW97] G. T. Leavens and J. M. Wing. Protective interface specifications. In M. Bidoit and M. Dauchet, editors, *TAPSOFT '97: Theory and Practice of Software Development, 7th International Joint Conference CAAP/FASE, Lille, France*, volume 1214 of *Lecture Notes in Computer Science*, pages 520–534. Springer-Verlag, 1997. Available from <ftp://ftp.cs.iastate.edu/pub/techreports/TR96-04/TR.ps.gz>.
- [Mey86] B. Meyer. Genericity is versus inheritance. In *OOPSLA '86 Conference Proceedings*, volume 21 of *ACM SIGPLAN Notices*, 1986.
- [Mey88] B. Meyer. *Object-Oriented Software Construction*. Prentice Hall, 1988.
- [Mey92a] B. Meyer. Design by contract. In D. Mandrioli and B. Meyer, editors, *Advances in object-oriented software engineering*. Prentice Hall, 1992.
- [Mey92b] B. Meyer. *Eiffel: The Language*. Prentice Hall, 1992.
- [Mey02] J. Meyer. *Design and Implementation of Interactive Program Verification Tools*. PhD thesis, Fernuniversität Hagen, 2002. (to appear).
- [MH69] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Melzter and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.
- [Min96] N. Minsky. Towards alias-free pointers. In P. Cointe, editor, *ECOOP '96 European Conference on Object-Oriented Programming*, volume 1098 of *Lecture Notes in Computer Science*, pages 189–209. Springer-Verlag, 1996.
- [MMPH97] P. Müller, J. Meyer, and A. Poetzsch-Heffter. Programming and interface specification language of JIVE — specification and design rationale. Technical Report 223, Fernuniversität Hagen, 1997.
- [MMPH99] P. Müller, J. Meyer, and A. Poetzsch-Heffter. Making executable interface specifications more expressive. In C. H. Cap, editor, *JIT '99 Java-Informationen-Tage 1999*, Informatik Aktuell. Springer-Verlag, 1999. Available from www.informatik.fernuni-hagen.de/pi5/publications.html.
- [MMPH00] J. Meyer, P. Müller, and A. Poetzsch-Heffter. The JIVE system—implementation description. Available from www.informatik.fernuni-hagen.de/pi5/publications.html, 2000.
- [MMPN93] O. L. Madsen, B. Møller-Pedersen, and K. Nygaard. *Object-Oriented Programming in the BETA Programming Language*. Addison-Wesley, 1993.
- [Mor94] C. Morgan. *Programming from Specifications*. Prentice Hall, 1994.
- [MPH97a] P. Müller and A. Poetzsch-Heffter. Formal specification techniques for object-oriented programs. In M. Jarke, K. Pasedach, and K. Pohl, editors, *Informatik 97: Informatik als Innovationsmotor*, Informatik Aktuell. Springer-Verlag, 1997.
- [MPH97b] P. Müller and A. Poetzsch-Heffter. Preserving the correctness of object-oriented programs under extension. In R. Berghammer and F. Simon, editors, *Programming Languages and Fundamentals of Programming*. Christian-Albrechts-Universität Kiel, 1997. Technical Report 9717.

