Wolfgang Bibel, Koichi Furukawa,
Mark Stickel (editors):

Deduction

Dagstuhl-Seminar-Report; 58
08.03.-12.03.93 (9310)

ISSN 0940-1121

Tel.: +49-6871 - 2458
Fax: +49-6871 - 5942

Das Internationale Begegnungs- und Forschungszentrum für Informatik (IBFI) ist eine gemeinnützige GmbH. Sie veranstaltet regelmäßig wissenschaftliche Seminare, welche nach Antrag der Tagungsleiter und Begutachtung durch das wissenschaftliche Direktorium mit persönlich eingeladenen Gästen durchgeführt werden.

Bezugsadresse: Geschäftsstelle Schloß Dagstuhl
Informatik, Bau 36
Universität des Saarlandes
Postfach 1150
66041 Saarbrücken
Germany
Tel.: +49 -681 - 302 - 4396
Fax: +49 -681 - 302 - 4397

Dagstuhl Seminar on Deduction

Wolfgang Bibel
Technische Hochschule Darmstadt

Koichi Furukawa
Keio University

Mark Stickel
SRI International

Logic is an essential formalism for computer science and artificial intelligence. It is used in such diverse and important activities as

- Problem specification.
- Program transformation, verification, and synthesis.
- Hardware design and verification.
- Logic programming.
- Deductive databases.
- Knowledge representation, reasoning, diagnosis, and planning.
- Natural language understanding.
- Mathematical theorem proving.

The universality of the language of logic, the certainty about the meaning of statements in logic, and the implementability of operations of logic, all contribute to its usefulness in these endeavors.

Implementations of logical operations are realized in the field of automated deduction, which has introduced fundamental techniques such as unification, resolution, and term rewriting, and developed automated deduction systems for propositional, first-order, higher-order, and nonclassical logics.

This meeting was convened to give international researchers on deduction the opportunity to meet and discuss techniques, applications, and research directions for deduction. Presentations covered many topics of current research in the field. At least equally valuable was the opportunity to discuss our successes, failures, plans, and dreams. We have achieved some great successes, such as solving open problems in mathematics and verifying a microprocessor design, and deductive techniques are embedded in logic programming, deductive database, and artificial intelligence systems. However, automated deduction systems are not yet used extensively by mathematicians, logicians, or hardware and software developers. At the start of a new German national project on deduction, our discussions of goals for the field, as well as its methods, could prove important.

The success of this meeting was due in no small part to the Dagstuhl Seminar Center and its staff for creating such a friendly and productive environment. The organizers and participants greatly appreciate their effort.

Contents

Hierarchical Theorem Proving Using Rewrite Techniques

Jürgen Avenhaus, Klaus Becker, Claus-Peter Wirth
Universität Kaiserslautern

We present a method to handle rewrite systems R with a built-in algebra $\mathcal{A}$. The algebra $\mathcal{A}$ may be a well-known one (such as the integer or rational numbers) or may itself be given by a convergent rewrite system R_0. The approach allows one to define new (partial) functions on $\mathcal{A}$ by positive/negative conditional rewrite rules.

To define the semantics of such a specification we use a mild form of order-sorted specifications to deal with undefined terms and give a careful definition of how to evaluate the negative conditions in a rewrite rule. As a result of this definition, adding new equations to the specification is a monotone operation in the sense of logic. We carry over the well-known methods to prove confluence of the rewrite relation $\rightarrow_R$, provided $\rightarrow_R$ is terminating. To prove that R is terminating we extend the RPO-approach to our setting. Such a proof may need to prove a theorem in $\mathcal{A}$.

The benefits of this work are

- Hierarchical theorem proving is supported
- Explicit knowledge on the world of interest can be stated
- Efficiency of the built-in operators can be used
- Expressiveness of specification is enhenced
- Partial functions are allowed

Distributed knowledge-based equational theorem proving

Jörg Denzinger
Universität Kaiserslautern

Distributing the theorem proving task to several experts is a promising idea. Our approach, the team work method, allows the experts to compete for a while and then they are forced to cooperate. Each expert has a referee that judges the work done by the expert (competition) and that selects useful results of his expert (thus achieving cooperation). The referees report to a supervisor that uses all results of the best expert and the selected results of the other experts to generate a new starting input for a next round. This is repeated until a proof is found. Experts use different tactical control

knowledge, referees use assessment knowledge and the supervisor is based on strategical control knowledge.

We used the team work method to distribute equational theorem proving based on unfailing completion. Experiences showed that for many examples remarkable (i.e. "super-linear") speed-ups can be achieved.

Sort Unfolding Is Not Horn-Clause Resolution

Hassan Ait-Kaci , Andreas Podelski
Digital Paris Research Lab

We will describe, and formally justify, an algorithm performing lazy sort unfolding in the normalization of order-sorted feature (OSF) structures modulo a sort theory. We will argue that this algorithm has a radically different operational effect and logical semantics than the resolution-based method used in most other systems with sort definitions. All other formalisms and systems known to us, most in linguistics, that support sort definitions, see sorts as monadic predicates defined as Horn-clauses and then operationally enforce sort constraints by Horn-clause resolution. The essential but very important difference is that our algorithm does not generate the solutions of a sort theory's axioms, but rather is a maximally passive structural constraint enforcing scheme that weeds out any violation of a sort's defining axiom by objects claimed or derived to be of that sort.

We claim that the techniques we have developed for this algorithm go beyong their mere use in feature structure formalism. They provide an elegant and efficient means to incorporate knowldege-based reasoning in a deduction process as well as offer a powerful and clean facility for object-oriented logic programming.

Building models while searching refutations

Ricardo Caferra
LIFIA-IMAG FRANCE

We report results of a previous work (together with N. Zabel) and a current research (together with C. Bourely). It concerns a new (resolution-based) method for simultaneous search of refutations and models.

The main idea of the approach is to set conditions on premises allowing both to apply inferences rules and to avoid their application. It is proved refutational complete and to be a decision procedure for some classes. It has been extended (keeping the same basic idea) in order to incorporate equality. It has been applied to some non-trivial examples, in particular to a problem solved by Wos and Winker in 1982 in a (too)

assited way. Our solution is (almost) completely automatized. In Wos and Winker approach all important decisions are taken by the user. In our approach the very few decisions taken by the user are strongly suggested by the method.

