Franz Baader, Jörg Siekmann,
Wayne Snyder (editors):

6th International Workshop on Unification

Dagstuhl-Seminar-Report; 42
29.07.-31.07.92 (9231)

ISSN 0940-1121

Preface

This workshop was the sixth in a series of three-days meetings on unification and related topics, the previous ones having been in Val d'Ajol (France), Lambrecht (Germany), Leeds (UK), and Barbizon (France). As its predecessors, UNIF'92 was meant to be an opportunity to meet old and new colleagues, to present recent (even unfinished) work, and to discuss new ideas and trends in unification and related fields. In addition, these workshops are a good opportunity for young researchers and researchers working in related areas to get an overview of the current state of the art in unification theory.

The very positive response to our invitation has shown that unification theory still is a rather active research area. We had 52 participants from 8 countries, of which France and Germany provided the largest contingents. A travel grant by NSF made it possible to have a larger than usual participation from the US.

The program consisted of 10 sessions with short talks (15 or 25 minutes), followed by discussions, a lively panel discussion on past and future developments in unification theory, and system demonstrations.

The sessions were organized around the following topics:

- Type Reconstruction
- AC and AC1 Unification
- General E-unification and Narrowing
- Higher-Order Unification
- Generalizations of Unification
- Constraint Solving
- Feature and Order-Sorted Unification
- Combination Problems
- Unification in Specific Theories
- Complexity
- Applications

Dagstuhl castle and its staff provided a very convenient and stimulating environment, which greatly contributed to the success of the workshop.

List of Talks

Type-Reconstruction in the Second-Order Lambda-Calculus

A.J. Kfoury
Boston University, Boston, USA

The Second-Order Lambda-Calculus, also called System F, is a system that assigns type expressions to pure lambda terms. The motivation for System F and other type lambda-calculi is both foundational and pragmatic. One the one hand, they naturally arise in connection to proof systems in intuitionistic logic; on the other, they formalize various type disciplines that have been successfully incorporated in modern programming languages. In this talk we restrict our attention to System F. Not all lambda terms are typable in System F, i.e. can be assigned some type by the system. One outstanding open problem is "type-reconstruction" (TR) for System F, which is the problem of deciding whether an arbitrary lambda term M is typable in the system. "Strong type-reconstruction" (STR) is a version of TR where free variables in M may be assigned fixed, predefined types. Another problem related to the preceding two is "type-checking" TC), which asks: given term M and type t, can M be assigned type t by System F? We survey several recent results providing partial answers to TR, STR, and TC for System F. We discuss decidable and undecidable cases of these problems, relate them to different forms of unification, and suggest approaches for tackling outstanding open problems in this area.

Associative Commutative Matching Based on the Syntacticity of the AC Theory

Mohamed Adi, Claude Kirchner
INRIA Lorraine & CRIN, Nancy, France

We present a new Associative-Commutative matching algorithm. It is based on the syntacticity of associative-commutative theories. As shown by Tobias Nipkow, it is possible to built a matching algorithm from a resolvant presentation of an AC theory and we have shown how this algorithm can be improved in such a way that most redundant computations are avoided. The resulting algorithm has been implemented and, compared to the algorithms that solve the AC-matching problem using solving of inhomogenous linear Diophantine equations, it gives much better performances, in particular a first match is computed several orders of magnitudes faster.

AC1-Unification/Matching in Linear Logic Programming

Steffen Hölldobler, Josef Schneeberger, Michael Thielscher
TH Darmstadt

Linear logic programming is an approach to model changeable objects like situations or states in a first-order equational logic without the need to state frame axioms explicitely. To answer queries posed to a linear logic program requires to solve certain (special) AC1-unification and matching problems. In the talk, we will give decidability results, determine the type, and present unification algorithms for these problems.

Integrating AC1-Unification/Matching into the Process of Completion Modulo AC1

Martin Henz
DFKI Saarbrücken

In our approach to completion modulo $AC1$, we allow both undirected and directed equations to contain constraints. These constraints may—in addition to zero-disequations—contain $AC1$-unification problems. This allows us to integrate $AC1$-unification into the process of completion; we may store unification problems that appear hard to solve and wait until they can be simplified by applying a newly computed rewrite rule.

Counterexamples to Completeness Results for Basic Narrowing

Eric Haemon
Vrije Universiteit Amsterdam, Amsterdam,
Niederlande

Narrowing is a generalization of term rewriting. It can be used as an algorithm to determine whether two terms unify modulo a certain (C)TRS R. It can also be used as the operational semantics for a language which integrates functional and Horn-clause programming. Basic narrowing is a more efficient form of narrowing.

It has been conjectured that basic narrowing is complete for semi-complete TRS's (Yamamoto) and that basic conditional narrowing is complete for semi-complete orthogonal CTRS's (Giovannetti & Moiso). We have found counterexamples for these conjectures. Furthermore, we show that one of the assumptions in Hoelldobler's completeness proof for basic conditional narrowing is incorrect and we give a way of repairing this problem. We will give certain syntactical restrictions that make basic narrowing complete for semi-complete TRS's. Finally, we show that narrowing is complete for level-confluent CTRS's that may contain variables in the right-hand side of a rule that do not appear in its left-hand side.

Narrowing and Basic Forward Closures

Stefan Kurtz
Universität Bielefeld

Forward closures are a common notion in the field of term rewriting systems. They can be seen as a result of a partial evaluation process of the narrowing relation. Our idea is to restrict this evaluation process to the basic narrowing relation, which leads to the notion of basic forward closures. If one uses the basic forward closure of a term rewriting system in a narrowing procedure, one can discard all the positions, which are introduced by the right-hand sides of the term rewriting rules, thus leading to a narrowing procedure called left-to-right bottom-up narrowing. This procedure is complete and terminates, if the basic forward closure of the term rewriting system is finite.