- [MPH98] P. Müller and A. Poetzsch-Heffter. Kapselung und Methodenbindung: Javas Designprobleme und ihre Korrektur. In C. H. Cap, editor, *JIT '98 Java-Informationen-Tage 1998*, Informatik Aktuell. Springer-Verlag, 1998. Available from www.informatik.fernuni-hagen.de/pi5/publications.html (in German).
- [MPH99a] P. Müller and A. Poetzsch-Heffter. Alias control is crucial for modular verification. In A. Moreira and D. Demeyer, editors, *Object-Oriented Technology. ECOOP'99 Workshop Reader*, volume 1743 of *Lecture Notes in Computer Science*, pages 154–156. Springer-Verlag, 1999. (position paper).
- [MPH99b] P. Müller and A. Poetzsch-Heffter. Universes: A type system for controlling representation exposure. In A. Poetzsch-Heffter and J. Meyer, editors, *Programming Languages and Fundamentals of Programming*. Fernuniversität Hagen, 1999. Technical Report 263, URL: www.informatik.fernuni-hagen.de/pi5/publications.html.
- [MPH00a] J. Meyer and A. Poetzsch-Heffter. An architecture for interactive program provers. In S. Graf and M. Schwartzbach, editors, *Tools and Algorithms for the Construction and Analysis of Software (TACAS)*, volume 276 of *Lecture Notes in Computer Science*, pages 63–77, 2000.
- [MPH00b] P. Müller and A. Poetzsch-Heffter. Modular specification and verification techniques for object-oriented software components. In G. T. Leavens and M. Sitaraman, editors, *Foundations of Component-Based Systems*. Cambridge University Press, 2000.
- [MPH00c] P. Müller and A. Poetzsch-Heffter. A type system for controlling representation exposure in Java. In S. Drossopoulou, S. Eisenbach, B. Jacobs, G. T. Leavens, P. Müller, and A. Poetzsch-Heffter, editors, *Formal Techniques for Java Programs*. Technical Report 269, Fernuniversität Hagen, 2000. Available from www.informatik.fernuni-hagen.de/pi5/publications.html.
- [MS96] D. R. Musser and A. Saini. *STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library*. Addison-Wesley, 1996.
- [MTHM97] R. Milner, M. Tofte, R. Harper, and D. MacQueen. *The Definition of Standard ML (revised)*. MIT Press, 1997.
- [Mül95] P. Müller. Specification and implementation of an annotation language for an object-oriented programming language. Master's thesis, Technische Universität München, 1995. (In German).
- [MZW97] A. Moormann Zaremski and J. M. Wing. Specification matching software components. *ACM Transactions on Software Engineering and Methodology*, 1997.
- [Nel91] G. Nelson, editor. *Systems Programming with Modula-3*. Prentice Hall, 1991. Current version of language definition available from www.research.digital.com/SRC/m3defn/html/m3.html
- [NO98] T. Nipkow and D. von Oheimb. Java^{light} is type-safe — definitely. In *Proc. 25th ACM Symp. Principles of Programming Languages*, pages 161–170. ACM Press, New York, 1998.
- [NVLA99] J. Noble, J. Vitek, D. Lea, and P. S. Almeida. Aliasing in object oriented systems. In A. Moreira and D. Demeyer, editors, *Object-Oriented Technology. ECOOP'99 Workshop Reader*, volume 1743 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