Resolution Decision Procedures and Automated Model Building

Alexander Leitsch
University of Technology, Vienna AUSTRIA

A proof theoretical method, based on resolution refinements, is used to decide a wide range of clause classes. The basic idea consists in defining specific resolution refinements which terminate on (decidable) clause classes. By this approach one can successfully handle almost all of the classical decidable prefix classes; moreover a wide range of non-prefix classes (the function symbols need not be of Skolem type) can be decided by resolution. Based on the approach of W.H.Joyner we give a method-oriented approach to the decision problem in clause logic. First A-ordering refinements are presented and it is outlined, how the one-variable class can be decided by this type of refinement; it is emphasized that the use of a-posteriori ordering and condensing is necessary in order to get termination. It is shown that A-ordering refinements are not suited for deciding classes having syntax properties strongly connected to the propositional structure of clauses; e.g. no A-ordering refinement terminates on the class Bernays-Schönfinkel-Horn. For classes of this type hyperresolution turns out to be an adequate refinement. We define the classes PVD and OCC1N, which are general (non-Horn) termination classes for hyperresolution. In order to decide these classes a refinement generator (for semantical settings) instead of a fixed refinement is required. Resolution decision procedures are at the same time very efficient theorem provers. As decision refinements keep term complexity and clause size low, it is suggested to design a theorem prover based on a preclassification of sets of clauses by means of decision theory. The termination classes of hyperresolution can be used as raw material for a algorithmic model building method which does not use case-splitting. In case of PVD and OCC1N the Herbrand models can be turned into finite models by filtration.

Uses of Many–valued Logic in Hardware Verification

Reiner Hähnle
University of Karlsruhe

Recent advances in deduction techniques for many–valued logic (MVL) created the possibility of designing theorem provers for MVL that can compete in performance with their classical counterparts. This holds for propositional as well as for first–order logic.

On the other hand, a review of the literature where MVL theorem proving can play a rôle, shows that instead of genuine MVL reasoning reductions to classical logic are used. The implementations are usually based on decision diagrams.

We argue that genuine MVL proof procedures are potentially more efficient (and certainly more general) and we point out some problems of hardware verification, where techniques developed in many–valued deduction might be applied advantageously.

What Theorem Proving can Learn from Model Theory

Bernd I. Dahn
Berlin

Model theory is concerned with the study of the relations between properties of theories and properties of it's model classes. Looking at the provability of a formula as a property of the class of models of the given theory, properties of that class can be used to reason about the provability of the formula.

This requires a logic, that has a good model theory (like first order logic, topological logic L_{top} or the logic with the quantifier "there exist (uncountably) many" L_{Q_1}) and a detailed knowledge of the algebraic properties of the concrete class of models.

Besides negative results showing that each proper extension of first order logic loses some valuable model theoretic property, there are positive model theoretic results that can be useful in the design of theorem provers for specific theories. E. g. due to a theorem by Ryll-Nardzewski the branching of cases for countably categorical theories can be bounded. Axiomatizability and quantifier elimination results can restrict the complexity of the formulas to be considered.

The invariance of the validity of formulas under certain algebraic constructions (e. g. subdirect products, homomorphic images etc.) can be applied to transform the problem given to a prover. This is demonstrated by the design of a non-heuristic control mechanism for the work of background experts in a prover for the theory of lattice ordered groups within the ILF system, where the selection of the expert for a

given subproblem as well as the evaluation of it's results is guided by model theoretic and algebraic principles.

The Productive Use of Failure in Inductive Theorem Proving

Alan Bundy
University of Edinburgh

Lemma discovery and generalisation are two of the major hurdles in automating inductive proof. We show how to use the failure of an initial proof attempt to suggest appropriate lemmata and generalisations. We build upon *rippling*, a heuristic which plays a pivotal role in guiding inductive proof search. The goal of rippling is to reduce the difference between the induction conclusion and the induction hypothesis by the selective application of appropriate rewrite rules. This can fail in various ways. Different failure patterns suggest different kinds of patches, eg the lack of a suitable rewrite rule suggests the proof of a lemma in the form of the missing rule; the inapplicability of a lemma suggests either a nested induction or a generalisation, according to the reason for the inapplicability. We use metavariables to stand for the unknown parts of the lemma/generalisation and instantiate these metavariables during the subsequent proof using higher-order unification.

Computing Induction Axioms

Christoph Walther
Technische Hochschule Darmstadt

We analyse techniques and heuristics for computing induction axioms as proposed in Boyer and Moore's "A Computational Logic" (1979). The aim of this research is to obtain a better understanding of the semantics of the proposals, i.e. what are the intended effects, which limits exist, are the limits relevant, can relevant limits be overcome etc.

It can be shown that one central goal of the proposals is to compute well-founded relations which (considered as sets) are supersets of given well-founded relations or to compare well-founded relations by set-theoretic inclusion. Another central goal is to guess useful instantiations for universally quantified variables which cannot be used in the Boyer&Moore logic.

The analysis uncovers faults of the subsumption heuristic which easily can be repaired yielding a non-heuristic comparison test for compeeting induction schemes.

It can be shown that merging is a heuristic for guessing 'successful' instantiations of universally quantified variables in induction hypotheses prior to a proof. This means

that the merging heuristic is obsolete if universally quantified variables are allowed in induction hypotheses.

Recent Theorem-Proving Activities at Argonne

Ewing Lusk
Argonne National Laboratory

The theorem-proving research group at Argonne (Bill McCune, Larry Wos, and Ewing Lusk) continues work on high-performance and parallel implementations of clause-based inference systems and the application of these systems to open problems in mathematics and logic. This talk surveyed recent activities in these areas. Bill McCune has recently used the Otter theorem prover to answer a series of previously open questions in the area of single axioms for the theories of groups, abelian groups, and ternary Boolean algebras. In every case Otter discovered new single axioms shorter than those previously known. He has also implemented a new theorem prover based on AC unification, and obtained proofs of classical problems that are beyond the reach of Otter, including the first known first-order proof that a Robbins algebra in which a+b=a for some a and b is a Boolean algebra. Larry Wos is working on finding shorter proofs for implicational calculus formulas from known proofs found by Otter or by others. For example, using Lukaciewicz's proof of Church's axiom from the Lukaciewicz axioms, which contains 33 steps, he was able to induce Otter to construct a proof using only 22 steps. The talk also briefly described Ewing Lusk's tools for writing and studying parallel programs, and their use in the development of various parallel theorem provers at Argonne.

Tractable Inference and Obviousness

David McAllester
Massachussetts Institute of Technology

We take the view that a proof verification system should provide a notion of proof such that a proof is a sequence of "acceptable" steps. A step is acceptable if the statement made in that step is "obvious" in a technical sense provided by the system. The main challenge in the construction of such a system is providing a notion of obviousness (or acceptibility for proof steps) that is natural for human proof writers and yet allows the system to efficiently determine if a given step is obvious. This talk discusses the use of inference rules in defining notions of obviousness. The talk emphasizes sets of inference rules that define polynomial time decidable inference relations. Powerful general purpose rule sets can be constructed using nonstandard syntax for first order logic. A syntax for first order logic based on Montague grammar is discussed along

with a corresponding polynomial time inference relation defined by inference rules in this syntax. The general nature of inference rules as a model of computation is also discussed. There are two main theorems in this area. First, the forward chaining closure of any set of premises under any set of inference rules can be computed in time proportional to the number of rules firings. This theorem allows for the simple construction of a variety of polynomial time algorithms. Second, inference rules provide a very simple descriptive characterization of the complexity class P. A set of expressions is a polynomial time decidable language if and only if it is the set of theorems of some polynomial time rule set.

