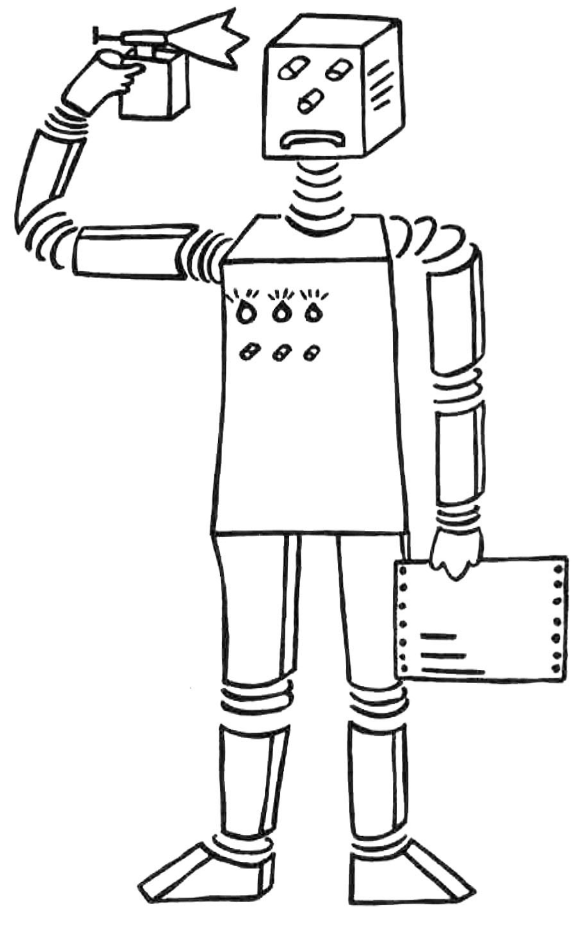
SEKI-REPORT

Artificial
Intelligence
Laboratories

Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern 1, W. Germany



Opening the AC-Unification Race

Hans-Juergen Buerckert et al.

SEKI Report SR-88-11
July 1988

Opening the AC-Unification Race

HANS-JÜRGEN BÜRCKERT
Universität Kaiserslautern, FB Informatik, Postfach 3049, D-6750 Kaiserslautern, F.R. Germany
net-address: buerckert@unido.uucp

ALEXANDER HEROLD
European Computer-Industry Research Centre, Arabellastr. 17, D-8000 München, F.R. Germany
net-address: herold@ecrcvax.uucp

DEEPAK KAPUR
Dept. of Computer Science, State University of New York at Albany, Albany, NY 12222, U.S.A.
netaddress: kapur@albanycs.albany.edu

JÖRG H. SIEKMANN
Universität Kaiserslautern, FB Informatik, Postfach 3049, D-6750 Kaiserslautern, F.R. Germany
net-address: siekmann@unido.uucp

MARK E. STICKEL
Artificial Intelligence Center, SRI International, Menlo Park, CA 94025, U.S.A.
net-address: stickel@ai.sri.com

MICHAEL TEPP
Universität Kaiserslautern, FB Informatik (AG Siekmann), Postfach 3049, D-6750 Kaiserslautern, F.R. Germany
net-address: tepp@unido.uucp

HANTAO ZHANG
Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180, U.S.A.
net-address: zhang@albanycs.albany.edu

Abstract: This note reports about the implementation of AC-unification algorithms, based on the variable-abstraction method of Stickel and on the constant-abstraction method of Livesey, Siekmann, and Herold. We give a set of 105 benchmark examples and compare execution times for implementations of the two approaches. This documents for other researchers what we consider to be the state-of-the-art performance for elementary AC-unification problems.

Key words: Theory unification, AC-unification, linear Diophantine equation

1. Introduction

> *"... the unification computation occurs at the very heart of most deduction systems. It is the addition and multiplication of deduction work. There is accordingly a very strong incentive to design the last possible ounce of efficiency into a unification program. The incentive is very much the same as that for seeking maximally efficient realisations of the elementary arithmetic operations in numerical computation – and the problem is every bit as interesting." J.A. Robinson, 1971*

Theory unification [5, 11, 26, 34, 44, 47, 48, 51], an extension of J.A. Robinson's standard unification [45], has many applications in computer science (see J.H. Siekmann's survey on unification theory [48]). In particular it is used in most current deduction systems:

- resolution based theorem proving systems [2, 8, 40, 42, 54]
- theorem proving systems based on algebraic completion [15, 22, 23, 24, 30, 31, 32, 41]
- term rewriting modulo a set of equations [29, 31, 32, 37, 43, 53]
- narrowing modulo a set of equations [31, 32, 35, 41]
- logic programming [3, 16, 17, 28, 49]

Currently free Abelian semigroups, i.e., equational theories for associative and commutative (AC-) function symbols, are among the most important theories [9, 10, 11, 12, 14, 19, 21, 27, 33, 38, 39, 50, 52, 55].