- [NVP98] J. Noble, J. Vitek, and J. M. Potter. Flexible alias protection. In E. Jul, editor, *ECOOP '98: Object-Oriented Programming*, volume 1445 of *Lecture Notes in Computer Science*. Springer-Verlag, 1998.
- [OCL] OCL frequently asked questions. Archive at www.cs.ukc.ac.uk/research/sse/oclws2k/oclfaq.txt.
- [Ohe99] D. von Oheimb. Hoare logic for mutual recursion and local variables. In C. Pandu Rangan, V. Raman, and R. Ramanujam, editors, *Foundations of Software Technology and Theoretical Computer Science*, volume 1738 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.
- [Ohe00] D. von Oheimb. Axiomatic semantics for Java_{light}. In S. Drossopoulou, S. Eisenbach, B. Jacobs, G. T. Leavens, P. Müller, and A. Poetzsch-Heffter, editors, *Formal Techniques for Java Programs*. Technical Report 269, Fernuniversität Hagen, 2000. Available from www.informatik.fernuni-hagen.de/pi5/publications.html.
- [OMG] OMG. UML resource page. Available from www.omg.org/uml/.
- [Omo94] S. M. Omohundro. The Sather 1.0 specification. Technical report, International Computer Science Institute, 1994.
- [ON98] D. von Oheimb and T. Nipkow. Machine-checking the Java specification: Proving type-safety. In J. Alves-Foss, editor, *Formal Syntax and Semantics of Java*, volume 1523 of *Lecture Notes in Computer Science*. Springer, 1998.
- [OSR93] S. Owre, N. Shankar, and J. M. Rushby. The PVS specification language (beta release). Technical report, Computer Science Laboratory SRI International, April 1993.
- [OW97] M. Odersky and P. Wadler. Pizza into Java: Translating theory into practice. In *The 24th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. ACM Press, 1997.
- [Owi75] S. Owicki. *Axiomatic Proof Techniques for Parallel Programs*. Tr-75-251, Comp. Science Dept., Cornell University, 1975.
- [Par72] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 5(12):1053–1058, December 1972. Reprinted in [You79].
- [Pau91] L. C. Paulson. *ML for the working Programmer*. Cambridge University Press, 1991.
- [Pau94] L. C. Paulson. *Isabelle: A Generic Theorem Prover*, volume 828 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.
- [Per90] D. Perrin. Finite automata. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 1, pages 1–57. Elsevier Science Publishers, 1990.
- [PH97a] A. Poetzsch-Heffter. Prototyping realistic programming languages based on formal specifications. *Acta Informatica*, 34:737–772, 1997.
- [PH97b] A. Poetzsch-Heffter. Specification and verification of object-oriented programs. Habilitation thesis, Technical University of Munich, Jan. 1997. URL: www.informatik.fernuni-hagen.de/pi5/publications.html.
- [PH00] A. Poetzsch-Heffter. *Konzepte objektorientierter Programmierung*. Springer-Verlag, 2000.
- [PHM98] A. Poetzsch-Heffter and P. Müller. Logical foundations for typed object-oriented languages. In D. Gries and W. De Roever, editors, *Programming Concepts and Methods (PROCOMET)*, 1998.

- [PHM99] A. Poetzsch-Heffter and P. Müller. A programming logic for sequential Java. In S. D. Swierstra, editor, *Programming Languages and Systems (ESOP '99)*, volume 1576 of *Lecture Notes in Computer Science*, pages 162–176. Springer-Verlag, 1999.
- [Pre97] C. Prehofer. Feature-oriented programming: A fresh look at objects. In M. Akşit and S. Matsuoka, editors, *ECOOP '97: Object-Oriented Programming*, volume 1241 of *Lecture Notes in Computer Science*, pages 32–59. Springer-Verlag, 1997.
- [Rei95] W. Reif. The KIV approach to software verification. In M. Broy and S. Jähnichen, editors, *KORSO: Methods, Languages, and Tools for the Construction of Correct Software*, volume 1009 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.
- [RL00] C. Ruby and G. T. Leavens. Safely creating correct subclasses without seeing superclass code. In *OOPSLA 2000 Conference on Object-Oriented Programming, Systems, Languages, and Applications*, volume 35(10) of *ACM SIGPLAN Notices*, pages 208–228, October 2000.
- [Ros97] J. Rose. *Inner Classes Specification*. Sun Microsystems, Inc., 1997. Available from <http://java.sun.com/products/jdk/1.1/docs>.
- [RS92] L. Rapanotti and A. Socorro. Introducing FOOPS. Technical Report PRG-TR-28-92, Oxford University Computing Laboratory, 1992.
- [RS93] W. Reif and K. Stenzel. Reuse of proofs in software verification. In R. Shyamasundar, editor, *Foundation of Software Technology and Theoretical Computer Science*, volume 761 of *Lecture Notes in Computer Science*, pages 284–293. Springer-Verlag, 1993.
- [RSSB98] W. Reif, G. Schellhorn, K. Stenzel, and M. Balsler. Structured specifications and interactive proofs with KIV. In W. Bibel and P. Schmitt, editors, *Automated Deduction — A Basis for Applications*. Kluwer Academic Publishers, 1998.
- [Rüp94] A. Rüping. Modules in object-oriented systems. In R. Ege, M. Singh, and B. Meyer, editors, *TOOLS 14 — Technology of Object-Oriented Languages and Systems*. Prentice Hall, 1994.
- [RW92] M. Reiser and N. Wirth, editors. *Programming in Oberon*. ACM Press, 1992.
- [She95] D. Sheppard. *An Introduction to Formal Specification with Z and VDM*. McGraw-Hill, 1995.
- [SMC74] W. Stevens, G. Myers, and L. Constantine. Structured design. *IBM Systems Journal*, 13(2):115–139, May 1974. Reprinted in [You79].
- [Sny86] A. Snyder. Encapsulation and inheritance in object-oriented programming languages. In *OOPSLA '86 Conference Proceedings*, volume 21 of *ACM SIGPLAN Notices*, pages 38–45, 1986.
- [Sny87] A. Snyder. Inheritance and the development of encapsulated software systems. In B. Shriver and P. Wegner, editors, *Research Directions in Object-Oriented Programming*, pages 165–188. MIT Press, 1987.
- [SOM94] C. A. Szypersky, S. Omohundro, and S. Murer. Engineering a programming language: The type and class system of Sather. In J. Gutknecht, editor, *Programming Languages and System Architectures*, volume 782 of *Lecture Notes in Computer Science*, pages 208–227. Springer-Verlag, 1994.
- [ST88] D. Sannella and A. Tarlecki. Specifications in an arbitrary institution. *Information and Computation*, 76:165–210, 1988.

