

# Bibliography

- [1] HOL home page. <http://lal.cs.byu.edu/lal/hol-documentation.html>.
- [2] Isabelle home page. <http://www.cl.cam.ac.uk/Research/HVG/Isabelle/index.html>.
- [3] CARE home page. <http://svrc.it.uq.edu.au/CARE>.
- [4] S.P. Arnold and S.L. Stepoway. The reuse system: Cataloging and retrieval of reusable software. In Will Tracz, editor, *Tutorial: Software Reuse Emerging Technology*, pages 376–379. IEEE Computer Society Press, 1990.
- [5] S. Atkinson. A unifying model for retrieval from reusable software libraries. Technical Report 95-41, Software Verification Research Centre, The University of Queensland, December 1995.
- [6] R.J.R. Back. Data Refinement in the Refinement Calculus. Unpublished report, 1988.
- [7] J. Barnes, editor. *Ada 95 Rationale*, volume 1247 of *Lecture Notes in Computer Science*. Springer, January 1995.
- [8] V.R. Basili, L.C. Briand, and W.L. Melo. How reuse influences productivity in object-oriented systems. *Communications of the ACM*, 39(10):104–116, October 1996.
- [9] J.C. Bicarregui, J.S. Fitzgerald, P.A. Lindsay, R. Moore, and B. Ritchie. *Proof in VDM: A Practitioner's Guide*. FACIT Series. Springer-Verlag, 1994. ISBN no. 3-540-19813-X.

- 
- [10] T.J. Biggerstaff. A perspective of generative reuse. *Annals of Software Engineering*, 5:169–226, 1998.
- [11] T.J. Biggerstaff and A.J. Perlis, editors. *Software Reusability: Concepts and Models*, volume 1. ACM press, New York, 1989.
- [12] Grady Booch. *Object Oriented Design: with applications*. Benjamin Cummings, California, 1991.
- [13] Alan W. Brown, editor. *Component-Based Software Engineering*. IEEE Computer Society, Carnegie Mellon University, Software Engineering Institute, 1996.
- [14] B.A. Burton, R.W. Aragon, S.A. Bailey, K.D. Koehler, and L.A. Mayes. The reusable software library. *IEEE Software*, 4:25–33, July 1987.
- [15] D. Carrington, D. Duke, R. Duke, P. King, G. A. Rose, and G. Smith. Object-Z: An object-oriented extension to Z. In S. Vuong, editor, *Formal Description Techniques, II (FORTE'89)*, pages 281–296. Elsevier Science Publishers (North-Holland), 1990.
- [16] F. Collis and K. Harwood. *User's Guide to the CARE Finder*. Teletronics Pacing Systems, 1995.
- [17] J.L. Cybulski, R.D. Neal, A. Kram, and J.C. Allen. Reuse of early life-cycle artifacts: workproducts, methods and tools. *Annals of Software Engineering*, 5:227–252, 1998.
- [18] J. Davies. *Specification and Proof in Real-time CSP*. Distinguished Dissertations in Computer Science. Cambridge University Press, 1993.
- [19] J. Dawes. *The VDM-SL Reference Guide*. Pitman, 1991.
- [20] G. Dowek. Third order matching is decidable. *Annals of Pure and Applied Logic*, 69:135–155, 1994.