2. AC-Unification

AC-unification, that is unification of terms containing AC-function symbols, is the problem of solving equations in free Abelian semigroups. This can be done by a reduction to AC1-unification, the theory of associative, commutative functions with a unit, known as free Abelian monoids. Solving AC1-unification problems in turn is equivalent to solving some corresponding linear Diophantine equations that reflect the number of occurrences of the constants and variables in an AC1-solution.

There are essentially two different approaches to solve elementary AC-unification problems that are built by one AC-function symbol, free constants and variables. The first method (due to M.E. Stickel [50]) is to replace the constants by variables (*variable-abstraction*) and then to solve the corresponding homogeneous linear Diophantine equation [9, 10, 12, 13, 14, 27, 38, 50, 52, 55]. Since there are too many solutions (different constants may be identified, when they are replaced by variables), the solutions have to be post-processed to remove conflicting constants. It is also possible to solve the Diophantine equation under certain constraints on the variables, such that the conflicting constants are not produced in the first place. In the second approach (due to M. Livesey and J.H. Siekmann [39]) these constraints are treated explicitly (*constant-abstraction*). This is achieved by a homogeneous equation for the variables and in addition by certain inhomogeneous equations for the constants [20, 21, 39].

In order to demonstrate the two approaches we assume a binary infix function "·" as the AC-function symbol. We use a more convenient representation; terms are flattened (parentheses and dots are omitted) to strings, the remaining variables and constants are ordered, and multiple occurrences are represented by an exponent:

$$((a \cdot x) \cdot (b \cdot (c \cdot ((b \cdot x) \cdot x)))) \text{ is represented as } ab^2cx^3.$$

Now consider the AC1-unification problem $\langle\, ab^2x^3 = c^2yz^3 \,\rangle$. Any solution of this problem has the property that each variable and each constant has to occur the *same* number of times on both sides of the equation. For a most general solution we can also assume that all problem variables are substituted and that the solution introduces only new variables but no new constants. Hence the following equations hold for the occurrences X_v, Y_v, Z_v of a variable v and for the occurrences X_d, Y_d, Z_d of a constant $d \in \{a, b, c\}$ in the solution:

- $3X_v$	$=$	$Y_v + 3Z_v$	for each new variable v
- $3X_a + 1$	$=$	$Y_a + 3Z_a$	for the constant a
- $3X_b + 2$	$=$	$Y_b + 3Z_b$	for the constant b
- $3X_c$	$=$	$Y_c + 3Z_c + 2$	for the constant c

Solving these linear Diophantine equations in non-negative integers corresponds to solving the above AC1-unification problem. In order to obtain the corresponding AC-solutions we instantiate every subset of the newly introduced variables of each AC1-solution by the unit.

The number of AC-solutions grows exponentially in the number of introduced variables of the AC1-solutions. For example the problem $\langle xyz = u^4 \rangle$ has exactly one most general AC1-unifier, introducing 15 new variables, hence the minimal AC-solution set contains about 32 000 (2^{15}) independent AC-unifiers (see the table below).

3. A Set of 105 Benchmark Problems

During a visit at the University of Kaiserslautern, M.E. Stickel, together with A. Herold and J.H. Siekmann, designed some benchmark problems (105 examples, see table below) for elementary AC-unification. A test of the Stickel implementation and the Herold/Siekmann implementation showed that Stickel's was much faster. This motivated H.-J. Bürckert and M. Tepp to improve the Herold/Siekmann implementation by speeding up the computation of AC-unifiers from the solutions of the Diophantine equations. An extra stimulus for improvement were the favorable timing figures for the Rewrite Rule Laboratory's AC-unification algorithm of D. Kapur and H. Zhang, which is essentially an implementation of Stickel's constrained variable-abstraction method with some additional heuristics [31, 32]. Each of the implementations has now gone through a few additional iterations of improvement in response to performance gains by the other.