- [Ste96] B. Steensgaard. Points-to analysis in almost linear time. In *Proc. 23rd ACM Symp. Principles of Programming Languages*, pages 32–41. ACM Press, 1996.
- [Str91] B. Stroustrup, editor. *The C++ Programming Language, 2nd Edition*. Addison-Wesley, 1991.
- [Suz80] N. Suzuki, editor. *Automatic Verification of Programs with Complex Data Structures*. Garland Publishing, 1980.
- [SWO95] M. Sitaraman, B. W. Weide, and W. F. Ogden. Using abstraction relations to verify abstract data type representations. Technical Report OSU-CISRC-9/95-TR39, Ohio State University, September 1995.
- [Szy92] C. A. Szypersky. Import is not inheritance — why we need both: Modules and classes. In O. L. Madsen, editor, *ECOOP '92 European Conference on Object-Oriented Programming*, volume 615 of *Lecture Notes in Computer Science*, pages 19–32. Springer-Verlag, 1992.
- [Szy98] C. A. Szypersky. *Component Software — Beyond Object-Oriented Programming*. Addison-Wesley, 1998.
- [TWW82] J. W. Thatcher, E. G. Wagner, and J. B. Wright. Data type specification: parameterization and the power of specification techniques. *ACM TOPLAS*, 4:711–773, 1982.
- [Ull94] J. D. Ullman. *Elements of ML Programming*. Prentice-Hall, 1994.
- [UR93] M. Utting and K. Robinson. Modular reasoning in an object-oriented refinement calculus. In R. S. Bird, C. C. Morgan, and J. C. P. Woodcock, editors, *Mathematics of Program Construction*, volume 669 of *Lecture Notes in Computer Science*, pages 344–367. Springer-Verlag, 1993.
- [Ver01] The Verificard project. www.verificard.org, 2001.
- [Wad90] P. Wadler. Linear types can change the world! In M. Broy and C. B. Jones, editors, *Programming Concepts and Methods (PROCOMET)*, 1990.
- [WE87] J. Welsh and J. Elder. *Introduction to Modula-2*. Prentice Hall, 1987.
- [WGSD89] M. Woodman, R. Griffiths, J. Souter, and M. Davies. *Portable Modula-2 Programming*. McGraw-Hill, 1989.
- [Win83] J. M. Wing. A two-tiered approach to specifying programs. Technical Report TR-299, Massachusetts Institute of Technology, Laboratory for Computer Science, 1983.
- [Win87] J. M. Wing. Writing Larch interface language specifications. *ACM Transactions on Programming Languages and Systems*, 9(1):1–24, January 1987.
- [Wir82] M. Wirsing. Structured algebraic specifications. In B. Robinet, editor, *Proc. AFCET Symp. on Mathematics for Computer Science*, pages 93–108, 1982.
- [Wir88] N. Wirth. From Modula to Oberon. *Software Practice and Experience*, 18(7), 1988.
- [Wir90] M. Wirsing. Algebraic specification. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 13, pages 675–788. Elsevier Science Publishers, 1990.
- [Wir96] N. Wirth. *Compiler Construction*. Addison-Wesley, 1996.
- [WK99] J. Warmer and A. Kleppe. *The Object Constraint Language, precise modeling with UML*. Addison-Wesley, 1999.
- [WPP⁺83] M. Wirsing, P. Pepper, H. Partsch, W. Dosch, and M. Broy. On hierarchies of abstract data types. *Acta Informatica*, 20:1–33, 1983.