Conditional Rewriting Presentations for General E-Unification

Bertrand Delsart
LIFIA, Grenoble, France

Using strictly resolvent (i.e. $\xrightarrow{+}_R \subseteq \longrightarrow_R \xrightarrow{*}{}_R^{\neq\Lambda}$) conditional rewriting presentations of equational theories leads to a new transformation rule. This rule defines an unifying framework for the existing topmost approaches to E-unification. Thus the development of common formal optimizations and implementation techniques is possible. Moreover, new algorithms can be expressed with this rule. For example, presentations based on different kinds of conditions lead to E-unification algorithms the behavior of which depends on

the axiom applied. We also present the main ideas of an efficient E-unification algorithm based on presentations the conditions of which contain only E-unification problems between variables.

Practical Unification of Higher-Order Patterns

Tobias Nipkow
TU München

Higher-Order Patterns (HOPs) are lambda-terms in beta-normal form whose free variables occur only in subterms of the form $F\ x_1 ... x_n$, where F is free and the x_i are distinct bound variables. Dale Miller showed that with respect to unification HOPs behave like first-order terms: unification is decidable and most general unifiers exist. We present three different versions of Miller's unification algorithm for HOPs:

- a succinct and high-level formulation in terms of transformation rules, - a recursive algorithm derived from the transformation rules, and - a version of the recursive algorithm using de Bruijn's notation.

Both the transformation rules and the recursive algorithms are close to their counterparts for first-order terms.

Minimal Modular Higher Order E-Unification

Franz Weber
Forschungszentrum Informatik Karlsruhe

Nearly all higher order unification algorithms which are used nowadays are a variant of the algorithm which was published in 1974 by Huet. Recently an extension of this algorithm to higher order E-unification was developped by Nipkow, Qian and Wang. The extended algorithm is able to incorporate arbitrary pseudolgebraic equational theories. Unfortunatly, the extended algorithm is no longer minimal in contrast to the original algorithm of Huet. The presentation analyzes the reasons for the lack of minimality and gives partial solutions for the problem. One reason is, that the E-simplification rule of the extended algorithm produces dependent solutions on different branches of the search tree. For this case a process will be described which filters out all solutions which are not preunifiers and depend on another solution. Also the E-imitation rule of the extended algorithm returns depedent solutions. The reasons for that may be found by analyzing a combined E-imitation and E-simplification rule. The branches generated by this combined rule may be statically computed and the dependency between those branches may be statically analyzed. In the case of C-unification and AC-unification half of the branches may be pruned due to such dependencies.

A Complete Transformation System for Polymorphic Higher-Order Unification

Ullrich Hustadt
MPI Saarbrücken

Polymorphic higher-order unification is a method for unifying terms in the polymorphically typed λ-calculus, that is, given a set of pairs of terms $S_0 = \{s_1 = t_2, \ldots, s_n = t_n\}$, called a unification problem, finding a substitution σ such that $\sigma(s_i)$ and $\sigma(t_i)$ are equivalent under the conversion rules of the calculus for all i, $1 \leq i \leq n$.

I present the method as a transformation system, i.e. as a set of schematic rules $U \Rightarrow U'$ such that any unification problem $\delta(U)$ can be transformed into $\delta(U')$ where δ is an instantiation of the meta-level variables in U and U'. By successive use of transformation rules one possibly obtains a solved unification problem with obvious unifier. I show that the transformation system is correct and complete, i.e. if $\delta(U) \Rightarrow \delta(U')$ is an instance of a transformation rule, then the set of all unifiers of $\delta(U')$ is a subset of the set of all unifiers of $\delta(U)$ and if $\mathcal{U}$ is the set of all unification problems that can be obtained from successive applications of transformation rules from an unification problem U, then the union of the set of all unifiers of all unification problems in $\mathcal{U}$ is the set of all unifiers of U.

The transformation rules presented here are essentially different from those in [GallierSnyder89] or [Nipkow90]. The correctness and completeness proofs are in lines with those of [GallierSnyder89].

A Combinatory Logic Rewriting Relation which Supports Narrowing

Marian Vittek
INRIA Lorraine & CRIN, Nance, France

Higher-order (equational) unification problems can be solved by means of combinatory logic theory. In this setting, one can hope to solve the unification problems using algebraic methods like narrowing. Unfortunately there is no known rewriting relation in the combinatory logic theory which decides the e-equality (equality induced by $\alpha\beta\eta$-equality in λ-calculus) and which is 'sufficiently algebraic' to get a narrowing based unification procedure.

In our approach we define a rewriting relation defined on terms, that needs to enrich the set of combinatory logic terms by λ-abstraction. On this enriched term algebra we have defined the rewriting relation consisting of the three weak reduction rules (from combinatory logic theory) and of a variant of the extensionality rule (from λ-calculus). This rewriting relation decides the e-equality between the combinatory logic terms and can be used as the base for a narrowing-like unification procedure.

The Decidability of Higher-Order Matching

David Wolfram
University of Oxford, Oxford, United Kingdom

We show that a group of matching problems in the third-order simply-typed λ-calculus is NP-Complete by a reduction from propositional satisfiability. Statman's mapping of higher-order unification problems to those in the pure simply-typed λ-calculus is also discussed, and used as a simplifying method.

The projection property is then introduced:

Does there exist a substitution π such that

$$head(\lambda x_1 \dots x_n.@(t_1, \dots, t_m)\pi) = x_i$$

where $i \in \{1, \dots n\}$?

If this property is undecidable, then higher-order matching is undecidable; if not, then a type restriction on variables in terms gives a group of decidable higher-order matching problems of arbitrarily high order.