A Typed λ-Calculus for Proving-by-Example and Bottom-Up Generalization Procedure

Masami Hagiya
University of Tokyo

We extend Logical Framework, one of the typed λ-calculi of Lambda Cube, and apply it to the problem of proving-by-example. The extended calculus allows recursions and inductions on natural numbers, and inferences on linear arithmetical terms are built into its type system. The extension corresponds to that of logic programming by constraints in the sense that the constraint solver cooperates with the ordinary type-checking algorithm. We then formulate a bottom-up procedure for generalizing a concrete proof by recovering inductions that are expanded in the concrete proof. The procedure can reconstruct inductions whose induction formula is a Σ_1-formula. The generalization procedure can be iterated to construct nested inductions. Consequently, it can find inductions whose induction formula is a limited form of bounded quantification.

Arithmetic for the partial continuous functions

Helmut Schwichtenberg
Mathematisches Institut der Universität München

Computable functionals have the Kreisel–Scott–Ersov partial continuous functionals as their natural domain. Hence in order to reason about functional programs it is natural to work in a higher order arithmetic with (program) constants for computable functionals and the sets D_ρ of partial continuous functionals of type ρ as the intended domains of the type ρ variables. To be able to talk about non–continuous functionals as well (like non–strict equality) we also allow function symbols denoting functions external to the model. A formal system of the strength of Peano arithmetic and based on the $\to \wedge\forall$ fragment of minimal logic is described which is then used to study the

proof as/about programs paradigms in this setting. The explicit representation of proofs as (essentially type-free) λ-terms makes it possible to do program development by proof transformation, specifically using Goad's idea of pruning proof trees. The system has been implemented (in SCHEME), using SCHEME evaluation as an efficient mechanism to normalize terms and proofs; a side result here is a stong completeness theorem for typed λ-calculus (cf. Berger/Schwichtenberg, LICS'91).

Extensions of Model Elimination Calculi

Reinhold Letz
TU Munich

In this talk extensions of model elimination calculi are presented, both concerning the reduction of proof length and the reduction of search space. The notions are introduced in the framework of connection tableaux, which generalizes model elimination and connection calculi in two respects; first, arbitrary subgoal selection functions are admissible in connection tableaux, secondly, the presence of tableaux as static proof objects throughout the deduction process facilitates the application of a larger number of refinements. In order to improve on the indeterministic power of connection tableaux, we introduce the so-called folding up and folding down operations which both can be viewed as controlled variants of the backward cut rule. The folding up operation is motivated by the fact that bottom-up lemmata can be extracted from the solution of a subtableau which can be reused later on at other parts of the tableau. Folding up achieves a particularly efficient realization of this idea by storing context unit lemmata in the tableau itself. As an optimistic version of folding up, the folding down operation is defined, and it is shown that folding down represents an efficient implementation of the factorization rule in tableaux. The way both operations are introduced also gives rise to a sharpening of the regularity condition, which forbids the presence of two identical literals on the same branch. Consequently, the resulting refinements of the new calculi, which are definitely superior to regular connection tableaux with respect to proof lengths, may also be better off concerning search pruning.

Connection calculi with built-in theories

Uwe Petermann
Universität Leipzig

We present a technique for building-in theories into first-order theorem provers and for proving their completeness. As the base calculus we adapt the pool calculus which is one of the formalizations of the connection method. The approach relies on a formal characterization of the interface between forground and background reasoner. This

interface has two components. The first component is a decidable set of theory connections. Theory connections generalize complementary pairs. Whenever the base calculus would have to detect a complementary pair the forground reasoner now has to detect a theory connection. The second component is an algorithm which computes a complete set of theory unifiers for each theory connection. This kind of theory unification is just the task of the background reasoner. It may be proved that the base calculus which has been modified this way is complete if there are enough theory connections. "Enough" is formalized by the notion of a complete set of theory connections. The approach may be applied among others to constraint theories, to taxonomic theories and to equational theories as their appear in the translation of modal into first-order logic. The examples show that complete sets of theory connections with solvable unification problem may exist only for subclasses of formulas. In some theories there is no upper bound of the number literals within a theory connection. This causes a high branching factor of the calculus. The way out is partial theory reasoning. That means that differently to total theory reasoning a theory connection and its unifier is not found in one step. Rather they are approximated by a number of transformations of literals. The partial theory connection calculus is complete if every theory connection may be approximated by a linear partial theory resolution derivation. For both the total and the partial theory connection calculus are given PTTP-like versions. Implementation is ongoing.

KL-ONE, Modal Logic, Generalized Quantifiers and First-Order Predicate Logic Theorem Proving

Hans-Jürgen Ohlbach
MPI Saarbrücken

There are very close correspondences between certain parts of the KL-ONE knowledge representation language and standard propositional modal logic. More sophisticated versions of KL-ONE correspond to multimodal logic of graded modalities. In this logic there are operators "for more than n ...", "for less than n ..." and "for exactly n ..." which also play a role in generalized quantifier theory.

In the talk I give an overview on these correspondances and I present a new method for mapping these systems to first order logic which avoids many of the problems in earlier approaches.

Some Thoughts on Automated Proof Discovery, Especially HOL and Analysis

Woody Bledsoe
University of Texas at Austin

We discussed our SET-VAR prover, for proving a portion of Second-Order Logic. Namely, those theorems of the form SOME A P(A), where P(A) contains the variable A ONLY in the form (t e A) (ie, t is an element of A), or ¬(t e A), and where A must be instantiated by a set of the form z: Q(z), where Q(z) is a formula in FOL. (We could have written (t e A) instead of A(t), where A is a monatic predicate variable.)

We gave examples of proofs using SET-VAR, discussed it soundness, and conjectured it completeness, for this extention of FOL. And showed its proof of a number of theorems, including the Intermediate Value Theorem, using the Least Upper Bound axiom.

We also compared its performance with other such systems. And stated that all other such known systems, though fundamentally important, need improvements before they can prove certain important theorems (such as the Intermediate Value Theorem, the Heine-Borel Theorem, etc.).

We also stated that to proceed further with such important theorems we must employ higher-level planning stategies, use goals, motives, etc.

