Higher Order Unification 30 years later (Extended Abstract)

Gérard Huet

INRIA-Rocquencourt
BP 105, F-78153 Le Chesnay Cedex
Gerard.Huet@inria.fr

Abstract. The talk will present a survey of higher order unification, covering an outline of its historical development, a summary of its applications to three fields: automated theorem proving, and more generally engineering of proof assistants, programming environments and software engineering, and finally computational linguistics. It concludes by a presentation of open problems, and a few prospective remarks on promising future directions. This presentation assumes as background the survey by Gilles Dowek in the Handbook of automated theorem proving [28].

1 Theory

1.1 Early History

The problem of automating higher order logic was first posed by J. A. Robinson in [101, 102]. Peter Andrews formulated a version of the resolution principle for accommodating Church's Simple Theory of Types [16, 4], and as early as 1964 Jim Guard's team at Applied Logic Corporation had been independently investigating higher order logic versions of unification [42, 41], but the complexity of the problem appeared when higher order unification was shown to be undecidable in 1972, independently by G. Huet [46] and C. Lucchesi [61], a result to be sharpened by W. Goldfarb [39]. Jensen and Pietrzykowky [90, 50], at University of Waterloo, gave a complete recursive enumeration of unifiers, but the process was over-generative in an untractable fashion, since computational quagmires happened where at every step of the process all previous solutions were subsumed by the next one. Huet proposed a more tractable enumeration of pre-unifiers for better conditioned problems, where rigidity of one of the problem terms led to sequential behavior. By delaying ill-conditioned problems in the form of constraints, it became possible to implement a complete version of the resolution principle for Church's Type Theory [49]. An extensive theory of unification in various algebraic settings, building on early work of G. Plotkin [91] was developed by G. Huet in his Thèse d'Etat [47], while P. Andrews and his collaborators at CMU went on implementing constrained resolution in the proof assistant TPS [5].

Thus the original motivation for higher-order unification was its use as a fundamental algorithm in automated theorem-provers. We shall see below that the design and implementation of proof assistants is still its main application area, not just in the setting of classical logic with versions of the resolution principle, but in more general meta-theoretic frameworks, where it is the work horse of pattern-matching and more generally proof search. For instance, extensions of the term-rewriting paradigm of equational logic, building on higher-order unification, were proposed by Nipkow, Jouannaud and Okada, Breazu-Tannen, Blanqui, and many others [76, 10], while C. Benzmüller investigated the addition of equality paramodulation to higher-order resolution [8, 9].

1.2 λ Prolog

It was soon realised that resolution was better behaved in the restricted context of Horn sentences, and Colmerauer and Kowalski took advantage of this quality in the framework of first-order logic with the design of PROLOG, which spurred the whole new field of Logic Programming. The generalization of this promising technology to higher-order logic was successfully accomplished by D. Miller with his λ-PROLOG system [65, 73]. G. Nadathur further investigated the λ-PROLOG methodology and its efficient implementation [71, 72, 70, 73, 74]. In the Mali team in Rennes, Yves Bekkers and colleagues implemented a λ-PROLOG machine with specific dynamic memory management [7, 14]. The most impressive application of the λ-PROLOG formalism was accomplished by F. Pfenning in his ELF system, a logic programming environment for Edinburgh's Logical Framework [86, 89].

The theory of higher-order unification in more powerful higher order frameworks, encompassing dependent types, polymorphism, or inductive types, was investigated by C. Elliott[31, 32], G. Pym [97], G. Dowek [20, 26], F. Pfenning [87, 88] etc in a series of publications in the 80's and 90's. We refer the interested reader to the extensive presentation in G. Dowek's survey [28].

1.3 Matching

An important special case of unification (i.e. solving equations in term structures) is when one of the terms is closed. Unification in this special case is pattern matching, a specially important algorithm in symbolic computation. Huet's semi-decision algorithm turned into an algorithm generating a finite set of solutions in the case of second-order terms [47, 48, 22]. Furthermore, it was possible to combine pattern-matching for linear terms with first-order unification, as Miller demonstrated with his higher-order patterns [66, 78]. However, the general problem in the full higher-order hierarchy remained open for a long time. G. Dowek showed that the third-order problem was decidable [27], while extensions in various other directions led to undecidability [21, 24]. Padovani showed that the fourth-order case was decidable [81–83]. Wolfram gave an algorithm which always terminate [116], but whose completeness is dubious. More recently, Loader announced the undecidability of higher-order matching in [59, 60], but the argument does not carry over if the η rule of extensionality is added to β computation rule of λ calculus. This vexing but important problem is thus still open after 30 years of intense investigation.