- [WSR00] R. Wilhelm, M. Sagiv, and T. Reps. Shape analysis. In D. A. Watt, editor, *Compiler Construction*, volume 1781 of *Lecture Notes in Computer Science*, pages 1–17. Springer-Verlag, 2000.
- [XdRH97] Q. Xu, W.-P. de Roever, and J. He. The rely-guarantee method for verifying shared variable concurrent programs. *Formal Aspects of Computing*, 9(2):149–174, 1997.
- [XP99] H. Xi and F. Pfenning. Dependent types in practical programming. In *Proc. 26th ACM Symp. Principles of Programming Languages*, pages 214–227. ACM Press, New York, 1999.
- [XS98] Q. Xu and M. Swarup. Compositional reasoning using the assumption-commitment paradigm. In W.-P. de Roever, H. Langmaack, and A. Pnueli, editors, *Compositionality: The Significant Difference*, volume 1536 of *Lecture Notes in Computer Science*, pages 565–583. Springer-Verlag, 1998.
- [You79] E. N. Yourdon. *Classics in Software Engineering*. Yourdon Press, 1979.

List of Figures

2.1	Abstract Syntax of Mojave	41
2.2	Object Store with Ownership Relation	53
2.3	Object Structure for a Doubly Linked Integer List	55
2.4	Object Structure for a Doubly Linked Object List with Iterator ..	58
2.5	Object Structure for a Doubly Linked List with Iterator and Position.	62
2.6	Implementation of <code>equalsList</code>	63
2.7	Implementation of Property Pattern	65
2.8	Object Structure for a <code>MyBean</code> -Object	65
2.9	Example for Type Combinator	66
3.1	Conceptual Theory Structure for One Class	79
3.2	Theory Structure for Two Modules	80
3.3	Language-Independent Axioms and Rules.....	98
5.1	Non-local Dependency	149
5.2	Case 1 for Field Updates	150
5.3	Case 2 for Field Updates	150
5.4	Case 1 for Method Invocations	152
5.5	Case 2 for Method Invocations	153
5.6	Case 3 for Method Invocations	153
5.7	Automaton for Example 5.2.1	163
5.8	Automaton for Module <code>LIST</code>	167

Index

- \mathcal{L} 247
- \mathcal{U} 84
- \mathfrak{A} 162
- Γ 92
- $\Gamma_{body(m)}$ 92
- Γ_{post} 92
- $\Gamma_{pre(m)}$ 92
- Σ -formula 223
- δ 176
- ρ 91
- σ 162
- τ 94
- $:$ 43, 86
- @ 43, 86
- *_ 66
- \mathbb{R} - 68
- λ - 85
- λ - 85
- λ_M - 81
- λ_T - 81
- \triangleleft - 83
- \triangleleft - 83
- $(-)$ 90
- $\langle - := - \rangle$ 90
- $\langle -, - \rangle$ 90
- \rightarrow - 157
- \equiv - 127

- aB* 84
- Abstract field 9, 126
- Abstraction 125
 - explicit 126
 - implicit 126
- Abstraction function 126
- Abstraction function 126
- Abstraction relation 126
- Access mode 18, 48
- Accessibility property 178
- accessible* 83
- accessibleL* 89
- AccessMode* 41