An Inductively Complete Inference System

Taisuke Sato
Electrotechnical Laboratory Tsukuba

We present a set R of rules for inductive inference. There are three rules :

- The case rule infers from a clause C two clauses A v C, not(A) v C where A is an arbitrary atom.
- The inverse-or rule is the inverse of or introduction. Given a clause L v C, it infers C by deleting L.
- The anti-substitution rule infers a clause C from its instance $C\sigma$ where σ is a substitution.

R is inductively complete in the sense that given a finite set a of clauses, it can infer every possible clause set S such that $S \vdash a$.

Applications include machine learning, Programming from examples, and onventing laws for observed facts expressed in terms of clauses.

SATCHMORE: SATCHMO with RElevancy

Donald W. Loveland
Duke University, U.S.A.

As effective as forward chaining can be, it suffers from the inability to work only with relevant information, i.e., information that relates to the goal. We have developed a method for propagating goal information (relevant clauses and variable bindings) to (usually) non-Horn clauses being processed, and we present this method as a relevancy detection algorithm used with the SATCHMO (forward-chaining) prover. We use the version of SATCHMO that utilizes Prolog for backchaining on Horn clauses, with forward chaining on non-Horn clauses. We mark potentially relevant non-Horn clause head literals, and then require that all! head literals be marked relevant before a clause is used for forward chaining. This relevancy testing can be implemented quite simply in Prolog such that the search is not extended beyond that done for SATCHMO. Two examples are considered for which SATCHMO takes days to find a solution where SATCHMORE takes seconds. One example involves inclusion of an irrelevant non-Horn clause and the second involves propagation of variable bindings from the goal.

Non-Horn Magic Sets on MGTP

Ryuzo Hasegawa, Yoshihiko Ohta, Katsumi Inoue
ICOT Institute for the New Generation Computer Technology

We present a new method that combines top-down and bottom-up computations on model generation theorem provers.

This method called non-Horn magic sets is a natural extension of Horn magic sets and is applicable to range-restricted non-Horn clauses. We show two types of non-Horn magic sets transformations: breadth-first NHM and depth-first NHM. The first one evaluates the antecedent literals in a clause in parallel. The other evaluates them sequentially while propagating the bindings in an antecedent literal to the next by using continuation predicates. We prove the soundness and completeness of the two transformations. We also prove a theorem saying that non-Horn magic sets have the same power as SATCHMORE (SATCHMO with relevancy testing) which is proposed by Loveland et al. Several experiments have been made by running MGTP on the parallel inference machine PIM/m. The results show that non-Horn magic sets can avoid useless case-splitting thereby shortening execution time drastically. For instance, the benchmarks requiring over one hour execution time are solved within a second.

An Update on Mechanical Verification: Machine Code and Microprocessors

Robert S. Boyer
University of Texas at Austin and Computational Logic, Inc. (Clinc)

The Boyer-Moore mechanical theorem-prover (a.k.a. Nqthm, see the book *A Computational Logic Handbook*, Boyer and Moore, Academic Press, 1988) has been increasingly employed in the verification of computing systems. Two interesting examples of verification are discussed. One is the verification of dozens of small machine code programs for the Motorola 68020, where the machine code was obtained via "industrial strength" compilers for C and Ada. Included are such standard examples as quick sort and binary search, and all but one subroutine in the Berkeley Unix C string library. (Work of Yuan Yu.) The other is the verification of the "Clinc Stack", a computing system consisting of a microprocessor (the FM9001), an assembler, several compilers and several applications, with the entire edifice being mechanically verified from the level of gates and registers upwards through higher and higher layers of abstraction. The FM9001 has been successfully fabricated as an LSI Logic gate array, and runs the applications exactly as predicted by the verifications. (Work of W. Hunt, B. Brock, J Moore, M. Kaufmann, A. Flatau, W. Young, and M. Wilding.)

Basic Paramodulation

Harald Ganzinger
MPI Saarbrücken
(joint work with L. Bachmair, Chr. Lynch, and W. Snyder)

We introduce a class of restrictions for the ordered paramodulation and superposition calculi (inspired by the *basic* strategy for narrowing), in which paramodulation inferences are forbidden at terms introduced by substitutions from previous inference steps. In addition we introduce restrictions based on term selection rules and redex orderings, which are general critieria for delimiting the terms which are available for inferences. These refinements are compatible with standard ordering restrictions and are complete without paramodulation into variables or using functional reflexivity axioms. We prove refutational completeness in the context of deletion rules, such as simplification by rewriting (demodulation) and subsumption, and of techniques for eliminating redundant inferences.

Constraints and Unification

Deepak Kapur, Paliath Narendran
State University of New York at Albany

Unification problems over equational theories can be transformed to solving a system of constraints over various domains. Solutions to unification problems can be considered as as building a decision tree by using constraints based on subterms appearing in them. This framework not only provides an elegant way for thinking about unification, but also gives efficient, optimal algorithms for unifiability check as well as for generating a complete set of minimal unifiers. Complexity analysis can be performed easily; heuristics can be devised to utilize additional structural information in terms appearing in a unification problem. This is illustrated using unification problems with associative-commutative function symbols (which may or may not have idempotency property and identity). This framework also seems to be useful for type-inference problems in the presence of subtype relation, union and refinement types.

DEDUCTION WITH CONSTRAINTS

Claude Kirchner
INRIA Lorraine & CRIN

Deduction with constraints is a paradigm already widely recognized in the context of logic programming. A general framework for deduction with constraints is presented with emphasis on its advantages and the new problems that should be solved in order to extend complete deduction procedures like completion, resolution or narrowing. Recent results contributing to solving these problems will be presented.

This talk is based on our early work on deduction with constraints [1] and after showing the main concepts, we present ongoing researches on this topic. Constrained formulae are first-order formulae completed by constraints; their semantics is to schematize all the instances of the formula satisfying the constraint part. This technique has many advantages which already have been recognized in the context of logic programming. In the framework of deduction processes too, it is most useful to homogenize the deduction rules by letting apparent for example the unification, typing and orientation problems. First, this allows studying strategies at low level, refining them and hopefully deriving efficient ones, for instance by delaying the solving of difficult constraints. Second, constraints can account for many aspects of structure-sharing: several instances of a term may be replaced by one constraint. Last, and this is perhaps the main point, constrained deduction is more expressive than classical deduction in the following sense: it makes possible to represent infinitely many objects by a single one. For instance, the constrained equality $[f(x) = f(y), (x \neq y)]$ schematizes an infinite number of classical equalities which cannot be equivalently replaced by a finite number of equalities.

We shall see that this approach has useful applications in automated theorem proving in particular when focussing on unification, disunification and orientation problems. The proposed deduction rules are very general and we feel that most approaches followed in automated deduction with constraints fit into this framework. But the inference rules presented here should not be loosely implemented. In order to get well-behaved strategies a careful scheduling should be thought about too. In particular we will motivate the necessity of restructuration rules, the role of basicity and the main differences between the standard rewriting and constraint rewriting.