- [21] R. Duke, G. Rose, and G. Smith. Object-Z: a specification language advocated for the description of standards. Technical Report 94-45, Software Verification Research Centre, 1994.
- [22] M.F. Dunn and J.C. Knight. Software reuse in an industrial setting: A case study. In *Proceedings of the Thirteenth International Conference on Software Engineering*, pages 329–338, Austin, TX, 1991.
- [23] R. Ellis. *Data Abstraction and Program Design*. Pitman, 1991.
- [24] D. Fafchamps. Organizational factors and reuse. *IEEE Software*, pages 31–41, September 1994.
- [25] B. Fischer, F. Kievernagel, and W. Struckman. VCR: A VDM-based software component retrieval tool. Technical Report 94-08, Technical University of Braunschweig, Germany, November 1994.
- [26] W. Frakes and C. Terry. Software reuse: Metrics and models. *ACM Computing Surveys*, 28(2):415–435, June 1996.
- [27] W.B. Frakes and B.A. Nejme. An information system for software reuse. In Will Tracz, editor, *Tutorial: Software Reuse Emerging Technology*. IEEE Computer Society Press, 1990.
- [28] R.J. Gautier and P.J.L. Wallis, editors. *Software reuse with Ada*, volume 16 of *IEE Computing Series*. Peter Peregrinus Ltd, London, 1990.
- [29] J.A. Goguen. Principles of parameterized programming. In T.J. Biggerstaff and A.J. Perlis, editors, *Software Reusability*, volume 1, chapter 7, pages 159–225. Addison-Wesley, 1989.
- [30] J.A. Goguen. Parameterized programming and software architecture. In Murali Sitaraman, editor, *Fourth International Conference on Software Reuse*, pages 2–10. IEEE Computer Society Press, 1996.

- [31] J.V. Guttag, J.J. Horning, and J.M. Wing. Larch in 5 easy pieces. Technical Report 5, DEC SRC, 1985.
- [32] K. Harwood, P. Lindsay, and R. Matthews. An Approach to Constructing Verified Software. In *Seventeenth Annual Computer Science Conference*, pages 777–786, University of Canterbury, Christchurch, January 1994.
- [33] Keith Harwood. CARE2 and its type system. In J. Grundy, M. Schwenke, and T. Vickers, editors, *Work-in-progress papers of IRW/FMP'98*, number TR-CS-98-09, pages 13–21. The Australian National University, September 1998.
- [34] I. Hayes. Specification directed module testing. *IEEE Trans. Software Engineering*, 12(1):124–133, 1986.
- [35] I. J. Hayes and C. B. Jones. Specifications are not (necessarily) executable. *IEE/BCS Software Engineering Journal*, 4(6):330–338, November 1989.
- [36] D. Hemer. An algorithm for pattern-matching mathematical expressions. In L. Groves and S. Reeves, editors, *Proceedings of Formal Methods Pacific '97*, Discrete Mathematics and Theoretical Computer Science, pages 103–123. Springer Verlag, July 1997.
- [37] D. Hemer and P. Lindsay. Formal specification of an abstract syntax for the CARE language. Technical Report 95-44, Software Verification Research Centre, The University of Queensland, 1995.
- [38] D. Hemer and P. Lindsay. Specification-based strategies for module reuse. Technical Report 99-11, Software Verification Research Centre, The University of Queensland, 1999.
- [39] D. Hemer and P.A. Lindsay. Formal specification of proof obligation generation in CARE. Technical Report 95-13, Software Verification Research Centre, The University of Queensland, 1995.

- [40] D. Hemer and P.A. Lindsay. The CARE toolset for developing verified programs from formal specifications. In O. Frieder and J. Wigglesworth, editors, *Proceeding of the Fourth International Symposium on Assessment of Software Tools*, pages 24–35. IEEE Computer Society Press, May 1996.
- [41] D. Hemer and P.A. Lindsay. Reuse of verified design templates. In J. Fitzgerald, C. Jones, and P. Lucas, editors, *Formal Methods Europe '97*, number 1313 in Lecture Notes in Computer Science, pages 495–514. Springer, September 1997.
- [42] David Hemer. Concrete syntax for the CARE fragment language. Technical Report 95-14, The University of Queensland, February 1995.
- [43] C.A.R. Hoare. Proof of Correctness of Data Representations. *Acta Infomatica*, 1:271–281, 1972.
- [44] G.P. Huet. A unification algorithm for typed  $\lambda$ -calculus. *Theoretical Computer Science*, 1:27–57, 1975.
- [45] T. Isakowitz and R.J. Kauffman. Supporting search for reusable software objects. *IEEE Transactions on Software Engineering*, 22(6):407–423, June 1996.
- [46] J-J. Jeng and B.H.C Cheng. Specification matching for software reuse: A foundation. In *Proc. of ACM Symposium on Software Reuse*, pages 97–105, April 1995.
- [47] C. B. Jones, K. D. Jones, P. A. Lindsay, and R. Moore. *mural: A Formal Development Support System*. Springer-Verlag, 1991.
- [48] C.B. Jones. *Systematic Software Development Using VDM*. Prentice-Hall International, second edition, 1990.
- [49] R.K. Jullig. Applying formal software synthesis. *IEEE Software*, pages 11–22, May 1993.