- accessMode* 87
- Adaptability 19
- AFieldId* 86
- aI* 84
- Alias
 - dynamic 14
 - static 14
- Alias control 53
- alive* 90
- All-rule 98
- ALocation* 88
- Ancestor 30
- Annotation
 - method 93
 - statement 93
- Antecedent 93
- Assumpt-axiom 98
- Assumpt-elim-rule 98
- Assumpt-intro-rule 98
- Assumption 93
- Authenticity requirement 154
- Axiom
 - assumpt- 98
 - cast- 94
 - **dep1** 157
 - **dep2** 157
 - **dep3** 157
 - false- 98
 - field-read- 94
 - field-write- 94
 - **field1** 87
 - **field2** 87
 - **field3** 87
 - **field4** 88
 - **field5** 88
 - **field6** 88
 - **field7** 88
 - **field8** 88
 - **field9** 88
 - **import1** 82

- **import2** 82
- **import3** 82
- **import4** 82
- **import5** 82
- **import6** 82
- inclusive 82
- new- 94
- skip- 94
- **store1** 90
- **store2** 90
- **store3** 90
- **store4** 90
- **store5** 90
- **store6** 91
- **store7** 91
- **store8** 91
- **store9** 91
- **store10** 91
- **store11** 91
- **store12** 91
- **store13** 91
- **subM1** 81
- **subM2** 81

- Behavior
 - functional 123
- Behavioral interface specification 4
- Behavioral subtyping 136
- Belong to 54, 89
- body* 93
- booleanDT* 84
- booleanT* 41

- Call-rule 95
- Call-var-rule 96
- Cast-axiom 94
- CFieldId* 86
- Child universe 30, 54
- cid* 41
- cidV* 84
- Class-rule 96
- ClassId* 41
- Client 50
- Client code 3
- Client interface 18
- CLocation* 88
- Closed program 44
- Co-dependency 13
- col* 129
- Combination 19
- Component 2
- Component-based development 2
- Composability 19

- Conjunct-rule 98
- Consequent 93
- Core 45
- Correctness
 - closed programs 110
 - modular 111
- Cover 13
- ctid* 41
- Current universe 59
- cut* 129

- Data abstraction 125
- dc* 156
- Declaration type 43, 87
- Def-clause 128
- Defines-clause 128
- Definition 3.2.1 99
- Definition 3.2.2 100
- Definition 3.2.3 100
- Definition 3.2.4 108
- Definition 3.2.5 109
- Definition 3.3.1 110
- Definition 3.3.2 111
- Definition 5.4.1 180
- Definition 5.4.2 183
- dep* 266
- dep1** 157
- dep2** 157
- dep3** 157
- Dependee 144
- Dependency
 - dynamic 30
 - static 30
- Dependent 144
- Depends 9
- Depends-clause 156
- Depends-relation 147
- Descendant 30
- Desugaring of modifies-clauses 176
- Disjunct-rule 98
- Downward closure 176
- dtype* 87
- dyn* 85
- Dynamic dependency 30
- Dynamic alias 14
- Dynamic component 52
- Dynamic type 54, 84
- Dynamization 85
- DynType* 84

- Encapsulation 18, 47
 - representation 55
- ex-rule 98

- Example 1.3.1 7
- Example 1.3.2 8
- Example 1.3.3 11
- Example 1.3.4 13
- Example 1.4.1 24
- Example 3.3.1 113
- Example 5.1.1 145
- Example 5.1.2 154
- Example 5.2.1 161
- Example 5.2.2 172
- Example 5.2.3 175
- Example 6.1.1 197
- Example 6.4.1 208
- Explicit abstraction 126
- Extended state problem 11
- Extended state 6, 12