[1] C. Kirchner, H. Kirchner, and M. Rusinowitch :
Deduction with symbolic constraints.
Revue Francaise d'intelligence artificielle, 4(3):9–52, 1990. Special issue on Automatic Deduction.

Ω-MKRP: A Proof Development Environment

Jörg Siekmann, Xiaorong Huang, Manfred Kerber, Michael Kohlhase,
Erica Melis, Dan Nesmith, Jörn Richts
Universität des Saarlandes

Motivation: In the following we describe the basic ideas underlying Ω-MKRP, an interactive proof development environment. The requirements for this system were derived from our experiences in proving an interrelated collection of theorems of a typical textbook on semi-groups and automata with the first-order theorem prover MKRP. An important finding was that although current automated theorem provers have evidently reached the power to solve non-trivial problems, they do not provide sufficient assistance for proving such a textbook.

Requirements and our Approach: Our basic idea is to build a proof development environment which combines both the reasoning power of automated theorem provers as logic engines *and* of the proof planning mechanism proposed by Alan Bundy et al. The proof planning approach, which is largely human-oriented and lends itself to a more natural interactive interface, is complementary to the more machine-oriented traditional theorem proving approach. Below, we discuss two key features such systems should have, followed by the concrete solution we choose.

Appropriate Communication Language: The input languages of classical automated theorem provers are usually variants of first-order logic. When encoding the theorems of the mentioned textbook in first-order logic, we were forced to use sophisticated formulation techniques. That is, although such a language is sufficient *in principle*, *in practice* it is too weak to allow an adequate representation. Since the technical mathematical language in textbooks is much closer to sorted higher-order logic, we have developed such an input language called $\mathcal{POST}$. Since the user expects that apart from the knowledge of predicate logic, the system should also possess a large

amount of mathematical definitions and theorems, we are filling a database with the mathematical knowledge contained in the textbook mentioned above.

In addition to the problem formulation language, the proof format is crucial for an adequate interface. As a common proof format for both the user and the system, we have chosen the generally established natural deduction formalism.

Integration of Automated and Interactive Theorem Proving: There is a gap between the problem formulation language $\mathcal{POST}$ and the proof format, on the one hand, and the input and output languages of the underlying first-order theorem provers on the other hand. In order to bridge this gap we have to transform $\mathcal{POST}$ into the input languages of the underlying provers and to transform the output proofs (such as a clause proof of a resolution-based prover) into natural deduction proofs.

An advanced tool for proving mathematical theorems should also enable the user to communicate his proof strategies to the system, since every automated theorem prover still needs guidance through the search space for more difficult problems and this situation will not change fundamentally in the foreseeable future. In order to provide such facilities, we basically follow the proof planning approach of Alan Bundy, although certain modifications are necessary in order to incorporate higher-level human proof strategies such as analogy.

A problem solving cycle in our system can be described briefly as follows: The user formulates his problem in $\mathcal{POST}$. In order to solve the problem he can load definitions and already-proved theorems from the database, he can invoke a method to split the original problem by hand or let a planner do this, and he can pass a subproblem to a logic engine. When the subproblem is within the capability of the logic engine, a corresponding proof will be found and translated back and then the user will continue with the cycle from above.

Outlook: Currently we are implementing these ideas. As the first logic engine the MKRP system is already incorporated into the Ω-MKRP, other systems will follow, in particular logic engines for higher-order logic and mathematical induction.

Axiomatic Type Classes

Tobias Nipkow
TU München

Axiomatic type classes are a type system based approach to axiomatic theories. The aim is to support interpretations between theories in order to facilitate the following kind of reasoning: "In all linear orderings the *max*-function is commutative. The integers are linearly ordered. Hence *max* on integers is commutative". The basis is the concept of *type classes* in the functional programming language Haskell. Type classes are collections of types which provide certain operations. Adding axioms to type classes results in an axiomatic theory. Concrete theories (types) are shown to be instances of abstract theories (classes) via an interpretation of the abstract operations

in the concrete theory together with a proof of the abstract axioms. The concepts are demonstrated using various orderings and lattices.

The main advantages of the concept are

- simplicity: it only requires a simple extension of ML polymorphism;
- theory management is performed by type inference;
- overloading of functions is permitted.

Constraint Logic Programming with Negation

Gert Smolka
DFKI Saarbrücken

We present a framework GCLP for constraint logic programming with negation, which is designed to replace the conventional model based on SLDNF-resolution. GCLP assumes that a constraint system is given by a signature and a first-order theory, and that a program describes procedures by equivalences reminiscent of Clark's completion. As declarative semantics we take all three-valued models of the constraint theory and the program, where only procedures can have partial denotations (an approach due to Fitting and Kunen). A constraint is any first-order formula respecting the constraint signature.

GCLP is based on the well-known idea that the purpose of computation consists in making information explicit. This can be formalized by taking constraints as explicit information, and by calling a constraint ϕ an explication of a formula σ if $\phi \rightarrow \sigma$ is valid in every three-valued model of the constraint theory and the program. An interpreter is complete, if for every query and every explication it can compute an at least as tight explication when given the query and sufficient resources. We give an idealized complete interpreter and proceed to an abstract operational framework providing for the design and analysis of practical interpreters. The framework captures computation as don't care nondeterministic rewriting of expressions modulo a congruence accounting for constraint simplification and propagation. In contrast to the SLDNF-resolution, GCLP captures the notions of don't know choice and finite failure within the calculus.

An Algebraic Generalization of Information Subsumption

Kuniaki Mukai
Faculty of Environmental Information, Keio University, Japan

Extensional subsumption relation on feature algebras is generalized for coalgebras for class functors. For any set-based functor that preserves weak pullbacks, subsumption constraints have a solution in the final coalgebra for the functor if and only if they have a solution in some coalgebra for the functor. This result is a generalization of the unification lemma formalized and proved by J. Barwise, which, however, assumes the anti-foundation axiom and treats only the power class functor.

With some more reasonable assumptions, extendability of constraints to those consisting of bisimulations and subsumptions is decidable from a simple combinatorial reasoning due to our algebraic generalization of information subsumption. As feature structures are a coalgebra for a functor which satisfies the above condition, this result is a generalization of the decidability on external version of feature subsumption problems obtained by J. Dörre.

This present work is based on the final coalgebra theorem proved by P.Aczel and N.Mendler, which says that in ZFC minus the foundation axiom, every set-based functor has a final coalgebra.

Two Procedures for logic programs based on stable models

Ken Satoh
Fujitsu Labs., Kawasaki, Japan

In this talk, I give two procedures for logic programs. Stable model semantics is shown to be quite promising because of relationship with non-monotonic reasoning formalisms such as Autoepistemic Logic, Default Logic and Abduction. However, the original definition of stable model is given by fixed point and until recently we do not know how to compute stable models.