- [50] E. Kazmierczak, P. Kearney, O. Traynor, and L. Wang. A modular extension to Z for specification, reasoning and refinement. Technical report 95-15, Software Verification Research Centre, The University of Queensland, February 1995.
- [51] A. Kelley and I. Pohl. *A Book on C*. Benjamin Cummins, third edition, 1995.
- [52] J. Kontio. A case study in applying a systematic method for COTS selection. In *Proceedings of the 18th International Conference on Software Engineering*, 1996.
- [53] C.W. Krueger. Software reuse. *ACM Computing Surveys*, 24(2):131–183, June 1992.
- [54] W. Lam. A case-study of requirements reuse through product families. *Annals of Software Engineering*, 5:253–278, 1998.
- [55] K. Lano. *The B Language and Method: A Guide to Practical Formal Development*. FACIT Series. Springer-Verlag, 1996.
- [56] Nam-Yong Lee and C.R. Litecky. An empirical study of software reuse with special attention to ada. *IEEE Transactions on Software Engineering*, 23(9):537–549, September 1997.
- [57] P. Lincoln and J. Christian. Adventures in associative-commutative unification. In Claude Kirchner, editor, *Unification*, pages 393–416. Academic Press, 1990.
- [58] P.A. Lindsay. A survey of mechanical support for formal reasoning. *Software Engineering Journal*, 3(1):3–27, January 1988.
- [59] P.A. Lindsay and D. Hemer. An industrial-strength method for the construction of formally verified software. In *Proceedings of the 1996 Australian Software Engineering Conference*, pages 27–36. IEEE Computer Society Press, July 1996.
- [60] P.A. Lindsay and D. Hemer. A template-based approach to construction of verified software. Technical Report 96-23, Software Verification Research Centre, 1996.

- [61] P.A. Lindsay and D. Hemer. Using CARE to construct verified software. In M.G. Hinchey and S. Liu, editors, *Proc. 1st Int Conf on Formal Eng Methods (ICFEM'97)*, pages 122–131. IEEE Computer Society Press, November 1997. SVRC TR 97-40.
- [62] M. Lowry, A. Philpot, T. Pressburger, and I. Underwood. Amphion: Automatic programming for scientific subroutine libraries. In *Proc. 8th International Symposium on Methodologies for Intelligent Systems*, volume 869 of *Lecture Notes in Computer Science*, pages 326–335. Springer Verlag, October 1994.
- [63] M. Lowry, A. Philpot, T. Pressburger, and I. Underwood. A formal approach to domain-oriented software design environments. In *Proc. 9th Knowledge Based Software Engineering Conference*, pages 48–57, 1994.
- [64] Rex Matthews and Trudy Weibel. Code synthesis in CARE. Technical Report 95-24, Software Verification Research Centre, The University of Queensland, October 1995.
- [65] M.D. McIlroy. Mass produced software components. *Software Engineering Concepts and Techniques*, pages 88–98, 1969.
- [66] R. Mili, A. Mili, and R.T. Mittermeir. Storing and retrieving software components: A refinement based system. *IEEE Transactions on Software Engineering*, 23(7):445–458, July 1997.
- [67] Dale Miller. Unification of simply typed lambda-terms as logic programming. In P.K. Furukawa, editor, *Proc. 1991 Joint Int. Conf. Logic Programming*, pages 253–281. MIT Press, 1991.
- [68] NATO Communications and Information Systems Agency. *NATO Standard for the Development of Reusable Software Components*, November 1995. draft.
- [69] J.E. Nicholls. Z Notation, Version 1.2. Technical report, Z Standards Panel, September 1995.