- False-axiom 98
- fid* 89
- Field
 - abstract 9, 126
 - Field-read-axiom 94
 - Field-write-axiom 94
 - field1** 87
 - field2** 87
 - field3** 87
 - field4** 88
 - field5** 88
 - field6** 88
 - field7** 88
 - field8** 88
 - field9** 88
 - FieldId* 86
 - Frame problem 143
 - Frame properties 3, 143
 - Friend 18
 - Functional behavior 123

 - grndT* 41
 - Ground type 42
 - Guard 153
 - guard* 159

 - H* 162
 - h* 237
 - Helper method 196
 - History constraint 208
 - Hoare lemma 27
 - Hoare triple 93

 - If-rule 94
 - iid* 41
 - impl* 93
 - Implementation-rule 96

 - Implicit abstraction 126
 - ImplId* 41, 43
 - Import 40
 - import1** 82
 - import2** 82
 - import3** 82
 - import4** 82
 - import5** 82
 - import6** 82
 - imports* 82
 - Inclusive axiom 82
 - index* 131
 - Information hiding 47
 - init* 85
 - Inside 53
 - intDT* 84
 - Interface 50
 - client 18
 - specialization 18
 - Interface specification 4
 - Interface object 52
 - Interface type 50
 - InterfaceId* 41
 - intT* 41
 - inv* 204
 - Inv-rule 98
 - Invariant
 - module 170, 208
 - program 208
 - relevant 200
 - invL* 202
 - Invocation-rule 95
 - Invocation-var-rule 95
 - invrep* 202
 - invSF* 202
 - itid* 41

 - JIVE 217

 - l* 247
 - L-equivalence 127
 - lal* 108
 - Language property operator 99
 - Lemma
 - Hoare 27
 - Lemma 3.1.1 82
 - Lemma 3.1.2 83
 - Lemma 3.1.3 84
 - Lemma 3.1.4 85
 - Lemma 3.1.5 85
 - Lemma 3.1.6 86
 - Lemma 3.1.7 86
 - Lemma 3.1.8 89

- Lemma 3.1.9 89
- Lemma 3.1.10 91
- Lemma 3.2.1 101
- Lemma 3.2.2 101
- Lemma 3.2.3 101
- Lemma 3.2.4 109
- Lemma 3.2.5 109
- Lemma 3.3.1 111
- Lemma 3.3.2 116
- Lemma 4.2.1 127
- Lemma 4.2.2 129
- Lemma 5.2.1 158
- Lemma 5.2.2 158
- Lemma 5.2.3 159
- Lemma 5.2.4 160
- Lemma 5.2.5 168
- Lemma 5.2.6 168
- Lemma 5.4.1 180
- Lemma 5.4.2 181
- Lemma 5.4.3 183
- Lemma 6.2.1 203
- Lemma 6.3.1 206
- Lemma D.1.1 237
- Lemma D.1.2 238
- Lemma D.1.3 242
- Lemma D.2.1 243
- Lemma D.2.2 244
- Lemma D.2.3 244
- Lemma D.2.4 246
- Lemma D.2.5 247
- Lemma D.3.1 261
- Lemma D.4.1 266
- Lemma D.4.2 266
- Lemma D.4.3 269
- list of* 224
- locA* 89
- Local creation property 180
- Local update property 180
- Locality 148
- Locality requirement 149
- Locality rule 149
- Location 88
 - relevant 146
- Location* 88
- locC* 89
- Logical variable 223
- Lopex 217
- lrtype* 89

- Main module 45
- Method annotation 93
- Method implementation 43
- mkAFieldId* 86
- mkCFieldId* 86
- ModId* 41
- Modifies-clause 176
 - desugaring 176
- Modular correctness 111
- Modular development 2
- Modular soundness 27, 112
- Modularity theorem 154, 179
- Module 46
 - Module* 41
 - module* 82
- Module invariant 170, 208
- Mojave 39

- new* 90
- New-axiom 94
- Notdepends-relation 148
- Null type 42
- nullDT* 84
- nullT* 41