Unification of Terms with Integer Exponents

Hubert Comon
University Paris-Sud, Paris, France

ρ-terms are ordinary terms in which some parts are allowed to be iterated along fixed paths. The number of iterations is part of the syntax of the terms and may include integer variables. There are restrictions in the ρ-terms if Chen&Hsiang: The iterated part should not itself contain iterated parts (no nested iterations). It is also forbidden to iterate terms containing variables. In this paper, we drop these two restriction along some special constructions. And we show that unification of such terms is decidable and finitary.

Negation Elimination in Equational Formulae

Hubert Comon, Maribel Fernández
CNRS and LRI, Paris, France

An equational formula is a first order formula over an alphabet $\mathcal{F}$ of function symbols and the equality predicate symbol. Such formulae are interpreted in the algebra $T(\mathcal{F})$ of finite trees. A unification problem is any equational problem which does not contain any negation (in particular, it should not contain any disequation). We give a terminating set of transformation rules such that an equational formula ϕ is (semantically) equivalent to a unification problem iff its irreducible form is a unification problem. This result can be formulated in another way: our set of transformation rules computes a finite complete set of "most general unifiers" for a formula each time such a finite set exists.

The above results are extended to quotients of the free algebra by a congruence $=_E$ which can be generated by a set of shallow permutative equations E.

Complement Problems and Tree Automata

Denis Lugiez
CRIN-INRIA, Nancy, France

Given a term t, a set of terms $R = \{t_1, \ldots, t_n\}$, to solve the complement problem $t \neq t_1 \ldots t \neq t_n$ is to find if there is a ground instance of t which is not a ground instance of any of the $t_i's$. We propose a new solution of this problem when some functions are associative and commutative and the $t_i's$ are linear. This solution relies on tree-automata which are a powerful tool to recognize regular tree languages. We describe some extensions to other theories and to some non-linear cases.

Difference Unification

David Basin
Max-Planck-Institut für
Informatik, Saarbrücken

Toby Walsh
Edinburgh University,
Edinburgh, Scotland

In this paper we introduce difference unification, a procedure that supports the general application of a rewrite procedure called rippling in theorem proving and term simplification. A difference unifier takes as inputs two terms (or formulas) s and t. It returns s and t annotated with wave-fronts (places where the terms differ), and a set of substitutions such that the skeleton of the annotated terms (that is, the terms formed by deleting all the differences) are equal under substitution. Although difference unification generalizes first-order unification, it is much more than unification. It is an attempt to make two terms identical not just by variable instantiation, but also by structure hiding; the hidden structure is the part of the term within the wave-front that serves to direct rippling. We will present a difference unification algorithm and proves various properties it possesses (like soundness and completeness). We will also identify some future directions like higher-order difference matching.

Unifying Cycles

Jörg Würtz
DFKI Saarbrücken

Two-literal clauses of the form $L \leftarrow R$ occur quite frequently in logic programs, deductive databases, and—disguised as an equation—in term rewriting systems. These clauses define a cycle if the atoms L and R are weakly unifiable, i.e., if L unifies with a new variant of
The obvious problem with cycles is to control the number of iterations through the cycle. In this paper we consider the cycle unification problem of unifying two literals G and
modulo a cycle. We review the state of the art of cycle unification and give new results for a special type of cycles called unifying cycles, i.e., cycles $L \leftarrow R$ for which there exists a substitution σ such that $\sigma L = \sigma R$. Altogether, these results show how the deductive process can be efficiently controlled for special classes of cycles without losing completeness.

Relative Simplification: A Unifying Principle for Constraint Programming

Gert Smolka
DFKI Saarbrücken

The constraint logic programming model is obtained from the conventional Horn clause model by replacing unification with constraint simplification over arbitrary structures. More recent frameworks for constraint programming (ALPS, CC, AKL, Hydra) require that entailment between constraints is tested for incrementally. Two of these frameworks (AKL, Hydra) provide for deep guards, which require incremental entailment checking between constraints and formulae with defined relations.

The talk will introduce the notion of relative simplification, which is a unifying principle behind the mentioned approaches to constraint programming. In particular, relative simplification defines a uniform operational interface between constraint systems and constraint programming frameworks.

Relative Simplification for and Independence of CFT

Ralf Treinen, Gert Smolka
DFKI Saarbrücken

CFT is a new constraint system providing records as logical data structure for constraint (logic) programming. It can be seen as a generalization of the rational tree system employed in Prolog II, where finer-grained constraints are used, and where subtrees are identified by keywords rather than by position.

CFT is defined by a first-order structure consisting of so-called feature trees. Feature trees generalize the ordinary trees corresponding to first-order terms by having their edges labeled with field names called features. The mathematical semantics given by the feature tree structure is complemented with a logical semantics given by five axiom schemes, which we conjecture to comprise a complete axiomatization of the feature tree structure.

We present a decision method for CFT, which decides entailment and disentailment between possibly existentially quantified constraints. Since CFT satisfies the independence property, our decision method can also be employed for checking the satisfiability of conjunctions of positive and negative constraints. This includes quantified negative constraints such as $\forall y \forall z(x \neq f(y, z))$.

Extensible Unification as Basis for the Implementation of CLP Languages

Christian Holzbaur
University of Vienna, Wien, Österreich

We address various aspects of the proposal to use user-defined extensible unification as the basic formalism for the implementation of constraint logic programming (CLP) languages. The close connection between unification theory and CLP, exhibited through the theoretical work of Jaffar et al., justifies the proposed step to make this link explicit and, particularly, operational.

The idea with extensible unification in the context of logic programming is that the user identifies the set of interpreted functors through the provision of a signature. The unification semantics of terms built from interpreted functors are specified in the form of predicates in the language whose unification part is to be extended.

If CLP languages are implemented via extensible unification, they will inherit the capability of being extended on the very same basis, leading to the attractive construction of towers of (metacircular) CLP languages. AC and word unification algorithms, for example, typically require the solution of Diophantine equations and systems thereof.

Order-Sorted Feature Theory Unification

Hassan Aït-Kaci, Andreas Podelski, Seth Copen Goldstein
DEC, Paris, France

Order-sorted feature (OSF) terms generalize first-order rational terms whereby constructors become partially ordered sorts, and argument positions become symbolic feature symbols. We add the notion of class to OSF terms in order to impose structural constraints on objects. This is realized thanks to a monotonic mapping from sorts to OSF terms. We call such a mapping an OSF theory. The use of structural constraints from an OSF theory in the normalization of an OSF term is called OSF *Theory Unification*. It allows objects to be implicitly constrained by their classes.

In this manner, we obtain a formal system that models record-like objects with recursive class definitions accommodating multiple inheritance, and equational constraints among feature paths, including self-reference. The problem of normalizing an object to fit class templates is undecidable in general. We propose a complete and efficient set of rules to perform this normalization whenever it may be done.

We also propose a weaker unification problem that is complete with respect to a specific class of formulas. We show the weaker problem to be decidable and give normalization rules that achieve OSF unification in almost linear time. We obtain a complete algorithm for the general OSF theory unification problem with the addition of just one rule. The complete set of rules always terminates on an inconsistent formula, but may diverge on some consistent ones.

Feature Algebras as Coalbegras: A Category Perspective on Unification

Bill Rounds
University of Michigan, Ann Arbor, USA

We investigate the notion of extensionality in feature algebras: an algebra is extensional if whenever two objects have the same features, then they are the same. We characterize the notion in terms of subsumption relationships induced by algebra homomorphisms. We prove representation theorems for extensional algebras, and we show how the notion of extensionality can be related to the same notion in (non-wellfounded) set theory, by using feature algebras to construct a set theory model, one in which Aczel's axiom (AFA) fails.

A Complete and Decidable Feature Theory

Rolf Backofen, Gert Smolka
DFKI Saarbrücken

Feature Graphs are a universal data structure employed in computational linguistics and logic programming. We present a complete and recursive axiomatization of the structure of all feature graphs. Our completeness proof exhibits a decision procedure for the first-order theory of feature graphs. Moreover, the axiomatization provides a handy characterization of the elementarily equivalent models. We present a rational tree model of the feature graph theory clarifying the relation ship between feature graphs and rational trees.

On Approaches to Order-Sorted Rewriting

Andreas Werner
Universität Karlsruhe

Order-sorted rewriting builds a nice framework to handle partially defined functions and subtypes. Differing from many-sorted rewriting, the critical pair lemma and Birkhoff's completeness theorem do not hold in general. To retain a critical pair lemma, order-sorted rewriting has been restricted to sort decreasing term rewriting systems. In the last year efforts have been made in order to get a more general critical pair lemma. In this talk a

new approach to order-sorted rewriting will be presented and will be compared it with the other ones.

Combination Techniques and Decision Problems for Disunification

Franz Baader
DFKI Saarbrücken

Klaus Schulz
CIS München

Previous work on combination techniques considered the question of how to combine unification algorithms for disjoint equational theories E_1, ..., E_n in order to obtain a unification algorithm for the union $E_1 \cup \ldots \cup E_n$ of the theories. Here we will introduce a variant of the combination algorithm given in [BS] which allows us to treat finite systems of equations and inequations. Our main result says that solvability of finite systems of equations and disequations with respect to $E_1 \cup \ldots \cup E_n$ is decidable if solvability of finite systems of equations and disequations under linear constant restrictions is decidable for the disjoint equational theories $E_i (i = 1, \ldots, n)$. This implies that the existential fragment of the theory of the (ground) term algebra modulo associativity of a finite numbert of function symbols is decidable, and a similar result follows for functions symbols which are associative and commutative. The first result seems to be new-it closes a gap between previous decidability results in [BS] and an undecidability result by R. Treinen [Tr]. The second result has been proved earlier by H. Comon [Co].

[BS] F. Baader, K.U. Schulz, "Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures", DFKI-Research Report RR-91-33, also in Proceedings of the 11th International Conference on Automated Deduction, LNAI 607, (1992), pp. 50-65
[Co] H. Comon "Unification and Disunification. Theories et Applications", PhC thesis, Institut National
Polytechnique de Grenoble, Grenoble, France, 1988. [Tr] R. Treinen, "A New Method for Undecidability Proofs of First Order Theories", J. Symbolic Computation 11 (1992)

Higher-Order E-Unification for Arbitrary Theories

Zhenyu Qian, Kang Wang
Universität Bremen

This paper presents an algorithm consisting of three transformation rules for pre-unification of simply typed λ-terms w.r.t. α, β and η conversions and an arbitrary first-order equational theory E. The algorithm is parameterized by E-unification algorithms that admit free function symbols. It is proved that the algorithm is complete if the given E-unification algorithm is complete.

The result is relevant to implementations of higher-order logic programming languages and higher-order proof systems.

Unification in a Combination of Equational Theories with Shared Constants and its Application to Primal Algebras

Christophe Ringeissen
CRIN-CNRS & INRIA-Lorraine, Nancy, France

We extend the results on combination of disjoint equational theories to combination of equational theories where the only function symbols shared are constants. This is possible because there exist finitely many proper shared terms (the constants) which can be assumed irreducible in any equational proof of the combined theory. We establish a connection between the equational combination framework and a more algebraic one. A unification algorithm provides a symbolic constraint solver in the combination of algebraic structures whose finite domains of values are non disjoint and correspond to constants. Primal algebras are particular finite algebras of practical relevance for manipulating hardware descriptions.

Unification problems modulo distributivity

Evelyne Contejean
University Paris-Sud, Paris, France

Unification modulo the two-sided distributivity of a symbol $*$ over a symbol $+$ in the term algebra $T(\{+, *\}, \times)$ is still an open problem. The syntactic approach used by Arnborg and Tiden for the one-sided distributivity is not possible for the two-sided one. We have proved that the solutions of some particular (called starry balanced) unification problems with distributivity have some strong properties: solving such a problem modulo D boils down to solve the same problem modulo AC1. Moreover we can discribe "almost all" solutions of a starry balanced problem thanks to a particular term algebra $T(\{+, \Box\})$ called structure algebra: solutions are "schematic instances" of the unique solution of the original problem modulo AC1.

Complexity of E-Unification Problems

Paliath Narendran
State University of New York, Albany, USA

Complexity issues in Unification have been investigated a great deal since Paterson and Wegman published their linear-time algorithm for standard unification. E-unification, or unification in the presence of an equational theory E, is much more complicated, most of the problems being undecidable in general. Research has so far concentrated on two major issues (i) E-unifiability where one only has to check whether there exists a unifier for the input terms, and (ii) computing a complete set of E-unifiers for terms especially when these sets are known to be always finite. In Kapur and Narendran (1989) we briefly surveyed the results and presented them in tabular form. The present talk updates that survey and discusses several significant open problems.

A Unification– and Object–Based Symbolic Computation System

Georgios D. Grivas
ETH Zürich, Schweiz

The most important of the primitive operations for symbolic computation is the matching of terms. The main factor for efficient rule-based programming is the number of unifications performed and attempted in the course of a computation. The main goal of this work is to speed up the pattern matching operation for the rule- and object-based symbolic computation system *AlgBench*. Unlike Mathematica, *AlgBench* is designed in an object-oriented way and supports also two-way pattern matching (unification). In the chosen computation model all evaluations are done by pattern matching. The core of the system is class-based. Every pattern object class represents a class of patterns and is a subclass of the class of the composite expressions. Thus, we have a clear design and new pattern object classes can be easily added to the system. Huet's and the mark and retract algorithms for standard unification as well as Stickel's algorithm for associative commutative unifications are implemented in an object-oriented style. We extend Mathematica's type-constrained pattern matching by taking into account inheritance information from a user-defined hierarchy of object types (heads of composite expressions). The argument unification is basically instance variable unification. In order to have efficiency in a rule- and object-based symbolic computation system the improvement of the pattern matching operation in an object-oriented way seems to be very appropriate.

Retrieving Library Functions by Unifying Types Modulo Linear Isomorphism

Mikael Rittri
Chalmers University, Göteborg, Sweden

An improved method to retrieve a library function via its Hindley/Milner type is presented. A function is retrieved if one can instantiate the bound variables of its type, and the free variables of the query type, to get linearly isomorphic types. By linear isomorphism is meant the isomorphisms that hold in any symmetric monoidal closed category; Soloviev

has shown that they are presented by five equational axioms:

$$\begin{aligned} A \times B &\cong B \times A \\ (A \times B) \times C &\cong A \times (B \times C) \\ 1 \times A &\cong A \\ (A \times B) \to C &\cong A \to (B \to C) \\ 1 \to C &\cong C \end{aligned}$$

A unification algorithm modulo this equivalence has been given by Narendran, Pfenning and Statman. I use it in a retrieval system for the functional language Lazy ML. Further details can be found in PMG report 66, Chalmers 1992, address as above.

Conditional Rewriting Modulo a Built-in Algebra

Jürgen Avenhaus, Klaus Becker
Universität Kaiserslautern

Many programming languages have built-in operations to enhance efficiency. Rewriting can be seen as a high-level programming language. But so far it was not clear how to integrate built-in operations and at the same time preserve the well known techniques for proving termination and confluence. We present a method to integrate a built-in algebra into conditional rewriting systems. First, equational specifications will get assigned a suitable semantics that takes into account the predefined structure and allows for partially defined functions. The interpretation of "semantically and syntactically defined mixed objects" is based on sort hierarchies. As a consequence of this sort hierarchy a great deal of classical rewriting theory can be carried over to our context. We can prove local confluence by considering critical pairs. Furthermore we can construct reduction orderings, that incorporate knowledge about the built-in algebra and can be used to prove termination of the rewriting system.

Undecidability of the Horn clause Implication problem

Jerzy Marcinkowski, Leszek Pacholski
University of Wroclaw, Wroclaw, Poland

We prove that the problem "given two horn clauses $\mathcal{H}_1 = (\alpha_1 \wedge \alpha_2 \rightarrow \beta)$ and $\mathcal{H}_2 = (\gamma_1 \wedge ... \wedge \gamma_k \rightarrow \delta)$, where $\alpha_1, \beta, \gamma_i, \delta$ are atomic formulas, decide if $\mathcal{H}_2$ is a consequence of $\mathcal{H}_1$" is not recursive.

The theorem follows from the series of more or less technical lemmas.

Definition 1. For a Horn clause $\mathcal{H} = (\alpha_1 \wedge \alpha_2 \rightarrow \beta)$, and a set G of ground clauses a G-$\mathcal{H}$-derivation tree is a tree labelled by unit clausses in such a way, that for each node t there exists a substitution σ with the property that the left and the right son of the node t are labelled by $\sigma(\alpha_1)$ and $\sigma(\alpha_2)$ respectively, and t is labelled by $\sigma(\beta)$, and moreover the leaves are labelled with elements of G.