Fortunately, there are some proposals on bottom-up algorithm to compute stable models and we enhance it in treatment of integrity constraints. This is one of procedures in my talk and I give the simplest form of the bottom-up procedures.

In terms of top-down proof procedure or query evaluation method, there was a proposal by Kowalski and Eshghi, but it is only correct for a restricted class of logic programs. In this talk, we show a query evaluation method which is correct for all consistent logic programs. It combines Kowalski and Eshghi's abductive method and Kowalski and Sadri's integrity constraint check method. This method is applicable not only to a logic program but also to a logic program with integrity constraints.

Parallel Reasoning and Large inferences

J. A. Robinson
Syracuse University

Traditional ideas about deductive reasoning are strongly human-oriented: for example, the very common idea that proofs should be formalized as sequences of small inferences. But unification-based logics, such as those based on resolution, permit highly machine-oriented proofs whose constituents can be arbitrarily large inferences which need not be arranged or processed sequentially. Such proofs are networks of inferences which can be processed (in particular, checked for correctness) in parallel. Unification itself is a naturally parallel concept. Fast parallel algorithms (which are also extremely simple) are known for unification, and their proper deployment yields large-inference patterns (such as ultraresolution) featuring a very high degree of potential parallelism.

Proofs built from such 'unnatural' machine-oriented logics are unsuitable, and often even unintelligible, for humans. To make them directly useful to humans as aids to understanding, it would be necessary to develop techniques for rearranging and rescaling proofs, and for magnifying or zooming in on large inferences so that they can be seen as composite proofs built from smaller inferences.

On the Adequateness of the Connection Method

Steffen Hölldobler
TH Darmstadt

Roughly speaking, adequatness is the property of a theorem proving method to solve simpler problems faster than more difficult ones. Automated inferencing methods are often not adequate as they require thousands of steps to solve problems which humans seem to solve effortlessly, spontaneously, and with remarkable efficiency. Lokendra Shastri and Venkat Ajjanagadde — who call this gap the artificial intelligence paradox — claim that their connectionist inference system is a first step toward bridging this gap. In this talk we show that their inference method is just reasoning by reductions in the well-known connection method. In particular, we extend a reduction technique called evaluation of isolated connections such that this technique — together with other reduction techniques — solves all problems which can be solved by Shastri and Ajjanagadde's system under the same parallel time and space requirements. Consequently, we obtain a semantics for Shastri and Ajjanagadde's logic. But, most importantly, if Shastri and Ajjanagadde's logic really captures the kind of reasoning which humans can perform effortlessly, spontaneously, and with remarkable efficiency, then this talk shows that only a parallel implementation of the connection method may be adequate.

Search Space and Average Proof Length of Resolution

Hans Kleine Büning , Theodor Lettmann
Universität Paderborn

We introduce a definition of search space for resolution based proof procedures. This definition describes more clearly the differences between the restrictions of resolution. Applying this concept to monotone restrictions of the resolution, an example is the N–resolution, we show that the average proof length for propositional formulas is at most four times as large as for unrestricted resolution. The inequality shows that comparing proof procedures with respect to minimal proof length and to average proof length may lead to essentially different results, because for the N–resolution there exists infinite many formulas with superpolynomially minimal proofs whereas the resolution has polynomially minimal proof length.

Aspects of the Proof-System KOMET

W. Bibel
TH Darmstadt

Based on the experience with the development of the proof-system SETHEO, a new system, called KOMET (Connection MEthod Theorem prover), is under development. We aim at a computationally adequate system which roughly means that simple theorems are proved faster than harder ones. The talk begins with identifying six different structural features in (connection) proofs, namely simple extensions, hinged loops, cycles, data base entries, factors, and macro structures. For some of them new results are presented which have been integrated in the kernel version of KOMET.

The first result is the reachability graph which isolates all possible hinged loops in a preprocessing step and this way reduces the overhead needed for performing reduction steps. Data base entries for a given relation are treated as a single literal using index tree representation which speeds up the performance of database oriented problems enormously. Two results have a relevance for treating cycles. One allows the computation of a cycle in a grammatical (rather than a deductive) way thus eliminating search altogether. The second one allows the identification of unsuccessful cycles in advance and this way prevents the system to enter an infinite loop. Other features of KOMET such as reductions, non-normal form handling, lemmata-handling, integration of induction are also briefly mentioned.

Controlling Deduction through Declarative Constructs

Hans-Joachim Goltz
GMD-FIRST

The main point of this talk is to show that an important direction of the further development of logics used for deduction systems consists in the fact that declarative syntactic constructs can be used more powerfully for the representation of knowledge about the deduction system and for controlling deduction such that the search space can be reduced. Existing syntactic constructs can better be used for controlling deduction and new declarative syntactic constructs can be introduced for this purpose. These syntactic constructs can be used for different aims such as for building up terms, for the classification of the elements of the universe, for the classification of different kinds of functional symbols or of expressions (e.g. equations), and for the representation of meta knowledge.

Methods of controlling deduction by syntactic constructs are discussed for structuring the universe by a type system and for reducing the search space of unification.

Solvable Cases of the Semi-Unification Problem

Hans Leiß
CIS, Universität München

The semi-unification problem is the problem to find, for a given finite set of *inequations*

$$S = \{s_1 \leq_1 t_1, \ldots, s_n \leq_n t_n\}$$

between first-order terms s_i, t_i, a solution $T = (T_0, \ldots, T_n)$ such that $T_0(s_i)$ *matches* $T_0(t_i)$ for $i = 1, \ldots, n$, i.e.

$$T_i(T_0(s_i)) \equiv T_0(t_i) \qquad \text{for each } i = 1, \ldots, n.$$

Otherwise the non-existence of such a solution should be reported.

We present a simple proof of M.Baaz' result that solvable instances of the problem have most general solutions $(T_0, \ldots, T_n)$, in the sense that for other solutions $(R_0, \ldots, R_n)$, the main substitution R_0 can be factored through T_0 on the free vasriables of S. This is obtained from a sound and relatively complete transformation calculus for semi-unification problems, which is given in terms of equation systems with monadic function variables ranging over substitutions.

Due to a result of Kfoury e.a., the semi-unification problem is recursively unsolvable when at least two inequation relations $\leq_i$ (resp. two function variables for T_1, T_2) and a binary function constant are involved. The second part of the talk tries to isolate a

common structure in two solvable subproblems: (a) Kapur e.a.'s result that solvability is decidable when only one function variable T_1 occurs, and (b) the authors's result that solvability is decidable when each term of the instance S has only one maximal subterm constructed of individual and function variables.

