UniMoK: A System for Combining Equational Unification Algorithms*

Stephan Kepser[1] and Jörn Richts[2]

[1] SfS, University of Tübingen
Wilhelmstr. 133, 72074 Tübingen, Germany
kepser@sfs.nphil.uni-tuebingen.de
[2] Theoretische Informatik, RWTH Aachen
Ahornstr. 55, 52074 Aachen, Germany
richts@informatik.rwth-aachen.de

1 Combining Unification Algorithms

Equational unification algorithms can be used in resolution based theorem provers [9] and rewriting engines [6] to improve their handling of equality. Originally, the requirements of these theorem provers and rewrite engines were such that the unification algorithms had to compute complete sets of unifiers. But with the advent of constraint based approaches to theorem proving [4] and rewriting [8] the interest in unification algorithm that worked merely as decision procedures grew because minimal complete sets of unifiers can be very large – e.g., doubly exponential in the number of variables of the problem in the case of the theory AC – and are hence costly to compute.

Because actual unification problems usually contain function symbols from several different signatures, the following *combination problem* is an important task in unification theory: Given unification algorithms for equational theories $E_1, E_2, \ldots, E_n$ over pairwise disjoint signatures, provide a general method that gives a unification algorithm for the union $E_1 \cup E_2 \cup \ldots \cup E_n$ of these theories. Solutions for this problem were provided by Schmidt-Schauß [10] and Boudet [3] for the combination of algorithms calculating complete sets of unifiers and by Baader and Schulz [1] for combining decision procedures. The combination algorithm presented in [1] is mostly of theoretical interest, it contains many non-deterministic decisions, thus the search space that this algorithm spans is so huge, that it is unusable for practical implementations. Therefore the authors developed optimisation methods [7] for the combination algorithm by Baader and Schulz to gain an implementation that can be used in practise. This implementation is UniMoK.

UniMoK stands for UNification MOdule for KEIM. It contains algorithms for unification in certain equational theories and it provides several combination methods for them. All combination algorithms in UniMoK are extensions and optimisations of the combination method by Baader and Schulz [1].

* This work was supported by a DFG grant (SPP "Deduktion") and by the Esprit working group 22457 – CCL II of the EU.

P. Narendran and M. Rusinowitch (Eds.): RTA'99, LNCS 1631, pp. 248–251, 1999.

2 Basic Algorithms

The first aim of UniMoK is to provide an implementation of the combination method by Baader and Schulz and suitable component algorithms for some theories. The combination method of Baader and Schulz requires component algorithms that can solve so-called *E_i-unification problems with linear constant restrictions (LCR)* for the component theories E_i to be combined. An E_i-unification problem with LCR consists of a set of equations Γ and a linear order $<$ on the variables and constants of Γ where the terms in Γ are built from variables, free constants, and the signature of E_i. A substitution σ is a unifier of $(\Gamma, <)$ if $\sigma(s) =_{E_i} \sigma(t)$ for all equations $s \doteq t \in \Gamma$ and if $\sigma(x)$ does not contain any constant a with $x < a$ for all variables $x \in \Gamma$.

UniMoK offers algorithms for solving unification problems with LCR for the following equational theories:

- the free theory (syntactic unification),
- the theory A of an associative function symbol (with a depth bound, not the Makanin decision procedure),
- the theory AC of an associative and commutative function symbol,
- the theory ACI of an associative, commutative and idempotent function symbol,
- the theory BR of Boolean rings.

For most of these theories, the algorithm could be easily obtained by extending an existing algorithm for unification with constants. For the theory of Boolean rings, a method for constant elimination (see [10]) is used.

All these algorithms were implemented as decision procedures and as algorithms computing complete sets of unifiers. The implementation of the combination algorithm can cope with both kinds of algorithms, i.e., it can work as a decision procedure itself or compute complete sets of unifiers.

The combination algorithm of Baader and Schulz can also be used to combine constraint solvers for so-called quasi-free structures [2]. Unification algorithms are a special case of such constraint solvers. In order to use this property, component algorithms for rational trees and for feature structures were implemented.

3 Optimised Algorithms

The naive implementation of the combination method of Baader and Schulz is mostly for experimental purposes. Due to its large search space of non-deterministic choices this method is not useful for most practical problems. Therefore the authors developed an optimisation technique for this combination method called the deductive method [7]. The implementation of this deductive method is the central and most interesting part of UniMoK. In short, many decisions in the combination algorithms need not be non-deterministically guessed but can be deduced on the base of one of the component theories involved, the unification problem given, and other decisions already made. Hence the deductive

combination algorithm consults the component theories, if they can deduce that certain decisions have to be made deterministically in order for their subproblems to remain solvable. If a component returns such a decision, this decision might enable other components to deduce further decisions. If this process comes to an end before all decisions have been made, the combination algorithm has to make a non-deterministic choice. With this choice it can consult the components again.

The method obviously demands special deductive component algorithms that can deduce such decisions. An equational theory for which only a unification algorithm for problems with LCR exists, can still be used in this combination method but it does not contribute to the deductive process. UniMoK provides component algorithms computing decisions for the free theory and the theories A, AC, and ACI, for rational trees and feature structures. It also contains general algorithms for collapse-free and regular theories.

Due to its modular, object-oriented approach, UniMoK is simple to extend. To add a new equational theory E, one has to provide a method for deciding E-unification problems with LCR. In order to contribute to the optimisation, there should also be a method for participating in the deductive process. This is a method that takes a unification problem and a partially specified linear constant restriction as input and returns more information on the decisions to be made.

4 Implementation and Experimental Results

UniMoK is implemented in Common Lisp on base of the theorem prover development tool box Keim [5]. Keim is an open, modular, object-oriented system geared towards ease of use and extensibility, rapid prototyping and universality. It is not designed towards run-time efficiency. To use UniMoK, an installation of Keim is required. UniMoK, thereby, becomes a part of Keim, and theorem provers developed in Keim can use UniMoK for equational unification. In opposite to Keim, a major design goal behind UniMoK is the development of efficient code. Basically, all that is needed from Keim is the module for first order terms which could possibly be replaced with sustainable effort by something more efficient, if needed.

Experimental results show that it is crucial for the deductive combination method in which order the remaining non-deterministic decisions are selected. In [7] the authors presented the so-called iterative strategy, which chooses all non-deterministic decisions for one component first before proceeding to choices for the next component. Run time tests in [7] showed that this strategy is superior for the example problems used there.

However, new run time tests show that this is not true in general. The following table contains sets of example problems solved with the deductive combination method. The problems are randomly generated on the base of a signature containing several A, AC, ACI and free function symbols. Each set contains 200 problems; roughly half of the problems in each set are unifiable. All run times

are in seconds. The 'bktrk'-column gives the number of backtracking steps.

	Ded+Iter		Ded			Ded+Iter		Ded	
	time	bktrk	time	bktrk		time	bktrk	time	bktrk
1	816	1953	81	152	11	20	7	21	10
2	232	780	>1h		12	21	8	21	8
3	330	800	1158	1982	13	32	50	31	30
4	58	250	42	110	14	21	47	26	22
5	1362	3971	141	401	15	154	394	3931	12335
6	>1h		103	295	16	26	50	30	31
7	676	2217	189	689	17	319	1116	83	147
8	19	1	19	1	18	1250	2627	44	107
9	67	33	75	33	19	178	462	58	169
10	16	1	15	1	20	99	414	43	159

For some sets, e.g., sets 1, 5, and 6, the iterative strategy ('Ded+Iter') is much worse than the default strategy ('Ded') where all decisions of a certain kind (i.e., identifications; see [7]) are made first; for other sets (2, 3, 15) the iterative strategy is still superior.

Further experiments are needed to develop a strategy combining the strengths of both strategies. Making the component theories choose the next non-deterministic decision might be another option to enhance the selection strategy.

UniMoK is available at `http://www-lti.informatik.rwth-aachen.de/Forschung/unimok.html`.

References

1. Franz Baader and Klaus U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. *JSC*, 21:211–243, 1996.
2. Franz Baader and Klaus U. Schulz. Combination of constraint solvers for free and quasi-free structures. *Theoretical Computer Science*, 192:107–161, 1998.
3. Alexandre Boudet. Combining unification algorithms. *JSC*, 16(6):597–626, 1993.
4. Hans-Jürgen Bürckert. A resolution principle for clauses with constraints. In Mark E. Stickel, editor, *CADE-10*, pp. 178–192, LNAI 449, Springer, 1990.
5. Xiaorong Huang, Manfred Kerber, Michael Kohlhase, Erica Melis, Dan Nesmith, Jörn Richts, and Jörg H. Siekmann. Keim: A toolkit for automated deduction. In Alan Bundy, editor, *CADE-12*, pp. 807–810, LNAI 814, Springer, 1994.
6. Jean-Pierre Jouannaud and Hélène Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal on Computing*, 15:1155–1195, 1986.
7. Stephan Kepser and Jörn Richts. Optimisation techniques for combining constraint solvers. In Maarten de Rijke and Dov M. Gabbay, editors, *Frontiers of Combining Systems, FroCoS'98*. Kluwer Academic Publishers, 1998.
8. Claude Kirchner and Hélène Kirchner. Constrained equational reasoning. In Gaston H. Gonnet, editor, *Proceedings of SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation: ISSAC'89*, pages 382–389. ACM Press, 1989.
9. G.D. Plotkin. Building-in equational theories. *Machine Intelligence*, 7:73–90, 1972.
10. Manfred Schmidt-Schauß. Unification in a combination of arbitrary disjoint equational theories. *Journal of Symbolic Computation*, 8(1,2):51–99, 1989.