This note reports the current performance of the Stickel, the improved Herold/Siekmann, and the Kapur/Zhang implementation. We think the results demonstrate state-of-the-art performances for implementations of AC-unification that are embedded in larger theorem proving systems. All three implementations could be sped up by adopting different data structures for representing the unifiers that would improve benchmark performance but are incompatible with their use in the larger systems. The implementations are capable of handling the general case of AC-unification that includes arguments that are not variables or constants, which do not appear in this set of benchmark problems. Not all other known inefficiency or extraneous functionality has been removed.

The examples consist of all pairs of terms such that the first term has three variables or constants and the second term has four variables and constants and neither term has only constants (so that we are always testing AC-unification rather than AC-matching). Among them is problem "acuni-025", which was used by M.E. Stickel to illustrate AC-unification in the first paper on this topic [50] and has appeared in most papers on AC-unification since.

4. Results

The following table shows the result of the comparison of Stickel's implementation, the improved Herold/Siekmann implementation, and the implementation of Kapur and Zhang. The algorithms were implemented in Common-Lisp (Stickel and Herold/Siekmann) and Zeta-Lisp (Kapur/Zhang) and run on Symbolics 36xx with instruction fetch unit.

The second column in the table below lists the examples (x, y, z, u, v, w, t are variables, while a, b, c, d, e are constants). The third column contains the number of AC-unifiers for the given problem (in parentheses the number of AC1-unifiers). Column four gives Stickel's CPU-time, column five the CPU-time of the improved Herold/Siekmann implementation (in parentheses the time for solving the Diophantine equations), and column six that of the Kapur/Zhang implementation (all CPU-times are in seconds). Example "acuni-097" needed unreasonable time and space (it has 1,044,569 AC-unifiers, as determined by Zhang and Kapur and verified by Stickel).

In order to reduce machine effects like paging we measured several runs for each example and took the CPU-time of the fastest one.

example	problem	number of unifiers AC (AC1)	Stickel AC-time	Herold & Siekmann AC-time (AC1-time)	Kapur & Zhang AC-time
acuni-001	xab = ucde	2 (1)	0.018	0.009 (0.005)	0.012
acuni-002	xab = uccd	2 (1)	0.011	0.010 (0.006)	0.011
acuni-003	xab = uccc	2 (1)	0.008	0.010 (0.005)	0.008
acuni-004	xab = uvcd	12 (4)	0.047	0.031 (0.005)	0.031
acuni-005	xab = uvcc	12 (4)	0.032	0.030 (0.006)	0.029
acuni-006	xab = uvwc	30 (9)	0.096	0.084 (0.007)	0.064
acuni-007	xab = uvwt	56 (16)	0.171	0.230 (0.007)	0.127
acuni-008	xab = uucd	2 (1)	0.018	0.009 (0.005)	0.012
acuni-009	xab = uucc	2 (1)	0.011	0.009 (0.004)	0.008
acuni-010	xab = uuvc	12 (4)	0.040	0.030 (0.006)	0.032
acuni-011	xab = uuvw	30 (9)	0.075	0.087 (0.007)	0.068
acuni-012	xab = uuvv	12 (4)	0.030	0.029 (0.005)	0.022
acuni-013	xab = uuuc	2 (1)	0.013	0.010 (0.005)	0.009
acuni-014	xab = uuuv	12 (4)	0.027	0.030 (0.005)	0.022
acuni-015	xab = uuuu	2 (1)	0.008	0.009 (0.004)	0.007
acuni-016	xaa = ucde	2 (1)	0.013	0.009 (0.003)	0.011
acuni-017	xaa = uccd	2 (1)	0.009	0.011 (0.006)	0.009
acuni-018	xaa = uccc	2 (1)	0.006	0.010 (0.005)	0.009
acuni-019	xaa = uvcd	8 (3)	0.032	0.025 (0.006)	0.023
acuni-020	xaa = uvcc	8 (3)	0.020	0.024 (0.008)	0.016
acuni-021	xaa = uvwc	18 (6)	0.062	0.058 (0.009)	0.052
acuni-022	xaa = uvwt	32 (10)	0.114	0.144 (0.011)	0.102
acuni-023	xaa = uucd	2 (1)	0.009	0.008 (0.005)	0.009
acuni-024	xaa = uucc	2 (1)	0.006	0.009 (0.005)	0.006
acuni-025	xaa = uuvc	4 (2)	0.012	0.016 (0.007)	0.012
acuni-026	xaa = uuvw	10 (4)	0.025	0.038 (0.008)	0.022
acuni-027	xaa = uuvv	4 (2)	0.008	0.014 (0.006)	0.008
acuni-028	xaa = uuuc	2 (1)	0.007	0.009 (0.004)	0.008
acuni-029	xaa = uuuv	4 (2)	0.010	0.014 (0.005)	0.011
acuni-030	xaa = uuuu	2 (1)	0.005	0.007 (0.005)	0.005
acuni-031	xya = ucde	28 (8)	0.094	0.060 (0.006)	0.056
acuni-032	xya = uccd	20 (6)	0.050	0.045 (0.008)	0.039
acuni-033	xya = uccc	12 (4)	0.026	0.032 (0.010)	0.020
acuni-034	xya = uvcd	88 (8)	0.247	0.195 (0.007)	0.139
acuni-035	xya = uvcc	64 (6)	0.133	0.148 (0.009)	0.115
acuni-036	xya = uvwc	204 (6)	0.538	0.546 (0.009)	0.314
acuni-037	xya = uvwt	416 (4)	1.046	1.365 (0.010)	0.657
acuni-038	xya = uucd	60 (8)	0.154	0.120 (0.007)	0.112
acuni-039	xya = uucc	44 (6)	0.082	0.093 (0.008)	0.065
acuni-040	xya = uuvc	144 (6)	0.329	0.322 (0.009)	0.246
acuni-041	xya = uuvw	300 (4)	0.622	0.766 (0.008)	0.507
acuni-042	xya = uuvv	216 (4)	0.347	0.473 (0.008)	0.314
acuni-043	xya = uuuc	92 (6)	0.166	0.197 (0.008)	0.140
acuni-044	xya = uuuv	196 (4)	0.323	0.464 (0.010)	0.276
acuni-045	xya = uuuu	124 (4)	0.163	0.291 (0.009)	0.160