2 Applications

The talk will survey applications in three domains.

2.1 Proof technology

Early experiments with resolution systems as proof assistants were extremely naive, since full automation was aimed at, for mathematical theories which are undecidable or at least badly untractable. It is interesting that to this day research teams keep the pretense of solving deep mathematical problems in a fully automated fashion, despite all evidence to the contrary.

Hopefully better methodologies exist than attempting to solve mathematics with combinatorial blackboxes. Proof Assistants, following the early models of Automath, LCF and NuPRL, have been designed as interactive systems, where the mathematically sophisticated user drives a proof search process with the help of specialised, programmable tactics implementing decision or semi-decision procedures, which operate on formulas kept in user-understandable terms. Two paradigms are of paramount importance in this newer vision of proof search: the use of general meta-level formalisms (the so-called logical frameworks, where generic pattern-matching and unification algorithms are put to use), and the recognition of proof structures as higher-order objects themselves (through suitable instances of the Curry-Howard correspondance). This gave a second life to higher-order unification, at the level of proof search this time (and not just at the level of terms search). This methodology was clearly articulated by G. Dowek in his thesis [20, 23] and by by D. Miller and A. Felty [2, 35]. Typical of this approach are the proof assistants Isabelle [84], the Edinburgh Logical Framework and its ELF Logic Programming engine [86, 89], and the various versions of the Calculus of Constructions or Intuitionistic Type Theory implementations (Coq, Lego, Alf).

2.2 Program Schemas Manipulation and Software Engineering

The second-order matching algorithm presented in the 70's by Huet and Lang [48] was motivated by an application to program schemas recognition, in a software engineering paradigm of program transformations originally proposed by Burstall and Darlington. Indeed, all the Burstall and Darlington rewrite rules were easily encompasssed by this technology, but it seemed difficult to extrapolate this simple methodology to software engineering in the large. Still, grandiose projects of Inferential Programming based on Semantically Based Programming Tools were lavishly financed in the 80's on such principles [55]. What finally emerged when the dust settled was the notion of Higher-order Abstract Syntax as a fundamental structure for logic-based programming environments [67, 45, 85, 3, 51, 64].

2.3 Anaphora Resolution and Other Computational Linguistics Applications

Meanwhile, researchers in computational linguistics working at the frontier between syntax and semantics, and notably within Montague Semantics, had been investigating the resolution of anaphora constructions (pronouns, focus, ellipsis, etc.) through the higher order unification algorithm. Such experiments were facilitated by the widespread adoption of PROLOG as a computation paradigm in this community. Typical representatives of this approach were Dale Miller and Gopalan Nadathur at CMU [68], S. Shieber and colleagues in Harvard [62, 115], S. Pulman in SRI/Cambridge [96], and C. Gardent and colleagues in Saarbrücken [15, 37, 38, 36].

3 Prospective

The talk will end with some prospective remarks.

References

1. G. Amiot. The undecidability of the second order predicate unification problem. *Archive for mathematical logic*, 30:193–199, 1990.
2. Dale Miller Amy Felty. Specifying theorem provers in a higher-order logic programming language. In *Proceedings of the 9th Int. Conference on Automated Deduction (CADE-9)*, volume 310 of *Lecture Notes in Computer Science*, pages 61–80. Springer-Verlag, Argonne, Illinois, 1988.
3. Penny Anderson. Represnting proof transformations for program optimization. In *Proceedings of the 12th Int. Conference on Automated Deduction (CADE-12)*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 575–589, Nancy, France, 1994. Springer-Verlag.
4. P. B. Andrews. Resolution in type theory. *The Journal of Symbolic Logic*, 36(3):414–432, 1971.
5. P. B. Andrews, M. Bishop, S. Issar, D. Nesmith, F. Pfenning, and H. Xi. Tps: a theorem proving system for classical type theory. *J. of Automated Reasoning*, 16:321–353, 1996.
6. J. Avenhaus and C. A. Loría-Sáenz. Higher-order conditional rewriting and narrowing. In J.-P. Jouannaud, editor, *International Conference on Constaints in Computational Logic*, volume 845 of *Lecture Notes in Computer Science*, pages 269–284. Springer-Verlag, 1994.
7. Yves Bekkers, Olivier Ridoux, and Lucien Ungaro. Dynamic memory management for sequential logic programming languages. In *IWMM*, pages 82–102, 1992.
8. C. Benzmüller. Extensional higher-order resolution. In *Proceedings of the 15th Int. Conference on Automated Deduction (CADE-15)*, volume 1421 of *Lecture Notes in Artificial Intelligence*, pages 56–71. Springer-Verlag, Lindau, Germany, 1998.
9. C. Benzmüller. Equality and extensionality in automated higher-order theorem proving. Doctoral Dissertation, Universität des Saarlandes, Saarbrücken, 1999.
10. Frédéric Blanqui. Théorie des types et réécriture. Thèse de Doctorat, Université Paris-Sud, 2001.

11. P. Borovanský. Implementation of higher-order unification based on calculus of explicit substitution. In M. Bartošek, J. Staudek, and J. Wiedermann, editors, *SOFSEM : Theory and Practice of Informatics*, number 1012 in Lecture Notes in Computer Science, pages 363–368. Springer-Verlag, 1995.
12. A. Boudet and E. Contejean. AC-unification of higher-order patterns. In G. Smolka, editor, *Principles and Practice of Constraint Programming*, volume 1330 of *Lecture Notes in Computer Science*, pages 267–281. Springer-Verlag, 1997.
13. D. Briaud. Higher order unification as typed narrowing. Technical Report 96-R-112, Centre de recherche en informatique de Nancy, 1996.
14. Pascal Brisset and Olivier Ridoux. The architecture of an implementation of lambdaprolog: Prolog/mali. Technical Report RR-2392, INRIA, Oct. 1994.
15. M. Kohlhase C. Gardent and K. Konrad. Higher-order colored unification: A linguistic application. *J. of Logic, Language and Information*, 9:313–338, 2001.
16. A. Church. A formulation of the simple theory of types. *The Journal of Symbolic Logic*, 5(1):56–68, 1940.
17. H. Comon and Y. Jurski. Higher-order matching and tree automata. In M. Nielsen and W. Thomas, editors, *Conference on Computer Science Logic*, volume 1414 of *Lecture Notes in Computer Science*, pages 157–176. Springer-Verlag, 1997.
18. D. J. Dougherty. Higher-order unification via combinators. *Theoretical Computer Science*, 114:273–298, 1993.
19. D. J. Dougherty and P. Johann. A combinatory logic approach to higher-order E-unification. In D. Kapur, editor, *Conference on Automated Deduction*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 79–93. Springer-Verlag, 1992.
20. G. Dowek. Démonstration automatique dans le calcul des constructions. Thèse de Doctorat, Université de Paris VII, 1991.
21. G. Dowek. L'indécidabilité du filtrage du troisième ordre dans les calculs avec types dépendants ou constructeurs de types (the undecidability of third order pattern matching in calculi with dependent types or type constructors). *Comptes Rendus à l'Académie des Sciences*, I, 312(12):951–956, 1991. Erratum, ibid. I, 318, 1994, p. 873.
22. G. Dowek. A second-order pattern matching algorithm in the cube of typed λ-calculi. In A. Tarlecki, editor, *Mathematical Foundation of Computer Science*, volume 520 of *Lecture notes in computer science*, pages 151–160. Springer-Verlag, 1991.
23. G. Dowek. A complete proof synthesis method for the cube of type systems. *Journal of Logic and Computation*, 3(3):287–315, 1993.
24. G. Dowek. The undecidability of pattern matching in calculi where primitive recursive functions are representable. *Theoretical Computer Science*, 107:349–356, 1993.
25. G. Dowek. The undecidability of typability in the lambda-pi-calculus. In M. Bezem and J. F. Groote, editors, *Typed Lambda Calculi and Applications*, number 664 in Lecture Notes in Computer Science, pages 139–145. Springer-Verlag, 1993.
26. G. Dowek. A unification algorithm for second-order linear terms. Manuscript, 1993.
27. G. Dowek. Third order matching is decidable. *Annals of Pure and Applied Logic*, 69:135–155, 1994.
28. G. Dowek. *Higher-Order Unification and Matching*, chapter 16, pages 1009–1062. Elsevier, 2001.
29. G. Dowek, T. Hardin, and C. Kirchner. Higher-order unification via explicit substitutions. *Information and Computation*, 157:183–235, 2000.

30. G. Dowek, T. Hardin, C. Kirchner, and F. Pfenning. Unification via explicit substitutions: the case of higher-order patterns. In M. Maher, editor, *Joint International Conference and Symposium on Logic Programming*, pages 259–273. The MIT Press, 1996.
31. C. M. Elliott. Higher-order unification with dependent function types. In N. Dershowitz, editor, *Internatinal Conference on Rewriting Techniques and Applications*, volume 355 of *Lecture Notes in Computer Science*, pages 121–136. Springer-Verlag, 1989.
32. C. M. Elliott. *Extensions and applications of higher-order unification*. PhD thesis, Carnegie Mellon University, 1990.
33. W. M. Farmer. A unification algorithm for second-order monadic terms. *Annals of Pure and applied Logic*, 39:131–174, 1988.
34. W. M. Farmer. Simple second order languages for which unification is undecidable. *Theoretical Computer Science*, 87:25–41, 1991.
35. Amy Felty. Implementing tactics and tacticals in a higher-order logic programming language. *Journal of Automated Reasoning*, 11:43–81, 1993.
36. C. Gardent. Deaccenting and higher-order unification. *J. of Logic, Language and Information*, 9:313–338, 2000.
37. C. Gardent and M. Kohlhase. Focus and higher-order unification. In *16th International Conference on Computational Linguistics, Copenhagen*, 1996.
38. C. Gardent and M. Kohlhase. Higher-order coloured unification and natural language semantics. In ACL, editor, *34th Annual Meeting of the Association for Computational Linguistics, Santa Cruz*, 1996.
39. W. D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.
40. J. Goubault. Higher-order rigid E-unification. In F. Pfenning, editor, *5th International Conference on Logic Programming and Automated Reasoning*, number 822 in Lecture Notes in Artificial Intelligence, pages 129–143. Springer-Verlag, 1994.
41. W. E. Gould. A matching procedure for ω-order logic. Scientific report 4, AFCRL 66-781 (contract AF 19 (628)-3250) AD 646 560.
42. J. R. Guard. Automated logic for semi-automated mathematics. Scientific report 1, AFCRL 64-411 (contract AF 19 (628)-3250) AS 602 710, 1964.
43. M. Hagiya. Programming by example and proving by example using higher-order unification. In M.E. Stickel, editor, *Conference on Automated Deduction*, number 449 in Lecture Notes in Computer Science, pages 588–602. Springer-Verlag, 1990.
44. M. Hagiya. Higher-order unification as a theorem proving procedure. In K. Furukawa, editor, *International Conference on Logic Programming*, pages 270–284. MIT Press, 1991.
45. John Hannan and Dale Miller. Uses of higher-order unification for implementing programs transformers. In R.A. Kowalski an K.A. Bowen, editor, *International Conference and Symposium on Logic Programming*, pages 942–959, 1988.
46. G. Huet. The undecidability of unification in third order logic. *Information and Control*, 22:257–267, 1973.
47. G. Huet. Résolution d'équations dans les langages d'ordre 1,2, ..., ω. Thèse d'État, Université de Paris VII, 1976.
48. G. Huet and B. Lang. Proving and applying program transformations expressed with second order patterns. *Acta Informatica*, 11:31–55, 1978.
49. Gérard Huet. *Constrained resolution: a complete method for higher order logic*. PhD thesis, Case Western University, Aug. 1972.
50. D. C. Jensen and T. Pietrzykowski. Mechanizing ω-order type theory through unification. *Theoretical Computer Science*, 3:123–171, 1976.

51. Frank Pfenning Joëlle Despeyroux and Carsten Schürmann. Primitive recursion for higher-order syntax. In R. Hindley, editor, *Proceedings of the 3rd Int. Conference on Typed Lambda Calculus and Applications (TLCA'97)*, volume 1210 of *Lecture Notes in Computer Science*, pages 147–163, Nancy, France, 1997. Springer-Verlag.
52. Patricia Johann and Michael Kohlhase. Unification in an extensional lambda calculus with ordered function sorts and constant overloading. In Alan Bundy, editor, *Conference on Automated Deduction*, number 814 in Lecture Notes in Artificial Intelligence, pages 620–634. Springer-Verlag, 1994.
53. C. Kirchner and Ch. Ringeissen. Higher-order equational unification via explicit substitutions. In M. Hanus, J. Heering, and K. Meinke, editors, *Algebraic and Logic Programming, International Joint Conference ALP'97-HOA'97*, volume 1298 of *Lecture Notes in Computer Science*, pages 61–75. Springer-Verlag, 1997.
54. K. Kwon and G. Nadathur. An instruction set for higher-order hereditary harrop formulas (extended abstract). In *Proceedings of the Workshop on the lambda Prolog Programming Language, UPenn Technical Report MS-CIS-92-86*, 1992.
55. Peter Lee, Frank Pfenning, John Reynolds, Gene Rollins, and Dana Scott. Research on semantically based program-design environments: The ergo project in 1988. Technical Report CMU-CS-88-118, Computer Science Department, Carnegie Mellon University, 1988.
56. J. Levy. Linear second order unification. In H. Ganzinger, editor, *Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 332–346. Springer-Verlag, 1996.
57. Jordi Levy and Margus Veanes. On unification problems in restricted second-order languages. In *Annual Conf. of the European Ass. of Computer Science Logic (CSL98)*, Brno, Czech Republic, 1998.
58. Jordi Levy and Mateu Villaret. Linear second-order unification and context unification with tree-regular constraints. In *Proceedings of the 11th Int. Conf. on Rewriting Techniques and Applications (RTA'00)*, volume 1833 of *Lecture Notes in Computer Science*, pages 156–171, Norwich, UK, 2000. Springer-Verlag.
59. R. Loader. The undecidability of λ-definability. To appear in Church memorial volume, 1994.
60. R. Loader. Higher order β matching is undecidable. Private communication, 2001.
61. C. L. Lucchesi. The undecidability of the unification problem for third order languages. Technical Report CSRR 2060, Department of applied analysis and computer science, University of Waterloo, 1972.
62. Stuart M. Shieber Mary Dalrymple and Fernando C.N. Pereira. Ellipsis and higher-order unification. *Linguistic and Philosophy*, 14:399–452, 1991.
63. R. Mayr and T. Nipkow. Higher-order rewrite systems and their confluence. *Theoretical Computer Science*, 192:3–29, 1998.
64. Raymond McDowell and Dale Miller. Reasoning with higher-order abstract syntax in a logical framework. *ACM Transactions on Computational Logic (TOCL)*, 3:80–136, 2002.
65. Dale Miller. Abstractions in logic programming. In P. Odifreddi, editor, *Logic and computer science*, pages 329–359. Academic Press, 1991.
66. Dale Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*, 14:321–358, 1992.

67. Dale Miller and Gopalan Nadathur. A logic programming approach to manipulating formulas and programs. In *IEEE Symposium on Logic Programming*, pages 247–255, San Francisco, California, 1986.
68. Dale Miller and Gopalan Nadathur. Some uses of higher-order logic in computational linguistics. In *24th Annual Meeting of the Association for Computational Linguistics*, pages 247–255, 1986.
69. O. Müller and F. Weber. Theory and practice of minimal modular higher-order E-unification. In A. Bundy, editor, *Conference on Automated Deduction*, number 814 in Lecture Notes in Artificial Intelligence, pages 650–664. Springer-Verlag, 1994.
70. Gopalan Nadathur. An explicit substitution notation in a λprolog implementation. In *First International Workshop on Explicit Substitutions*, 1998.
71. Gopalan Nadathur. *A Higher-Order Logic as the Basis for Logic Programming*. PhD thesis, University of Pennsylvania, Dec. 1986.
72. Gopalan Nadathur and Dale Miller. Higher-order horn clauses. *J. of the Association for Computing Machinery*, 37:777–814, 1990.
73. Gopalan Nadathur and Dale Miller. Higher-order logic programming. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of logic in artificial intelligence and logic programming*, volume 5, pages 499–590. Clarendon Press, 1998.
74. Gopalan Nadathur and D.J. Mitchell. System description : A compiler and abstract machine based implementation of λprolog. In *Conference on Automated Deduction*, 1999.
75. T. Nipkow. Higher-order critical pairs. In *Logic in Computer Science*, pages 342–349, 1991.
76. T. Nipkow. Higher-order rewrite systems. In *Proceedings of the 6th Int. Conf. on Rewriting Techniques and Applications (RTA-95)*, volume 914 of *Lecture Notes in Computer Science*, pages 256–256, Keiserlautern, Germany, 1995. Springer-Verlag.
77. T. Nipkow and Ch. Prehofer. Higher-order rewriting and equational reasoning. In W. Bibel and P.H. Schmitt, editors, *Automated Deduction - A Basis for Applications*, volume 1, pages 399–430. Kluwer, 1998.
78. Tobias Nipkow. Functional unification of higher-order patterns. In *Proc. 8th IEEE Symp. Logic in Computer Science*, pages 64–74, 1993.
79. Tobias Nipkow and Zhenyu Qian. Modular higher-order e-unification. In *Proceedings of the 4th Int. Conf. on Rewriting Techniques and Applications (RTA)*, volume 488 of *Lecture Notes in Computer Science*, pages 200–214. Springer-Verlag, 1991.
80. Tobias Nipkow and Zhenyu Qian. Reduction and unification in lambda calculi with a general notion of subtype. *Journal of Automated Reasoning*, 12:389–406, 1994.
81. V. Padovani. Fourth-order matching is decidable. Manuscript, 1994.
82. V. Padovani. Decidability of all minimal models. In S. Berardi and M. Coppo, editors, *Types for Proof and Programs 1995*, number 1158 in Lecture Notes in Computer Science, pages 201–215. Springer-Verlag, 1996.
83. V. Padovani. Filtrage d'ordre supérieur. Thèse de Doctorat, Université de Paris VII, 1996.
84. Larry C. Paulson. Isabelle: The next 700 theorem provers. In P. Odifreddi, editor, *Logic and computer science*, pages 361–385. Academic Press, 1991.
85. F. Pfenning. Partial polymorphic type inference and higher-order unification. In *Conference on Lisp and Functional Programming*, pages 153–163, 1988.