Theorem Proving by Model Testing

David Plaisted, Heng Chu
University of North Carolina at Chapel Hill

We discuss our semantic hyper-linking theorem prover which shows unsatisfiability of a first-order formula by the failure of a persistent attempt to construct a model for it. The basic procedure is complete and especially efficient for sets of clauses that are highly non-Horn and have small literals. We augment this procedure with UR (unit-resulting) resolution to eliminate the Horn parts of proofs and with a refinement of resolution called rough resolution, to eliminate the large literals in proofs. This combination of three methods, two of them incomplete, works well and proves a number of theorems automatically that are beyond the reach of most theorem provers. These include the intermediate value theorem, am8, exq1, exq2, exq3, and three of Bledsoe's limit theorems. All proofs are obtained in reasonable times with small search spaces and default settings of all switches.

Dagstuhl-Seminar 9310: List of Participants (11.03.93)

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

Wolfgang **Bibel**
TH Darmstadt
FG Intellektik
Alexanderstr. 10
W-6100 Darmstadt
Germany
bibel@intellektik.informatik.th-darmstadt.de
tel.: +49-6151-16-21 00

Woody **Bledsoe**
University of Texas at Austin
Department of Computer Sciences
Taylor Hall
Austin TX 78712-1188
USA
bledsoe@cs.utexas.edu
tel.: +1-512-453-1101

Robert **Boyer**
University of Texas at Austin
Department of Computer Sciences
Taylor Hall 2.124
Austin TX 78712-1188
USA
boyer@cs.utexas.edu / boyer@cli.com
tel.: +1-512-471-97 45

Alan **Bundy**
University of Edinburgh
Dept. of Artificial Intelligence
80 South Bridge
Edinburgh EH1 IHN
Great Britain
bundy@ed.ac.uk
tel.: +44-31-650-27 16

Jochen **Burghardt**
Universität Karlsruhe
GMD-Forschungsstelle
Vincenz-Prießnitz-Str. 1
W-7500 Karlsruhe
Germany
burghard@karlsruhe.gmd.de
tel.: +49-721-6622-45

Ricardo **Caferra**
LIFIA-IMAG
46 avenue Felix Viallet
F-38031 Grenoble Cedex
France
caferra@lifia.imag.fr
tel.: +33 7657 4659

Bernd **Dahn**
Tetrower Ring 37
O-1044 Berlin
Germany
dahn@hubinf.informatik.hu-berlin.de
tel.: +49-30-561-0385

Jörg **Denzinger**
Universität Kaiserslautern
FB Informatik
Postfach 3049
W-6750 Kaiserslautern
Germany
denzinge@informatik.uni-kl.de
tel.: +49-631-205-2181

Elmar **Eder**
Universität Salzburg
Institut für Computerwissenschaften
Jakob Haringer Str. 5
A-5020 Salzburg
Austria
eder@cosy.sbg.ac.at
tel.: +43-662-8044-6314

Koichi **Furukawa**
Kaio University
Faculty of Environmental
Information
5322 Endo
Kanagawa
252
+81-466-47-5111
tel.: Japan

Harald **Ganzinger**
Max-Planck-Institut für Informatik
Im Stadtwald 15
W-6600 Saarbrücken 11
Germany
hg@mpi-sb.mpg.de
tel.: +49-681-302-5361 (Secr. 5360)

Hans-Joachim **Goltz**
GMD-FIRST
Rudower Chaussee 5
O-1199 Berlin
Germany
goltz@first.gmd.de
tel.: +49-030-67045298

Shigeki **Goto**
NTT Software Research Labs.
NTT Twins Buildings
1-9-1 Kohnan
Minato-ku
Tokyo 108
Japan
goto@ntt-20.ntt.jp
tel.: +81-3-3740-6050

Reiner **Hähnle**
Universität Karlsruhe
Fakultät für Informatik
Am Fasanengarten 5
W-7500 Karlsruhe
Germany
reiner@ira.uka.de
tel.: +49-721-608-4329

Masami **Hagiya**
University of Tokyo
Department of Information Science
7-3-1 Hongo
Bunkyo-ku
Tokyo 113
Japan
hagiya@is.s.u-tokyo.ac.jp
tel.: +81-3-3812-2111

Ryuzo **Hasegawa**
ICOT Institute for the New Generation
Computer Technology
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108
Japan
hasegawa@icot.or.jp
tel.: +81-3-3456-4365

Bernd **Heimrich**
Dt. Forschungsgemeinschaft
Referat II D6
Postfach 20 50 04
W-5300 Bonn 2
Germany
tel.: +49-228-885-25 23

Steffen **Hölldobler**
Technische Hochschule Darmstadt
Fachbereich Informatik
Alexanderstr. 10
W-6100 Darmstadt
Germany
steffen@intellektik.informatik.th-darmstadt.de
tel.: +49-6151-16-5469

Deepak **Kapur**
SUNY Albany
Dept. of Computer Science
L167A
Albany NY 12222
USA
kapur@cs.albany.edu
tel.: +1-0110518-442-4281

Claude **Kirchner**
INRIA & CRIN Lorraine
615 rue du Jardin Botanique
F-54602 Villers les Nancy Cedex
France
ckirchne@loria.loria.fr
tel.: +33-83593011

Hans **Kleine Büning**
Universität GH Paderborn
FB 17 - Mathematik/Informatik
Warburgerstr. 100
W-4790 Paderborn
Germany
kbcsl@uni-paderborn.de
tel.: +49-5251-60 33 61

Hans **Leiß**
Universität München
Centrum für Informations- und
Sprachverarbeitung (CIS)
Leopoldstr. 139
W-8000 München 40
Germany
leiss@cis.uni-muenchen.de
tel.: +49-89-36 40 72 Ext.21

Alexander **Leitsch**
TU Wien
Institut für Computersprachen
Resselgasse 3
A-1040 Wien
Austria
leitsch@logic.tuwien.ac.at
tel.: +43-1-58801-4095

Reinhold **Letz**
TU München
Forschungsgruppe KI
Augustenstr. 46
W-8000 München 2
Germany
letz@informatik.tu-muenchen.de
tel.: +49-89-521098

Donald W. **Loveland**
Duke University
Computer Science Dept.
210 North Building
Box 90129
Durham NC 27708-0129
USA
dwl@cs.duke.edu
tel.: +1-919-660-6542

Ewing **Lusk**
Argonne National Laboratory
Math. and Computer Science Div.
9700 South Cass Avenue
Argonne IL 60439
USA
lusk@mcs.anl.gov
tel.: +1-708-252-7852

David **McAllester**
Massachussetts Institute of Technology
AI Laboratory
545 Technology Square
Cambridge MA 02139
USA
dam@ai.mit.edu
tel.: +1-617-253-6599