example	problem	number of unifiers AC (AC1)	Stickel AC-time	Herold & Siekmann AC-time (AC1-time)	Kapur & Zhang AC-time
acuni-046	xyz = ucde	120 (27)	0.320	0.281 (0.006)	0.209
acuni-047	xyz = uccd	75 (18)	0.168	0.186 (0.011)	0.100
acuni-048	xyz = uccc	37 (10)	0.073	0.104 (0.016)	0.049
acuni-049	xyz = uvcd	336 (9)	0.840	0.843 (0.008)	0.517
acuni-050	xyz = uvcc	216 (6)	0.431	0.549 (0.012)	0.293
acuni-051	xyz = uvwc	870 (3)	2.102	2.428 (0.010)	1.316
acuni-052	xyz = uvwt	2161 (1)	5.030	7.086 (0.008)	3.308
acuni-053	xyz = uucd	486 (9)	0.996	1.082 (0.008)	0.453
acuni-054	xyz = uucc	318 (6)	0.513	0.714 (0.010)	0.264
acuni-055	xyz = uuvc	1200 (3)	2.339	2.929 (0.010)	1.014
acuni-056	xyz = uuvw	2901 (1)	5.435	7.936 (0.009)	2.330
acuni-057	xyz = uuvv	3825 (1)	5.673	8.913 (0.008)	2.584
acuni-058	xyz = uuuc	2982 (3)	4.730	7.103 (0.011)	1.701
acuni-059	xyz = uuuv	7029 (1)	10.695	19.321 (0.009)	4.615
acuni-060	xyz = uuuu	32677 (1)	39.865	98.544 (0.010)	19.136
acuni-061	xxa = ucde	2 (1)	0.017	0.010 (0.005)	0.012
acuni-062	xxa = uccd	2 (1)	0.009	0.010 (0.005)	0.009
acuni-063	xxa = uccc	2 (1)	0.006	0.010 (0.005)	0.009
acuni-064	xxa = uvcd	60 (8)	0.147	0.128 (0.007)	0.128
acuni-065	xxa = uvcc	12 (2)	0.027	0.032 (0.006)	0.023
acuni-066	xxa = uvwc	486 (9)	0.968	1.132 (0.010)	1.165
acuni-067	xxa = uvwt	3416 (4)	6.426	8.547 (0.011)	12.306
acuni-068	xxa = uucd	0 (0)	0.004	0.003 (0.003)	0.006
acuni-069	xxa = uucc	0 (0)	0.003	0.005 (0.004)	0.004
acuni-070	xxa = uuvc	2 (1)	0.010	0.011 (0.005)	0.009
acuni-071	xxa = uuvw	12 (2)	0.028	0.042 (0.007)	0.026
acuni-072	xxa = uuvv	0 (0)	0.003	0.004 (0.003)	0.004
acuni-073	xxa = uuuc	2 (1)	0.008	0.009 (0.005)	0.008
acuni-074	xxa = uuuv	12 (2)	0.019	0.032 (0.006)	0.018
acuni-075	xxa = uuuu	0 (0)	0.002	0.003 (0.003)	0.004
acuni-076	xxy = ucde	28 (8)	0.074	0.061 (0.005)	0.043
acuni-077	xxy = uccd	11 (4)	0.025	0.028 (0.007)	0.017
acuni-078	xxy = uccc	7 (3)	0.013	0.021 (0.006)	0.014
acuni-079	xxy = uvcd	228 (9)	0.515	0.508 (0.007)	0.225
acuni-080	xxy = uvcc	44 (2)	0.081	0.101 (0.007)	0.043
acuni-081	xxy = uvwc	1632 (4)	3.228	3.871 (0.010)	1.368
acuni-082	xxy = uvwt	13703 (1)	25.605	36.690 (0.010)	13.186
acuni-083	xxy = uucd	2 (1)	0.007	0.010 (0.005)	0.009
acuni-084	xxy = uucc	4 (2)	0.007	0.014 (0.006)	0.008
acuni-085	xxy = uuvc	18 (2)	0.034	0.047 (0.007)	0.027
acuni-086	xxy = uuvw	69 (1)	0.115	0.220 (0.005)	0.083
acuni-087	xxy = uuvv	7 (1)	0.011	0.023 (0.004)	0.011
acuni-088	xxy = uuuc	12 (2)	0.021	0.031 (0.007)	0.015
acuni-089	xxy = uuuv	47 (1)	0.057	0.110 (0.006)	0.037
acuni-090	xxy = uuuu	5 (1)	0.007	0.015 (0.004)	0.008