86. F. Pfenning. Logic programming in the LF logical framework. In G. Huet and G. Plotkin, editors, *Logical frameworks*, pages 149–181. Cambridge University Press, 1991.
87. F. Pfenning. Unification and anti-unification in the calculus of constructions. In *Logic in Computer Science*, pages 74–85, 1991.
88. F. Pfenning and I. Cervesato. Linear higher-order pre-unification. In *Logic in Computer Science*, 1997.
89. Frank Pfenning and Carsten Schürmann. System description: Twelf — a meta-logical framework for deductive systems. In *Proceedings of the 16th Int. Conference on Automated Deduction (CADE-16)*, volume 1632 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1999.
90. T. Pietrzykowski. A complete mecanization of second-order type theory. *J. of the Association for Computing Machinery*, 20:333–364, 1973.
91. G. Plotkin. Building-in equational theories. *Machine Intelligence*, 7:73–90, 1972.
92. C. Prehofer. Decidable higher-order unification problems. In A. Bundy, editor, *Conference on Automated Deduction*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 635–649. Springer-Verlag, 1994.
93. C. Prehofer. Higher-order narrowing. In *Logic in Computer Science (LICS'94*, pages 507–516, 1994.
94. C. Prehofer. Solving higher-order equations: from logic to programming. Doctoral thesis, Technische Universität München. Technical Report TUM-19508, Institut für Informatik, TUM, München, 1995.
95. C. Prehofer. *Solving Higher-Order Equations: From Logic to Programming*. Progress in theoretical coputer science. Birkhäuser, 1998.
96. S. Pulman. Higher-order unification and the interpretation of focus. *Linguistics and Philosophy*, 20:73–115, 1997.
97. D. Pym. Proof, search and computation in general logic. Doctoral thesis, University of Edinburgh, 1990.
98. Z. Qian. Higher-order equational logic programming. In *Principle of Programming Languages*, pages 254–267, 1994.
99. Z. Qian and K. Wang. Higher-order equational E-unification for arbitrary theories. In K. Apt, editor, *Joint International Conference and Symposium on Logic Programming*, 1992.
100. Z. Qian and K. Wang. Modular AC unification of higher-order patterns. In J.-P. Jouannaud, editor, *International Conference on Constaints in Computational Logic*, volume 845 of *Lecture Notes in Computer Science*, pages 105–120. Springer-Verlag, 1994.
101. J. A. Robinson. New directions in mechanical theorem proving. In A. J. H. Morrell, editor, *International Federation for Information Processing Congress, 1968*, pages 63–67. North Holland, 1969.
102. J. A. Robinson. A note on mechanizing higher order logic. *Machine Intelligence*, 5:123–133, 1970.
103. H. Saïdi. Résolution d'équations dans le système T de Gödel. Mémoire de DEA, Université de Paris VII, 1994.
104. M. Schmidt-Schauß. Unification of stratified second-order terms. Technical Report 12, J.W.Goethe-Universität, Frankfurt, 1994.
105. M. Schmidt-Schauß. Decidability of bounded second order unification. Technical Report 11, J.W.Goethe-Universität, Frankfurt, 1999.
106. M. Schmidt-Schauß and K.U. Schulz. Solvability of context equations with two context variables is decidable. In H. Ganzinger, editor, *Conference on Automated*

Deduction, number 1632 in Lecture Notes in Artificial Intelligence, pages 67–81, 1999.
107. A. Schubert. Linear interpolation for the higher order matching problem. In M. Bidoit and M. Dauchet, editors, *Theory and Practice of Software Development*, volume 1214 of *Lecture Notes in Computer science*, pages 441–452. Springer-Verlag, 1997.
108. A. Schubert. Second-order unification and type inference for Church-style polymorphism. In *Principle of Programming Languages*, pages 279–288, 1998.
109. W. Snyder. Higher-order E-unification. In M. E. Stickel, editor, *Conference on Automated Deduction*, volume 449 of *Lecture Notes in Artificial Intelligence*, pages 573–587. Springer-Verlag, 1990.
110. W. Snyder and J. Gallier. Higher order unification revisited: Complete sets of tranformations. *Journal of Symbolic Computation*, 8(1 & 2):101–140, 1989. Special issue on unification. Part two.
111. W. Snyder and J. Gallier. Higher-order unification revisited: complete sets of transformations. *Journal of Symbolic Computation*, 8:101–140, 1989.
112. J. Springintveld. Algorithms for type theory. Doctoral thesis, Utrecht University, 1995.
113. J. Springintveld. Third-order matching in presence of type constructors. In M. Dezani-Ciancagliani and G. Plotkin, editors, *Typed Lambda Calculi and Applications*, volume 902 of *Lecture Notes in Computer Science*, pages 428–442. Springer-Verlag, 1995.
114. J. Springintveld. Third-order matching in the polymorphic lambda calculus. In G. Dowek, J. Heering, K. Meinke, and B. Möller, editors, *Higher-order Algebra, Logic and Term Rewriting*, volume 1074 of *Lecture Notes in Computer Science*, pages 221–237. Springer-Verlag, 1995.
115. Fernando C.N. Pereira Stuart M. Shieber and Mary Dalrymple. Interactions of scope and ellipsis. *Linguistics and Philosophy*, 19:527–552, 1996.
116. D. A. Wolfram. The clausal theory of types. Doctoral thesis, University of Cambridge, 1989.
117. M. Zaionc. The regular expression description of unifier set in the typed λ-calculus. *Fundementa Informaticae*, X:309–322, 1987.
118. M. Zaionc. Mechanical procedure for proof construction via closed terms in typed λ-calculus. *Journal of Automated Reasoning*, 4:173–190, 1988.