- [70] M. Nielson, K. Havelund, K. Ritter Wagner, and E. Saaman. The RAISE language, method and tools. *Formal Aspects of Computing*, 1:85–114, 1989.
- [71] Tobias Nipkow. Functional unification of higher-order patterns. In *Proceedings of the 8th IEEE Symposium Logic in Computer Science*, pages 64–74. IEEE Computer Society Press, June 1993.
- [72] L.C. Paulson. Isabelle: The next 700 theorem provers. In P. Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press, 1990.
- [73] L.C. Paulson. *Isabelle - A Generic Theorem Prover*. Number 828 in Lecture Notes in Computer Science. Springer-Verlag, 1994.
- [74] D.E. Perry and S.S. Popovich. Inquire: Predicate-based use and reuse. In *Proceedings of the 8th Knowledge-Based Software Engineering Conference*, pages 144–151, September 1993.
- [75] J.S. Poulin. Reuse: Been there, done that. *Communications of the ACM*, 42(5):98–100, May 1999.
- [76] W. Pree. Component-based software development - a new paradigm in software engineering. *Software - Concepts and Tools*, pages 169–174, 1997.
- [77] R. Prieto-Diaz and Freeman P. Classifying software for reusability. *IEEE Software*, 4:6–16, January 1987.
- [78] C. Rich and R.C. Waters. Formalizing reusable software components in the programmer's apprentice. In T.J. Biggerstaff and A.J. Perlis, editors, *Software Reusability*, volume 2, chapter 15, pages 313–343. ACM Press, 1989.
- [79] M. Rittri. Using types as search keys in function libraries. In *Proceedings of the Fourth International Conference on Functional Programming and Computer Architecture*, pages 174–183. ACM Press, 1989.

- [80] E.J. Rollins and J.M. Wing. Specifications as search keys for software libraries. In K. Furukawa, editor, *Eighth International Conference on Logic Programming*, pages 173–187. MIT Press, June 1991.
- [81] C. Runciman and I. Toyn. Retrieving re-usable software components by polymorphic type. In *Proceedings of the Fourth International Conference on Functional Programming and Computer Architecture*, pages 166–173. ACM Press, 1989.
- [82] H.A. Schmid. Creating applications from components: A manufacturing framework design. *IEEE Software*, pages 67–75, November 1996.
- [83] D.R. Smith. KIDS: A semiautomatic program development system. *IEEE Transactions on Software Engineering*, 16(9):1024–1043, September 1990.
- [84] S. Sokolowski. *Applicative High Order Programming*. Chapman and Hall, 1991.
- [85] I. Sommerville. *Software Engineering*. Addison Wesley, third edition, 1989.
- [86] J. M. Spivey. *The fUZZ Manual*. Computing Science Consultancy, 2 Willow Close, Garsington, Oxford OX9 9AN, UK, 2nd edition, 1992.
- [87] J.M. Spivey. *The Z Notation: a Reference Manual*. Prentice-Hall, New York, 1989.
- [88] V. Stavridou, A. BoothRoyd, P. Bradley, B. Dutertre, L. Shackleton, and R. Smith. Formal methods and safety critical systems in practice. *High Integrity Systems*, 1(5):423–445, 1996.
- [89] M. Stickel, R. Waldinger, M. Lowry, T. Pressburger, and I. Underwood. Deductive composition of astronomical software from subroutine libraries. In Alan Bundy, editor, *Twelfth International Conference on Automated Deduction*, number 814 in Lecture Notes in Artificial Intelligence, pages 341–355. Springer Verlag, June 1994.