Michael A. **McRobbie**
Australian Nat. Uni.-Canberra
Centre for Information
Science Research
GPO. Box 4
Canberra ACT. 2601
Australia
Michael.A.McRobbie@arp.anu.edu.au
tel.: +61-6-249 2035

Kuniaki **Mukai**
Keio University
Fac. of Environmental Information
5322 Endo
Fujisawa 252
Japan
mukai@sfc.keio.ac.jp

Tobias **Nipkow**
TU München
Institut für Informatik
Arcisstraße 21
W-8000 München 2
Germany
nipkow@informatik.tu-muenchen.de
tel.: +49-89-2105-2690

Hans-Jürgen **Ohlbach**
Max-Planck-Institut für Informatik
Im Stadtwald
W-6600 Saarbrücken 11
Germany
ohlbach@mpi-sb.mpg.de
tel.: +49-681-302 5366

Uwe **Petermann**
Universität Leipzig
Institut für Informatik
Augustusplatz 10
O-7010 Leipzig
Germany
peterman@informatik.uni-leipzig.de
tel.: +49-41-719-2507

David **Plaisted**
University of North Carolina
Department of Computer Science
352 Sitterson Hall
Chapel Hill NC 27599 - 3175
USA
plaisted@cs.unc.edu

Andreas **Podelski**
Digital Equipment Corporation
Paris Research Laboratory
85 avenue Victor Hugo
F-92500 Rueil-Malmaison Cedex
France
podelski@prl.dec.com
tel.: +33-1-47 83 44 01

M. M. **Richter**
Universität Kaiserslautern
FB Informatik
Postfach 3049
W-6750 Kaiserslautern
Germany
richter@informatik.uni-kl.de

John Alan **Robinson**
Syracuse University
P.O. Box 51
Tully NY 13159-0051
USA
robinson@top.cis.syr.edu
tel.: +1-315-696 5281

Taisuke **Sato**
ETL Tsukuba
1-1-4 Umezono
Ibaraki 305
Japan
sato@etl.go.jp
tel.: +81-298-54-5624

Ken **Satoh**
Fujitsu Laboratories LTD
Knowledge Processing Laboratory
1015 Kamikodanaka
Nakahara-ku
Kawasaki 211
Japan
ksatoh@flab.fujitsu.co.jp
tel.: +81-44-754-2661

Peter **Schmitt**
Universität Karlsruhe
Fakultät für Informatik
Am Fasanengarten 5
W-7500 Karlsruhe
Germany
pschmitt@ira.uka.de
tel.: +49-721-6 08 40 00

Helmut **Schwichtenberg**
Universität München
Institut für Mathematik
Theresienstr.39
W-8000 München 2
Germany
schwicht@rz.mathematik.uni-muenchen.de
tel.: +49-89-2394-4413

Jörg **Siekmann**
DFKI Saarbrücken
Stuhlsatzenhausweg 3
W-6600 Saarbrücken 11
Germany
siekmann@dfki.uni-sb.de
tel.: +49-681-302-52 76

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

Mark **Stickel**
SRI International
333 Ravenswood Ave.
Menlo Park CA 94025
USA
stickel@ai.sri.com
tel.: +1-415-859-6151

Richard **Waldinger**
SRI International
333 Ravenswood Ave.
Menlo Park CA 94025
USA
waldinge@ai.sri.com
tel.: +1-415-859-2216

Christoph **Walther**
Technische Hochschule Darmstadt
Fachbereich Informatik
Alexanderstr. 10
W-6100 Darmstadt
Germany
walther@inferenzsysteme.informatik.th-darmstadt.de
tel.: +49-6151-16-44 92

Zuletzt erschienene und geplante Titel:

K. Compton, J.E. Pin , W. Thomas (editors):
Automata Theory: Infinite Computations, Dagstuhl-Seminar-Report; 28, 6.-10.1.92 (9202)

H. Langmaack, E. Neuhold, M. Paul (editors):
Software Construction - Foundation and Application, Dagstuhl-Seminar-Report; 29, 13.-17.1.92 (9203)

K. Ambos-Spies, S. Homer, U. Schöning (editors):
Structure and Complexity Theory, Dagstuhl-Seminar-Report; 30, 3.-7.2.92 (9206)

B. Booß, W. Coy, J.-M. Pflüger (editors):
Limits of Information-technological Models, Dagstuhl-Seminar-Report; 31, 10.-14.2.92 (9207)

N. Habermann, W.F. Tichy (editors):
Future Directions in Software Engineering, Dagstuhl-Seminar-Report; 32; 17.2.-21.2.92 (9208)

R. Cole, E.W. Mayr, F. Meyer auf der Heide (editors):
Parallel and Distributed Algorithms; Dagstuhl-Seminar-Report; 33; 2.3.-6.3.92 (9210)

P. Klint, T. Reps, G. Snelting (editors):
Programming Environments; Dagstuhl-Seminar-Report; 34; 9.3.-13.3.92 (9211)

H.-D. Ehrich, J.A. Goguen, A. Sernadas (editors):
Foundations of Information Systems Specification and Design; Dagstuhl-Seminar-Report; 35; 16.3.-19.3.9 (9212)

W. Damm, Ch. Hankin, J. Hughes (editors):
Functional Languages:
Compiler Technology and Parallelism; Dagstuhl-Seminar-Report; 36; 23.3.-27.3.92 (9213)

Th. Beth, W. Diffie, G.J. Simmons (editors):
System Security; Dagstuhl-Seminar-Report; 37; 30.3.-3.4.92 (9214)

C.A. Ellis, M. Jarke (editors):
Distributed Cooperation in Integrated Information Systems; Dagstuhl-Seminar-Report; 38, 5.4.-9.4.92 (9215)

J. Buchmann, H. Niederreiter, A.M. Odlyzko, H.G. Zimmer (editors):
Algorithms and Number Theory, Dagstuhl-Seminar-Report; 39; 22.06.-26.06.92 (9226)

E. Börger, Y. Gurevich, H. Kleine-Büning, M.M. Richter (editors):
Computer Science Logic, Dagstuhl-Seminar-Report; 40; 13.07.-17.07.92 (9229)

J. von zur Gathen, M. Karpinski, D. Kozen (editors):
Algebraic Complexity and Parallelism, Dagstuhl-Seminar-Report; 41; 20.07.-24.07.92 (9230)

F. Baader, J. Siekmann, W. Snyder (editors):
6th International Workshop on Unification, Dagstuhl-Seminar-Report; 42; 29.07.-31.07.92 (9231)

J.W. Davenport, F. Krückeberg, R.E. Moore, S. Rump (editors):
Symbolic, algebraic and validated numerical Computation, Dagstuhl-Seminar-Report; 43; 03.08.-07.08.92 (9232)

R. Cohen, R. Kass, C. Paris, W. Wahlster (editors):
Third International Workshop on User Modeling (UM'92), