References

[1] A. Armando, A. Smaill, and I.M. Green (1999). Automatic synthesis of recursive programs: The proof-planning paradigm. *J. Software Engineering* **6**(4), 329–356.

[2] D. Barker-Plummer (1992). Cliché programming in Prolog. In M. Bruynooghe, editor, *Proc. META'90*, pages 246–256.

[3] A. Bertoni, G. Mauri, and P. Miglioli (1983). On the power of model theory in specifying abstract data types and in capturing their recursiveness. *Fundamenta Informaticae* **VI**(2), 127–170.

[4] L. Blaine, L. Gilham, J. Liu, D.R. Smith, and S. Westfold (1998). PLANWARE: Domain-specific synthesis of high-performance schedulers. In D.F. Redmiles and B. Nuseibeh, editors, *Proc. ASE'98*, pages 270–279. IEEE Computer Society Press.

[5] H. Büyükyıldız and P. Flener (1998). Generalised logic program transformation schemas. In N.E. Fuchs, editor, *Proc. LOPSTR'97*, *LNCS* **1463**, 46–65. Springer-Verlag.

[6] E. Chasseur and Y. Deville (1998). Logic program schemas, semi-unification and constraints. In N.E. Fuchs, editor, *Proc. LOPSTR'97*, *LNCS* **1463**, 69–89. Springer-Verlag.

[7] N. Dershowitz (1983). *The Evolution of Programs*. Birkhäuser.

[8] Y. Deville (1990). *Logic Programming: Systematic Program Development*. Addison-Wesley.

[9] Y. Deville and J. Burnay (1989). Generalization and program schemata: A step towards computer-aided construction of logic programs. In E.L. Lusk and R.A. Overbeek, editors, *Proc. NACLP'89*, pages 409–425. MIT Press.

[10] P. Flener (1995). *Logic Program Synthesis from Incomplete Information*. Kluwer.

[11] P. Flener (1997). Inductive logic program synthesis with DIALOGS. In S. Muggleton, editor, *Proc. ILP'96*, *LNAI* **1314**, 175–198. Springer-Verlag.

[12] P. Flener and Y. Deville (1993). Logic program synthesis from incomplete specifications. *J. Symbolic Computation* **15**(5–6), 775–805.

[13] P. Flener and Y. Deville (1996). Logic program transformation through generalization schemata. In M. Proietti, editor, *Proc. LOPSTR'95*, *LNCS* **1048**, 171–173. Springer-Verlag.

[14] P. Flener, K.-K. Lau, and M. Ornaghi (1997). Correct-schema-guided synthesis of steadfast programs. In Y. Ledru and M. Lowry, editors, *Proc. ASE'97*, pages 153–160. IEEE Computer Society Press.

[15] P. Flener, K.-K. Lau, and M. Ornaghi (1998). On correct program schemas. In N.E. Fuchs, editor, *Proc. LOPSTR'97*, *LNCS* **1463**, 124–143. Springer-Verlag.

[16] P. Flener, H. Zidoum, and B. Hnich (1998). Schema-guided synthesis of constraint logic programs. In D.F. Redmiles and B. Nuseibeh, editors, *Proc. ASE'98*, pages 168–176. IEEE Computer Society Press.

[17] P. Flener and J.D.C. Richardson (1999). A unified view of programming schemas and proof methods. In: A. Bossi, editor, *Pre-Proc. LOPSTR'99*, pages 75–82. TR CS-99-16, University of Venice, Italy.

[18] P. Flener and S. Yılmaz (1999). Inductive synthesis of recursive logic programs: Achievements and prospects. *J. Logic Programming* **41**(2-3), 141–195.

[19] P. Flener, K.-K. Lau, M. Ornaghi, and J.D.C. Richardson (2000). An abstract formalisation of correct schemas. Accepted for publication in *J. Symbolic Computation*.

[20] N.E. Fuchs and M.P.J. Fromherz (1992). Schema-based transformation of logic programs. In T. Clement and K.-K. Lau, editors, *Proc. LOPSTR'91*, pages 111–125. Springer-Verlag.

[21] T.S. Gegg-Harrison (1991). Learning Prolog in a schema-based environment. *Instructional Science* **20**, 173–190.

[22] T.S. Gegg-Harrison (1994). Exploiting program schemata in an automated program debugger. *J. Artificial Intelligence in Education* **5**, 255–278.

[23] T.S. Gegg-Harrison (1995). Representing logic program schemata in λ-Prolog. In L. Sterling, editor, *Proc. ICLP'95*, pages 467–481. MIT Press.

[24] T.S. Gegg-Harrison (1997). Extensible logic program schemata. In J. Gallagher, editor, *Proc. LOPSTR'96*, *LNCS* **1207**, 256–274. Springer-Verlag.

[25] J.A. Goguen and R.M. Burstall (1992). Institutions: Abstract model theory for specification and programming. *J. ACM* **39**(1), 95–146.

[26] J.A. Goguen and J. Meseguer (1987). Unifying functional, object-oriented and relational programming with logical semantics. In B. Shriver and P. Wegner, editors, *Research Directions in Object-Oriented Programming*, pages 417–477. MIT Press.

[27] J.A. Goguen, J.W. Thatcher, and E. Wagner (1978). An initial algebra approach to specification, correctness and implementation. In R. Yeh, editor, *Current Trends in Programming Methodology, IV*, pages 80–149. Prentice-Hall.

[28] A. Hamfelt and J. Fischer Nilsson (1997). Towards a logic programming methodology based on higher-order predicates. *New Generation Computing* **15**(4), 421–448.

[29] G. Huet and B. Lang (1978). Proving and applying program transformations expressed with second-order patterns. *Acta Informatica* **11**, 31–55.