- [90] Bjarne Stroustrup. *The C++ Programming Language*. Addison Wesley, second edition, 1991.
- [91] D.M. Volpano and R.B. Kieburtz. The templates approach to software reuse. In T.J. Biggerstaff and A.J.Perlis, editors, *Software Reusability*, volume 1, chapter 9, pages 247–255. Addison-Wesley, 1989.
- [92] M. Ward. Using formal transformations to construct a component repository. In L. Dusink and P. Hall, editors, *Software Re-Use, Utrecht 1989*. Springer Verlag, November 1989.
- [93] D. Wikstrom. *Functional Programming using Standard ML*. Prentice Hall, 1987.
- [94] D.A. Wolfram. The decidability of higher-order matching. In Jorg Siekmann Franz Baader and Wayne Snyder, editors, *Proceedings of the Sixth International Workshop on Unification, UNIF'92*, 1992.
- [95] A.M. Zaremski. *Signature and Specification Matching*. PhD thesis, Carnegie Mellon Computer Science Dept, January 1996. Technical Report CMU-CS-96-103.
- [96] A.M. Zaremski and J.M. Wing. Signature matching: a tool for using software libraries. *ACM Transactions on Software Engineering and Methodology*, 4(2):146–170, April 1995.
- [97] A.M. Zaremski and J.M. Wing. Specification matching of software components. In *Third ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 1996.

# Index

$=_{\alpha}$ , 73  
*AC\_equiv*, 142  
*Assertion*, 30  
*AssertionName*, 30  
*BFragmentSpec*, 35  
*BinConn*, 27  
*BranchingFragment*, 34  
*CombLogic*, 191  
*ExprEquiv*, 195  
*Fmla*, 27  
*FragBody*, 37  
*FragmentAdapt*, 88, 173  
*FragmentName*, 33  
*FragmentQuery*, 160  
*FunctionName*, 26  
*FunctionParam*, 26  
*FunctionParamDecl*, 44  
*FunctionSig*, 27  
*IOPerm*, 84  
*Identifier*, 81  
*Inst*, 73  
*InteractionLevel*, 194  
*Library*, 195  
*MathVarDecls*, 28  
*ModAdapt*, 179  
*ModQuery*, 179  
*Module*, 179  
*OpName*, 30  
*OperatorDecl*, 29  
*Permutation*, 84  
*Quant*, 27  
*RelationName*, 27  
*RelationParam*, 27  
*RelationParamDecl*, 44  
*RelationSig*, 28  
*Renaming*, 81  
*Report*, 37  
*Response*, 152  
*Set*, 25  
*SetConstructor*, 25  
*SetName*, 25  
*SetParam*, 25  
*SetParamDecl*, 44  
*SimpleFragment*, 34  
*SimpleFragmentSpec*, 34  
*SpecBranch*, 37  
*Strategy*, 195  
*Template*, 43  
*TemplateAdapt*, 94  
*TemplateName*, 43

*Term*, 26  
*Type*, 31  
*TypeAdapt*, 88  
*TypeBody*, 32  
*TypeCheck*, 195  
*TypeName*, 31  
*TypeQuery*, 160  
*Unit*, 28, 178  
*UnitAdapt*, 94, 178  
*UnitName*, 28  
*UnitQuery*, 178  
*Var*, 26  
*VarDeclar*, 32  
*VarRenaming*, 82  
*adapt\_fragment*, 88  
*adapt\_template*, 94  
*adapt\_type*, 88  
*adapt\_unit*, 94  
*areMergeableInsts*, 124  
*areMergeableRenamings*, 160  
*closure*, 90  
*create\_new\_vars*, 130  
*fnsts*, 73  
*flatten*, 141  
*free Vars*, 74, 75  
*generate\_next\_match*, 152  
*instantiate*, 76–78  
*instantiatie*, 88  
*match*, 121, 125, 126, 129, 131  
*match<sub>AC</sub>*, 143  
*match<sub>TC</sub>*, 149, 151  
*match<sub>all</sub>*, 189  
*match<sub>and</sub>*, 185  
*match<sub>bfrag</sub>*, 167  
*match<sub>funct</sub>*, 126  
*match<sub>hybrid</sub>*, 191  
*match<sub>interact</sub>*, 153  
*match<sub>mvs</sub>*, 130  
*match<sub>one</sub>*, 190  
*match<sub>or</sub>*, 187  
*match<sub>quant</sub>*, 130  
*match<sub>sfrag</sub>*, 164  
*match<sub>some</sub>*, 190  
*match<sub>type</sub>*, 168  
*match<sub>unit</sub>*, 178  
*match<sub>vs</sub>*, 162  
*match<sub>vs</sub>\**, 172  
*match<sub>xor</sub>*, 186  
*matches*, 161  
*matches<sub>AC</sub>*, 143  
*matches<sub>TC</sub>*, 147  
*matches<sub>all</sub>*, 180  
*matches<sub>and</sub>*, 184  
*matches<sub>frag</sub>*, 166  
*matches<sub>one</sub>*, 182  
*matches<sub>or</sub>*, 187  
*matches<sub>sb</sub>*, 166  
*matches<sub>sfrag</sub>*, 163  
*matches<sub>sfrag</sub>\**, 173  
*matches<sub>some</sub>*, 181