example	problem	number of unifiers AC (AC1)	Stickel AC-time	Herold & Siekmann AC-time (AC1-time)	Kapur & Zhang AC-time
acuni-091	xxx = ucde	2 (1)	0.013	0.009 (0.004)	0.009
acuni-092	xxx = uccd	2 (1)	0.008	0.009 (0.005)	0.008
acuni-093	xxx = uccc	1 (1)	0.002	0.006 (0.004)	0.003
acuni-094	xxx = uvcd	140 (9)	0.254	0.314 (0.007)	0.092
acuni-095	xxx = uvcc	28 (2)	0.043	0.069 (0.007)	0.025
acuni-096	xxx = uvwc	6006 (6)	9.499	14.671 (0.012)	3.218
acuni-097	xxx = uvwt	1044569 (1)	*	* (0.013)	639.640
acuni-098	xxx = uucd	2 (1)	0.008	0.008 (0.004)	0.008
acuni-099	xxx = uucc	2 (1)	0.004	0.009 (0.004)	0.005
acuni-100	xxx = uuvc	12 (2)	0.023	0.033 (0.007)	0.016
acuni-101	xxx = uuvw	101 (1)	0.130	0.287 (0.007)	0.075
acuni-102	xxx = uuvv	13 (1)	0.013	0.033 (0.005)	0.010
acuni-103	xxx = uuuc	0 (0)	0.002	0.003 (0.003)	0.002
acuni-104	xxx = uuuv	1 (1)	0.003	0.007 (0.003)	0.003
acuni-105	xxx = uuuu	1 (1)	0.002	0.005 (0.002)	0.003

5. Conclusion

Bürckert and Tepp measured 2–16 msec for their Diophantine equation solving algorithm, while the whole computation ranged from 2 msec to 98 sec (or more than 10 min for example "acuni-097"). This shows the exponential explosion in the number of AC-solutions and demonstrates that investigations of fast Diophantine equation solving [6, 7, 11, 18, 25, 36, 59] cannot really improve AC-unification – at most they will hasten AC1-unification.

Hence we propose to take the AC1-unifiers or the solutions of the Diophantine equations as a representation for the AC-unifiers (see [4] for a more detailed discussion). A main problem with AC1-unification, however, has been that there was no extension for AC1-unification with free function symbols or with other theory unification algorithms, because of the collapse equation $1x = x$ specifying the unit (see [19, 20, 33, 56, 57, 58] for the combination problem of unification algorithms for regular, collapse free theories). Recent research by M. Schmidt-Schauß [46], however, gives a solution to the problem of combining arbitrary disjoint equational theories, and for the extension of arbitrary theory unification algorithms to free function symbols (see also [1]).

We finally want to emphasize that the benchmarks are just for elementary AC-unification problems, but most applications have at least additional free function symbols. Moreover the difference between the two approaches becomes particularily visible when such function symbols are involved. In this case Stickel abstracts "alien" subterms temporarily by variables and computes the solutions for the abstraction. A post-process generates the final solutions from the "pure" variable AC-problem and the abstraction (this can again be supported by constraints on the Diophantine equations). Herold and Siekmann on the other hand abstract the alien subterms by distinct "new" free constants and thus obtain subproblems, where different alien subterms have to be identified (see [55] for a more detailed discussion of these differences). Here both methods strongly differ and we would like to see a comparison of both approaches that determines which is better for which class of problems. Hence we need some benchmarks for non-elementary AC-unification problems.

Acknowledgements: Kapur and Zhang would like to acknowledge Sivakumar, who did the first implementation of Stickel/Fortenbacher's AC-unification algorithm in RRL. Numberless hints and ideas of H.J. Ohlbach were very useful in improving the Lisp code of the Herold/Siekmann implementation. N. Eisinger carefully read some former draft of this note.

This work was partially funded by the Deutsche Forschungsgemeinschaft, Sonderforschungsbereich 314 (H.-J. Bürckert), by the Siemens AG (M. Tepp), by the National Science Foundation under grants CCR-811116 (M.E. Stickel) and CCR-8408461 (D. Kapur, H. Zhang).

References

1. Boudet, A., Jouannaud, J.-P., Schmidt-Schauß, M., 'Unification in Boolean Rings and Abelian Groups', *Proc. of Conf. on Logic in Computer Science*, Edinburgh, to appear (1988)
2. Boy de la Tour, T., Caferra, R., Chaminade, G., 'Some Tools for an Inference Laboratory (ATINF)', *Proc. of 9th Int. Conf. on Automated Deduction, Springer LNCS* **310**, p. 744-745 (1988)
3. Bürckert, H.-J., 'Lazy Theory Unification in PROLOG: An Extension of the Warren Abstract Machine', *Proc. of German Workshop on Artificial Intelligence, Springer Fachberichte* **124**, p. 277-288 (1986)
4. Bürckert, H.-J., 'Solving Disequations in Equational Theories', *Proc. of 9th Int. Conf. on Automated Deduction, Springer LNCS* **310**, p. 517-526 (1988). See also: *SEKI-Report SR-87-15*, Universität Kaiserslautern (1987)
5. Bürckert, H.-J., Herold, A., Schmidt-Schauß, M., 'On Equational Theories, Unification, and Decidability', *Proc. of Int. Conf. on Rewriting Techniques and Applications, Springer LNCS* **256**, p. 204-215 (1987). See also:*J. of Symb. Comp.*, Kirchner, C. (ed.), *Special Issue on Unification*, to appear (1988)
6. Büttner, W., 'Unification in the Datastructure Multisets', *J. of Automated Reasoning* **2**, p. 75-88 (1986)
7. Clausen, M., Fortenbacher, A., 'Efficient Solution of Linear Diophantine Equations', *Internal Report* 32/87, Universität Karlsruhe (1987)
8. Eisinger, N., Ohlbach, H.J., 'The Markgraf Karl Refutation Procedure', in *Proc. of 8th Int. Conf. on Automated Deduction, Springer LNCS* **230**, p. 682-683 (1986)