[30] A.-L. Johansson (1994). Interactive program derivation using program schemata and incrementally generated strategies. In Y. Deville, editor, *Proc. LOPSTR'93*, pages 100–112. Springer-Verlag.

[31] Y. Kodratoff and J.-P. Jouannaud (1984). Synthesizing LISP programs working on the list level of embedding. In A.W. Biermann, G. Guiho, and Y. Kodratoff, editors, *Automatic Program Construction Techniques*, pages 325–374. Macmillan.

[32] I. Kraan, D. Basin, and A. Bundy (1996). Middle-out reasoning for synthesis and induction. *J. Automated Reasoning* **16**(1–2), 113–145.

[33] D. Lacey (1999). *Logic program synthesis via proof planning using λclam*. M.Sc. Thesis, School of Artificial Intelligence, Division of Informatics, University of Edinburgh.

[34] K.-K. Lau and S. D. Prestwich (1991). Synthesis of a family of recursive sorting procedures. In V. Saraswat and K. Ueda, editors, *Proc. ILPS'91*, pages 641–658. MIT Press.

[35] K.-K. Lau and M. Ornaghi (1994). On specification frameworks and deductive synthesis of logic programs. In L. Fribourg and F. Turini, editors, *Proc. LOPSTR/META'94*, *LNCS* **883**, 104–121. Springer-Verlag.

[36] K.-K. Lau and M. Ornaghi (1997a). Forms of logic specifications: A preliminary study. In J. Gallagher, editor, *Proc. LOPSTR'96*, *LNCS* **1207**, 295–312. Springer-Verlag.

[37] K.-K. Lau and M. Ornaghi (1997b). The relationship between logic programs and specifications: The subset example revisited. *J. Logic Programming* **30**(3), 239–257.

[38] K.-K. Lau, M. Ornaghi, and S.-Å. Tärnlund (1994). The halting problem for deductive synthesis of logic programs. In P. van Hentenryck, editor, *Proc. ICLP'94*, pages 665–683. MIT Press.

[39] K.-K. Lau, M. Ornaghi, and S.-Å. Tärnlund (1999). Steadfast logic programs. *J. Logic Programming* **38**(3), 259–294, 1999.

[40] J.W. Lloyd (1987). *Foundations of Logic Programming*. Springer-Verlag, 2nd edition.

[41] Z. Manna (1974). *Mathematical Theory of Computation*. McGraw-Hill.

[42] E. Marakakis and J.P. Gallagher (1994). Schema-based top-down design of logic programs using abstract data types. In L. Fribourg and F. Turini, editors, *Proc. LOPSTR/META'94*, *LNCS* **883**, 138–153. Springer-Verlag.

[43] N. Marti-Oliet and J. Meseguer (1996). Inclusions and subtypes II: Higher-order case. *J. Logic Computation* **6**(4), 541–572.

[44] J.D.C. Richardson and N.E. Fuchs (1998). Development of correct transformational schemata for Prolog programs. In N.E. Fuchs, editor, *Proc. LOPSTR'97*, *LNCS* **1463**, 263–281. Springer-Verlag.

[45] J.D.C. Richardson, A. Smaill, and I.M. Green (1998). System description: Proof planning in higher-order logic with λclam. In C. Kirchner and H. Kirchner, editors, *Proc. CADE'98*, *LNCS* **1421**, 129–133. Springer-Verlag.

[46] D. Sannella and A. Tarlecki (1997). Essential concepts of algebraic specification and program development. *Formal Aspects of Computing* **9**, 229–269.

[47] D.R. Smith (1982). Derived preconditions and their use in program synthesis. In D.W. Loveland, editor, *Proc. CADE'82*, pages 172–193.

[48] D.R. Smith (1984). The synthesis of LISP programs from examples: A survey. In A.W. Biermann, G. Guiho, and Y. Kodratoff, editors, *Automatic Program Construction Techniques*, pages 307–324. Macmillan.

[49] D.R. Smith (1985). Top-down synthesis of divide-and-conquer algorithms. *Artificial Intelligence* **27**(1), 43–96.

[50] D.R. Smith (1990). KIDS: A semiautomatic program development system. *IEEE Trans. Software Engineering* **16**(9), 1024–1043.

[51] D.R. Smith (1993). Constructing specification morphisms. *J. Symbolic Computation* **15**(5–6), 571–606.

[52] D.R. Smith (1994). Towards the synthesis of constraint propagation algorithms. In Y. Deville, editor, *Proc. LOPSTR'93*, pages 1–9. Springer-Verlag.

[53] D.R. Smith (1996). Toward a classification approach to design. *Proc. AMAST'96*, *LNCS* **1101**, 62–84. Springer-Verlag.

[54] Y.V. Srinivas and R. Jüllig (1995). SPECWARE: Formal support for composing software. In B. Möller, editor, *Proc. MPC'95*, *LNCS* **947**, 399–422. Springer-Verlag.

[55] L.S. Sterling and M. Kirschenbaum (1993). Applying techniques to skeletons. In J.-M. Jacquet, editor, *Constructing Logic Programs*, pages 127–140. John Wiley.

[56] P.D. Summers (1977). A methodology for LISP program construction from examples. *J. ACM* **24**(1), 161–175.

[57] W.W. Vasconcelos and N.E. Fuchs (1996). An opportunistic approach for logic program analysis and optimisation using enhanced schema-based transformations. In M. Proietti, editor, *Proc. LOPSTR'95*, *LNCS* **1048**, 174–188. Springer-Verlag.

[58] M. Wirsing (1990). Algebraic specification. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 675–788. Elsevier.