Lemma 2. There exists a Horn clause $\mathcal{H} = (\alpha_1 \wedge \alpha_2 \rightarrow \beta)$, and a finite set G of ground unit clauses such that it is undecidable if for a given word w, there exists a finite G-$\mathcal{H}$-derivation tree with a branch w.

The next two lemmas have a technical character and say, that it is possible to force a derivation tree to contain a given branch (**Forcing Lemma**) and to hide the large uncontrolled term that appears in the root of a derivation (**Hiding Lemma**).

Sequential Signatures

Delia Kesner
CNRS and LRI, Paris, France

Sequentiality is a property of monotonic predicates over partial terms, related to the possibility of *systematically* expanding any term step-by-step in order to turn the predicate true. This work is concerned with the sequentiality of *sort predicates* in order sorted algebras, where each sort predicate $Sort_\delta$ characterizes the partial terms of sort δ. Monotonicity of sort predicates guarantees that each time sorts decrease, there is more and more chance to well type terms. A signature Σ is defined to be *sequential* if for every sort $\delta \in \Sigma$, the sort predicate $Sort_\delta$ is sequential.

The idea of sequentializing the type checking is that terms will need to be evaluated as far as *necessary* in order to satisfy a subsort constraint. In general, a few computation steps could be sufficient, without reducing terms to full normal forms.

In this talk we provide a decision procedure for sequentiality of signatures and we provide a compilation scheme which allow to efficiently decrease the sort information of any term. Our characterization of signatures becomes in this way a necessary and sufficient condition in order to perform efficient type verifications in order-sorted algebras.

Partial Unification for Ordered Theory Resolution

Peter Baumgartner
Universität Koblenz

Theory resolution is a kind of two-level reasoning, where the concept of "syntactic complementarity" is generalized to "semantic complementarity" under a given theory. Thus, for implementations we need a special "background reasoner" that implements the theory. We describe such reasoners on an abstract level as proof calculi.

We are interested in the automatical construction of such calculi from given theories. As a main result, we present a technique that allows to compile a given Horn theory into a (possibly infinite) set of inference rules. These inference rules can roughly be seen as an order-restricted version of unit-resulting resolution. The compilation technique works in the spirit of Knuth-Bendix completion by adding new inference rules to shortcut critical pairs. However, instead of equations it deals with general Horn theories.

In summary, we achieve a complete combination of ordinary ordered resolution and ordered unit resulting resolution for the Horn subset of the specification.

Dagstuhl-Seminar 9231: List of Participants

UniversitätHassan **Ait-Kaci**
Digital Equipment Corporation
Paris Research Laboratory
85 avenue Victor Hugo
F-92500 Rueil-Malmaison Cedex
France
hak@prl.dec.com
tel.: +33-1-47.14.28.24

Jürgen **Avenhaus**
Universität Kaiserslautern
FB Informatik
Postfach 3049
W-6750 Kaiserslautern
Germany
avenhaus@informatik.uni-kl.de
tel.: +49-631-205-26 33

Hans-Jürgen **Bürckert**
DFKI Saarbrücken
Stuhlsatzenhausweg 3
W-6600 Saarbrücken 11
Germany
hjb@dfki.uni-sb.de
tel.: +49-681-302-53 21

Franz **Baader**
DFKI Saarbrücken
Stuhlsatzenhausweg 3
W-6600 Saarbrücken 11
Germany
baader@dfki.uni-sb.de
tel.: +49-681-302-53 19

Rolf **Backofen**
DFKI Saarbrücken
Stuhlsatzenhausweg 3
W-6600 Saarbrücken 11
Germany
backofen@dfki.uni-sb.de
tel.: +49-681-302-52 98

Peter **Baumgartner**
Universität Koblenz - Landau
Fachbereich Informatik
Rheinau 3 - 4
W-5400 Koblenz
Germany
peter@infko.uni-koblenz.de
tel.: +49-261-9119-4 26

Alexander **Bockmayr**
Max-Planck-Institut für Informatik
Im Stadtwald
W-6600 Saarbrücken
Germany
bockmayr@mpi.sb.mpg.de
tel.: +49-681-302-53 65

Alexandre **Boudet**
Université Paris Sud
Laboratoire de Recherche en Informatique
CNRS URA 490
Centre d'Orsay
F-91405 Orsay
Cedex
+33-1-69.41.64.76
tel.: France

Christoph **Brzoska**
Universität Karlsruhe
SFB 314
Postfach 6980
W-7500 Karlsruhe
Germany
brzoska@ira.uka.de
tel.: +49-721-608-35 64

Hubert **Comon**
Université Paris Sud
Laboratoire de Recherche en Informatique
Bât 490 Centre d'Orsay
F-91405 Orsay Cedex
France
comon@lri.lri.fr

Evelyne **Contejean**
Université Paris Sud
Laboratoire de Recherche en Informatique
Bât 490 CNRS URA 410
F-91405 Orsay Cedex
France
contejea@lri.lri.fr
tel.: +33-1-69.41.64.76

Bertrand **Delsart**
LIFIA-IMAG
46 avenue Felix Viallet
F-38031 Grenoble Cedex
France
bertrand@lifia.imag.fr
tel.: +33.76.57.48.05

Eric **Domenjoud**
INRIA Lorraine
Campus Scientifique
615 rue du Jardin Botanique
F-54600 Villers-lès-Nancy
France
eric.domenjoud@loria.fr
tel.: +33.83.59.30.19

Maribel **Fernandez**
Université Paris Sud
Laboratoire de Recherche en Informatique
Bât 490 Centre d'Orsay
F-91405 Orsay Cedex
France
marabel@lri.lri.fr
tel.: +33-1-69.41.65.92

Bernhard **Gramlich**
Universität Kaiserslautern
FB Informatik
Postfach 3049
W-6750 Kaiserslautern
Germany
gramlich@informatik.uni-kl.de
tel.: +49-631-205-26 14

Georgios **Grivas**
ETH - Zürich
Institut für Theoretische Informatik
ETH-Zentrum
CH-8092 Zürich
Switzerland
grivas@etha.ch
tel.: +41-1-254-73 91

Steffen **Hölldobler**
Technische Hochschule Darmstadt
FB Informatik Intellektik
Alexanderstr. 10
W-6100 Darmstadt
Germany
steffen@intellektik.informatik.th-darmstadt.de
tel.: +49-6151-16-54 69

Erik **Hamoen**
Vrije Universiteit Amsterdam
Faculteit Wiskunde en Informatica
De Boelelaan 1081a
NL-1081 HV Amsterdam
The Netherlands
erik@cs.vk.nl
tel.: +31-20-5 48 55 79

Martin **Henz**
DFKI Saarbrücken
Stuhlsatzenhausweg 3
W-6600 Saarbrücken 11
Germany
henz@dfki.uni-sb.de
tel.: +49-681-302-53 10

Christian **Holzbaur**
University of Vienna
Dept. of Medical Cybernetics & AI
Freyung 6
A-1010 Wien
Austria
christian@ai.univie.ac.at
tel.: +43-222-53 53 32 81 0

Ullrich **Hustadt**
Max-Planck-Institut für Informatik
Geb. 44
Im Stadtwald 15
W-6600 Saarbrücken 11
Germany
ULLRICH.HUSTADT@mpi-sb.mpg.de
tel.: +49-681-302-54 31

Patricia **Johann**
Universität des Saarlandes
FB Informatik
Im Stadtwald 15
W-6600 Saarbrücken
Germany

Jean-Pierre **Jouannaud**
Université Paris-Sud
Laboratoire de Recherche en Informatique
Bât 490
F-91405 Orsay Cedex
France

Delia **Kesner**
Université Paris Sud
Laboratoire de Recherche en Informatique
Bât 490
F-91405 Orsay Cedex
France
kesner@lri.lri.fr
tel.: +33-1-69.41.65.92

Assaf **Kfoury**
Boston University
Computer Science Deptartment
III Cummington Street
Boston MA 02215
USA
kfoury@cs.bu.edu
tel.: +1-617-353-89 19

Claude **Kirchner**
CRIN & INRIA Lorraine
Campus Scientifique
615 rue du Jardin Botanique
F-54600 Villers-lès-Nancy
France
claude.kirchner@loria.fr
tel.: +33-83.59.30.11

Hélène **Kirchner**
CRIN & INRIA Lorraine
Campus Scientifique
615 rue du Jardin Botanique
F-54600 Villers-lès-Nancy
France
helene.kirchner@loria.fr
tel.: +33-83.59.30.12

Francis **Klay**
CRIN & INRIA Lorraine
Campus Scientifique
615 rue du Jardin Botanique
F-54600 Villers-lès-Nancy
France
francis.klayr@loria.fr
tel.: +33-83.59.30.19

Michael **Kohlhase**
FB Informatik
Im Stadtwald 15
W-6600
Saarbrücken 11
+49-681-302-46 27
tel.: kohlhase@cs.uni-sb.de

Stefan **Kurtz**
Universität Bielefeld
Technische Fakultat
Postfach 10 01 31
W-4800 Bielefeld
kurtz@techfak.uni-bielefeld.de
tel.: +49-521-1 06 29 06

Denis **Lugiez**
INRIA Lorraine
Campus Scientifique
615 rue du Jardin Botanique
F-54600 Villers-lès-Nancy
France
lugiez@loria.fr
tel.: +33.83.59.30.20

Paliath **Narendran**
SUNY at Albany
Dept. of Computer Science
1400 Washington Avenue
Albany NY 12222
USA
dran@cs.albany.edu
tel.: +1-518-442-33 87

Tobias **Nipkow**
TU München
Institut für Informatik
Arcisstraße 21
W-8000 München 2
Germany
tobias.nipkow@informatik.tu-
muenchen.de
tel.: +49-89-2105-26 90

Leszek **Pacholski**
Academy of Sciences
IMPAN
Kopernika 18
PL-51 617 Wroclaw
Poland
pacholsk@plwruw11.bitnet
tel.: +48-71-25 12 71

Zhenyu **Qian**
Universität Bremen
Fachbereich Mathematik/Informatik
Postfach 33 04 40
W-2800 Bremen 33
Germany
qian@informatik.uni-bremen.de
tel.: +49-421-218-22 39

Christophe **Ringeissen**
INRIA Lorraine
Campus Scientifique
615 rue du Jardin Botanique
F-54600 Villers-lès-Nancy
France
ringeiss@loria.fr
tel.: +33.83.59.30.15

Mikael **Rittri**
Chalmers University of Technology
Department of Computer Sciences
S-412 96 Göteborg
Sweden
rittri@cs.chalmers.se
tel.: +46-31-77 21 000

William **Rounds**
The University of Michigan
EECS Department
Ann Arbor MI 48109
USA
rounds@engin.umich.edu
tel.: +1-313-764-94 18

Manfred **Schmidt-Schauß**
Universität Frankfurt
Fachbereich Informatik (20)
Robert-Mayer-Str. 11-15
W-6000 Frankfurt 11
Germany
schauss@informatik.uni-frankfurt.de
tel.: +49-69-798-85 97

Klaus **Schulz**
Universität München
Centrum für Informations- und
Sprachverarbeitung (CIS)
Leopoldstr. 139
W-8000 München 40
Germany
schulz@cis.uni-muenchen.dbp.de
tel.: +49-89-36 40 72

Jörg **Siekmann**
DFKI Sbr. u. Uni. des Saarlandes
Stuhlsatzenhausweg 3
W-6600 Saarbrücken 11
Germany
siekmann@dfki.uni-sb.de
tel.: +49-681-302-52 75/76

Gert **Smolka**
DFKI Saarbrücken
Stuhlsatzenhausweg 3
W-6600 Saarbrücken 11
Germany
smollka@dfki.uni-sb.de
tel.: +49-681-302-53 11

Wayne **Snyder**
Boston University
Computer Science Deptartment
III Cummington Street
Boston MA 02215
USA
snyder@cs.bu.edu
tel.: +1-617-353-89 25

Rolf **Socher**
Max-Planck-Institut für Informatik
Im Stadtwald 15
W-6600 Saarbrücken 11
Germany
socher@mpi-sb.mpg.de
tel.: +49-681-302-53 67

Ralf **Treinen**
DFKI Saarbrücken
Stuhlsatzenhausweg 3
W-6600 Saarbrücken 11
Germany
treinen@dfki.uni-sb.de
tel.: +49-681-302-53 14

Marian **Vittek**
INRIA Lorraine
Campus Scientifique
615 rue du Jardin Botanique
F-54600 Villers-lès-Nancy
France
vittek@loria.fr

Jörg **Würtz**
DFKI Saarbrücken
Stuhlsatzenhausweg 3
W-6600 Saarbrücken 11
Germany
wuertz@dfki.uni-sb.de
tel.: +49-681-302-53 15

Toby **Walsh**
University of Edinburgh
Dept. of Artificial Intelligence
80 South Bridge
Edinburgh EHI 1HN
Great Britain
t.walsh@ed.ac.uk
tel.: +44-31-650-27 28

Franz **Weber**
FZI Karlsruhe
Haid-und-Neu-Straße 10-14
W-7500 Karlsruhe 1
Germany
fweber@fzi.de
tel.: +49-721-96 54-6 14

Andreas **Werner**
Universität Karlsruhe
SFB 314
Postfach 6980
W-7500 Karlsruhe
Germany
werner@ira.uka.de
tel.: +49-721-608-35 64

David **Wolfram**
Oxford University
Computing Laboratory
Programming Research Group
11 Keble Road
Oxford OX1 3QD
Great Britain
David.Wolfram@prg.39
tel.: +44-865-27 38 40

B. Courcelle, H. Ehrig, G. Rozenberg, H.J. Schneider (editors):
Graph-Transformations in Computer Science, Dagstuhl-Seminar-Report; 53; 04.01.-08.01.93 (9301)

A. Arnold, L. Priese, R. Vollmar (editors):
Automata Theory: Distributed Models, Dagstuhl-Seminar-Report; 54; 11.01.-15.01.93 (9302)

W.S. Cellary, K. Vidyasankar , G. Vossen (editors):
Versioning in Data Base Management Systems, Dagstuhl-Seminar-Report; 55; 01.02.-05.02.93 (9305)

B. Becker, R. Bryant, Ch. Meinel (editors):
Computer Aided Design and Test , Dagstuhl-Seminar-Report; 56; 15.02.-19.02.93 (9307)

M. Pinkal, R. Scha, L. Schubert (editors):
Semantic Formalisms in Natural Language Processing, Dagstuhl-Seminar-Report; 57; 23.02.-26.02.93 (9308)

H. Bibel, K. Furukawa, M. Stickel (editors):
Deduction , Dagstuhl-Seminar-Report; 58; 08.03.-12.03.93 (9310)

H. Alt, B. Chazelle, E. Welzl (editors):
Computational Geometry, Dagstuhl-Seminar-Report; 59; 22.03.-26.03.93 (9312)

J. Pustejovsky, H. Kamp (editors):
Universals in the Lexicon: At the Intersection of Lexical Semantic Theories, Dagstuhl-Seminar-Report; 60; 29.03.-02.04.93 (9313)

W. Straßer, F. Wahl (editors):
Graphics & Robotics, Dagstuhl-Seminar-Report; 61; 19.04.-22.04.93 (9316)

C. Beeri, A. Heuer, G. Saake, S.D. Urban (editors):
Formal Aspects of Object Base Dynamics , Dagstuhl-Seminar-Report; 62; 26.04.-30.04.93 (9317)

R. Book, E.P.D. Pednault, D. Wotschke (editors):
Descriptional Complexity: A Multidisciplinary Perspective , Dagstuhl-Seminar-Report; 63; 03.05.-07.05.93 (9318)

M. Wirsing, H.-D. Ehrich (editors):
Specification and Semantics, Dagstuhl-Seminar-Report; 64; 24.05.-28.05.93 (9321)

M. Droste, Y. Gurevich (editors):
Semantics of Programming Languages and Algebra, Dagstuhl-Seminar-Report; 65; 07.06.-11.06.93 (9323)

G. Farin, H. Hagen, H. Noltemeier (editors):
Geometric Modelling, Dagstuhl-Seminar-Report; 66; 28.06.-02.07.93 (9326)

Ph. Flajolet, R. Kemp, H. Prodinger (editors):
"Average-Case"-Analyse von Algorithmen, Dagstuhl-Seminar-Report; 67; 12.07.-16.07.93 (9328)

J.W. Gray, A.M. Pitts, K. Sieber (editors):
Interactions between Category Theory and Computer Science, Dagstuhl-Seminar-Report; 68; 19.07.-23.07.93 (9329)

V. Marek, A. Nerode, P.H. Schmitt (editors):
Non-Classical Logics in Computer Science, Dagstuhl-Seminar-Report; 69; 20.09.-24.09.93 (9338)

A. Odlyzko, C.P. Schnorr, A. Shamir (editors):
Cryptography, Dagstuhl-Seminar-Report; 70; 27.09.-01.10.93 (9339)