9. Fages, F., 'Formes Canoniques dans les Algèbres Booléennes, et Application à la Démonstration Automatique en Logique de Premier Ordre', *Thèse de 3éme Cycle* (in French), Université Paris VI (1983)
10. Fages, F., 'Associative-Commutative Unification', *Proc. of 7th Int. Conf. on Automated Deduction, Springer LNCS* **170**, p. 194-208 (1984). See also: *J. of Symb. Comp.* **3**, (1987)
11. Fages, F., Huet, G., 'Complete Sets of Unifiers and Matchers in Equational Theories', *Proc. of CAAP'83, Springer LNCS* **159**, p.205-220 (1983). See also: *J. of Theoret. Comp. Sci.* **43**, p. 189-200 (1986)
12. Fortenbacher, A., 'Algebraische Unifikation', *Diplomarbeit* (in German), Universität Karlsruhe (1983)
13. Fortenbacher, A., 'An Algebraic Approach to Unification under Associativity and Commutativity', *Proc. of Int. Conf. on Rewriting Techniques and Applications, Springer LNCS* **202**, p.381-397 (1985). See also: *J. of Symb. Comp.* **3**, p. 217-229 (1987)
14. Franzen, M., Henschen, L.J., 'A New Approach to Universal Unification and Its Application to AC-Unification', *Proc. of 9th Int. Conf. on Automated Deduction, Springer LNCS* **310**, p.643-657 (1988)
15. Fribourg, L., 'A Superposition Oriented Theorem Prover', *J. of Theoret. Comp. Sci.* **35**, p.124-164 (1985)
16. Gallier, J., Raatz, S., 'SLD-Resolution Methods for Horn Clauses with Equality Based on E-Unification', *Proc. of Symp. on Logic Programming*, p. 168-179, (1986)
17. Goguen, J.A., Meseguer, J., 'EQLOG - Equality, Types, and Generic Modules for Logic Programming', in: DeGroot, D., Lindstrom, G. (eds.), *Logic Programming: Functions, Relations, and Equations*, Prentice Hall, p. 295-363 (1986)
18. Guckenbiehl, Th., Herold, A., 'Solving Linear Diophantine Equations', *MEMO-SEKI 85-IV-KL*, Universität Kaiserslautern (1985)
19. Herold, A., 'Combination of Unification Algorithms', *Proc. of 8th Int. Conf. on Automated Deduction, Springer LNCS* **230**, p.450-469 (1986).
20. Herold, A., 'Combination of Unification Algorithms in Equational Theories', *Dissertation*, Universität Kaiserslautern (1987)
21. Herold, A., Siekmann, J.H., 'Unification in Abelian Semigroups', *J. of Automated Reasoning* **3**, p. 247-284 (1987)
22. Hsiang, J., 'Topics in Automated Theorem Proving and Program Generation', *Ph. D. Thesis*, University of Illinois at Urbana-Champaign (1982)
23. Hsiang; J., 'Two Results in Term Rewriting Theorem Proving', *Proc. of Int. Conf. on Rewrite Techniques and Applications, Springer LNCS* **202**, p. 301-324 (1985)
24. Hsiang, J., Dershowitz, N., 'Rewrite Methods for Clausal and Non-Clausal Theorem Proving', *Proc. of 10th ETACS Int. Coll. on Automata, Languages, and Programming (ICALP)*, (1983)
25. Huet, G., 'An Algorithm to Generate the Basis of Solutions to Homogeneous Linear Diophantine Equations', *Information Processing Letters* **7**, p. 144-147 (1978)
26. Huet, G., Oppen, D.C., 'Equations and Rewrite Rules. A Survey', in: Book, R. (ed.), *Formal Languages: Perspectives and Open Problems*, Academic Press (1980)
27. Hullot, J.M., 'Compilation des Formes Canoniques dans des Théories Equationelles', *Thèse du 3ème Cycle* (in French), Université de Paris-Sud (1980)
28. Jaffar, J., Lassez, J.-L., Maher, M., 'Logic Programming Language Scheme', in: DeGroot, D., Lindstrom, G. (eds.), *Logic Programming: Functions, Relations, Equations*, Prentice Hall (1986)