- matches<sub>type</sub>*, 168
- matches<sub>unit</sub>*, 178
- matches<sub>vs</sub>*, 161, 171
- matches<sub>xor</sub>*, 186
- mergeInstSet*, 125
- mergeUnitAdapt*, 178
- out\_signature*, 146
- permutationmap*, 84
- permute*, 84
- permuteBSpec*, 85
- permuteFragCalls*, 85
- permuteSSpec*, 85
- permuteVars*, 85
- rename*, 81, 82
- renameVars*, 75, 82
- replacement*, 127
- replacementList*, 128
- rinsts*, 73
- search*, 195
- sig*, 75
- signature*, 146
- sinsts*, 73
- substPlaceHolders*, 76
- supporters*, 90
- trivInst*, 125
- unifies*, 147, 153
- unify*, 147, 155
- unitsOf*, 179
- varSet*, 75
- CARE
  - development methodology, 47
  - library, 50
  - script, 49
  - tools, 49
    - code synthesiser, 52
    - pretty printers, 52
    - proof obligation generator, 51
    - script interpreter, 50
    - template instantiator, 51
    - theorem provers, 51
    - worksheet manager, 52
  - worksheet, 50
- AC equivalence, 142
- AC expressions, 141
- adaptation
  - expressions, 72
  - fragments, 88
  - modules, 89
  - target code, 91
  - templates, 94
  - units, 80
- alpha equivalence, 73
- applicability conditions, 44
- assertions, 29
- associative commutative matching, 140
- associative expressions, 141
- commutative expressions, 141
- expressions, 24
  - formulae, 27

- sets, 25
- terms, 26
- flattening AC expressions, 141
- formal parameter instantiation, 72
- fragments, 33
  - guarded specification part, 36
  - higher-level, 37
  - implementation, 37
  - primitive, 37
  - specification, 34
  - verification, 39
- function signature, 27
- instantiate
  - formulae, 78
  - fragments, 88
  - sets, 76
  - terms, 76
  - variable declarations, 78
- instantiation, 72
  - merging, 124
- match
  - branching fragments, 166
  - formulae, 129
  - operator declarations, 169
  - parametric functions, 126
  - quantified formulae, 130
  - sets, 131
  - simple fragments, 163
  - terms, 125
  - types, 168
  - variable declarations, 161
- matching
  - associative commutative, 140
  - interactive, 150
  - type-constrained, 145
- merging instantiations, 124
- modules, 40
- operator declarations, 29
- parameters, 44
- proof obligations
  - non-execution, 39
  - partial correctness, 39
  - termination, 39
  - well-formedness, 39
- relation signature, 28
- renaming identifiers, 81
- renaming variables, 82
- reordering variables, 84
- subsetting, 89
- template body, 45
- templates, 40
  - applicability condition, 44
  - body, 45
  - parameters, 44
  - verification, 45
- theory, 29
- trivial instantiation, 125

types

    implementations, 32

    specifications, 31

    verification, 33

unification, 153

units, 28

    assertions, 29

    definitions, 29

    fragments, 33

    higher-level, 29

    operator declarations, 29

    primitive, 28

    theories, 29

    types, 31

variable declarations, 32