- oal* 108
- obj* 89
- Object type 87
- Object universe 30, 54
- ObjId* 84
- objU* 83
- Obligation 5.1 157
- Obligation 5.2 159
- Obligation 5.3 160
- Obligation 5.4 160
- Open program 44
- orep type 42
- orepT* 41
- otype* 87
- Outside 53
- Owner 54
- Ownership model 52
- Ownership relation 52

- Package 46
- par* 68
- Pivot location 169
- Postcondition 93
- Poststate 92
- Pre-post-pair 131
- Pre-post-specification 131
- Precondition 93
- Prelude theory 225
- Present 89
- presentL* 89
- Prestate 92
- Private protected 49
- Program

- closed 44
- open 44
- Program invariant 208
- Program component 93
- Program element 17
- Proof outline 225
- Proper subtype 81
- Property editor 64
- Public theory 78

- $R_{A,B}$ 163
- $R_{A,B}^N$ 163
- Range type 87
- rc* 164
- Reachability 91
- Readonly type 42
- ref* 84
- refDT* 84
- Reference type 42
- regExpr* 164
- Relevant invariant 200
- Relevant location 146
- rep type 59
- Representation 52, 128
- Representation encapsulation 55
- Representation containment 52
- Req-clause 131
- Requires-clause 131
- ret* 68
- roDT* 84
- roT* 41
- rtype* 87
- Rule
 - all- 98
 - assumpt-elim- 98
 - assumpt-intro- 98
 - call- 95
 - call-var- 96
 - class- 96
 - conjunct- 98
 - disjunct- 98
 - if- 94
 - implementation- 96
 - inv- 98
 - invocation- 95
 - invocation-var- 95
 - seq- 94
 - static-invocation- 95
 - static-invocation-var- 95
 - strength- 98
 - subst- 98
 - subtype- 96
 - weak- 98
 - while- 94
- rule
 - ex- 98

- safid* 86
- scfid* 86
- Scope 47
- Seq-rule 94
- Sequent 93
- set of* 224
- Signature 223
- SimpleAFieldId* 86
- SimpleCFieldId* 41, 86
- Skip-axiom 94
- Software component 2
- Sort 223
- Soundness
 - modular 27, 112
- Specialization interface 18
- Specification
 - interface 4
 - pre-post- 131
 - universal 4
 - well-formed 124
- Standard universe 54
- State 92
 - visible 196
- Statement annotation 93
- static* 83
- Static dependency 30
- Static alias 14
- Static-invocation-rule 95
- Static-invocation-var-rule 95
- stdU* 83
- Store* 90
- store1** 90
- store2** 90
- store3** 90
- store4** 90
- store5** 90
- store6** 91
- store7** 91
- store8** 91
- store9** 91
- store10** 91
- store11** 91
- store12** 91
- store13** 91
- Strength-rule 98
- subM1** 81
- subM2** 81
- Subst-rule 98
- Subtype 81

- proper 81
- Subtype-rule 96
- Succedent 93
- SVENJA 217
- Synthesis 19

- TA 134
- Target type 44
- Theorem 5.4.1 184
- Theory 22
 - prelude 225
 - public 78
- tid* 41
- tidD* 84
- trep type 42
- trepT* 41
- Triple 93
- Type 54
 - dynamic 54, 84
 - ground 42
 - null 42
 - orep 42
 - readonly 42
 - reference 42
 - rep 59
 - trep 42
- Type* 41
- Type universe 30
- Type combinator 66
- Type declaration 54
- Type universe 54, 56

- TypeDecl* 41
- TypeId* 41
- typeof* 85
- typeU* 83

- Unique variable 76
- Uniqueness 76
- univ* 84
- Universal specification 4
- Universe 30, 54
 - child 30, 54
 - current 59
 - object 30, 54
 - standard 54
 - type 30, 54, 56
- Universe* 83
- univV* 84

- Value* 84
- Variable
 - logical 223
- Verificard 219
- View 50
- Virtual method 43
- VirtualMethodId* 41, 43
- Visibility requirement 155
- Visible state 196

- Weak-rule 98
- Well-formed specification 124
- While-rule 94
- wt* 100