29. Jouannaud, J.-P., Kirchner, H., 'Completion of a Set of Rules Modulo a Set of Equations', *Proc. of 11th ACM Conf. on Principles of Programming Languages*, (1984)
30. Kapur, D., Narendran, P., 'An Equational Approach to Theorem Proving in First-Order Predicate Calculus', *Proc. of 7th Int. Joint Conf. on Artificial Intelligence*, Los Angeles, p. 1146-1153 (1985)
31. Kapur, D., Zhang, H., 'RRL: A Rewrite Rule Laboratory – A User's Manual', General Electric , (1987)
32. Kapur, D., Zhang, H., 'RRL: A Rewrite Rule Laboratory', *Proc. of 9th Int. Conf. on Automated Deduction, Springer LNCS* **310**, p. 768-769 (1988)
33. Kirchner, C., 'Methodes et Outils de Conception Systematique d'Algorithmes d'Unification dans les Théories Equationelle', ***Thèse de Doctorat d'Etat*** (in French), Université de Nancy (1985)
34. Kirchner, C. (ed.), 'Special Issue on Unification', *J. of Symb. Comp.*, to appear (1988)
35. Kirchner, C., Kirchner, H., 'Implementation of a General Completion Procedure Parametrized by Built-in Theories and Strategies', *Proc. of EUROCAL Conf.* (1985)
36. Lankford, D., 'A New Non-negative Integer Basis Algorithm for Linear Homogeneous Equations with Integer Coefficients', unpublished (1985)
37. Lankford, D., Ballantyne, R.M., Decision Procedures for Simple Equational Theories with Commutative-Associative Axioms: Complete Sets of Commutative-Associative Reductions', *Internal Report ATP-39*, University of Texas, Austin (1977)
38. Lincoln, P., Christian, J., 'Adventures in Associative-Commutative Unification (A Summary)', *Proc. of 9th Int. Conf. on Automated Deduction, Springer LNCS* **310**, p. 358-367 (1988)
39. Livesey, M., Siekmann, J.H., Unification of AC-Terms (Bags) and ACI-Terms (Sets)', *Internal Report*, University of Essex (1975) and Universität Karlsruhe (1976)
40. MKRP, 'The Markgraph Karl Refutation Procedure', *Internal Report*, Universität Kaiserslautern (1984)
41. Müller, J., 'THEOPOGLES - A Theorem Prover Based on First-Order Polynomials and a Special Knuth-Bendix Procedure', *Proc. of German Workshop on Artificial Intelligence, Springer Fachberichte* **152**, p. 241-250 (1987)
42. Ohlbach, H.J., 'Link Inheritance in Abstract Clause Graphs', *J. of Automated Reasoning* **3**, p. 1-34 (1987)
43. Peterson, G.E., Stickel, M.E., 'Complete Sets of Reductions for Equational Theories with Complete Unification Algorithms', *JACM* **28**, p. 322-364 (1981)
44. Plotkin, G., 'Building in Equational Theories', *Machine Intelligence* **7**, p. 73-90 (1972)
45. Robinson, J.A., 'A Machine Oriented Logic Based on the Resolution Principle', *JACM* **12**, p. 23-41 (1965)
46. Schmidt-Schauß, M., 'Combination of Arbitrary Disjoint Equational Theories', *Proc. of 9th Int. Conf. on Automated Deduction, Springer LNCS* **310**, p. 378-396 (1988). See also: *J. of Symb. Comp.*, Kirchner, C. (ed.), *Special Issue on Unification*, to appear (1988)
47. Siekmann, J.H., 'Unification and Matching Problems', *Ph. D. Thesis*, Essex University (1978)
48. Siekmann, J.H., 'Unification Theory. A Survey', *J. of Symb. Comp.*, Kirchner, C. (ed.), *Special Issue on Unification*, to appear (1988)
49. Smolka, G., Nutt, W., Goguen, J.A., Meseguer, J., 'Order-sorted Equational Computation', *Proc. of CREAS Workshop*, Austin, Texas, to appear (1987)
50. Stickel, M.E. 'A Complete Unification Algorithm for Associative-Commutative Functions', *Proc. of 4th Int. Joint Conf. on Artificial Intelligence*, Tblisi, p. 71-82 (1975)
51. Stickel, M.E., 'Mechanical Theorem Proving and Artificial Intelligence Languages', *Ph. D. Thesis*, Carnegie-Mellon University (1977)
52. Stickel, M.E. 'A Unification Algorithm for Associative-Commutative Functions', *JACM* **28**, p. 423-434 (1981)
53. Stickel, M.E., 'A Case Study of Theorem Proving by the Knuth-Bendix Method Discovering that $X^3 = X$ implies Ring Commutativity', *Proc. of 7th Int. Conf. on Automated Deduction, Springer LNCS* **170**, p. 248-258 (1984)
54. Stickel, M.E., 'Automated Deduction by Theory Resolution', *J. of Automated Reasoning* **1**, p. 333-357 (1985)
55. Stickel, M.E., 'A Comparison of the Variable-Abstraction and Constant-Abstraction Methods for Associative- Commutative Unification', *J. of Automated Reasoning* **3**, p. 285-289 (1987)
56. Tiden, E., 'Unification in Combinations of Equational Theories', *Ph. D. Thesis*, Stockholm (1986)
57. Tiden, E., 'Unification in Combinations of Collapse-Free Theories with Disjoint Sets of Function Symbols', *Proc. of 8th Int. Conf. on Automated Deduction, Springer LNCS* **230**, p.431-450 (1986)
58. Yellick, K., 'Combining Unification Algorithms for Confined Regular Equational Theories', *Proc. of Int. Conf. on Rewriting Techniques and Applications, Springer LNCS* **202**, p. 365-380 (1985)
59. Zhang, H., 'An Efficient Algorithm for Simple Diophantine Equations', *Technical Report 87-26*, Dept. of Computer Science, RPI (1987)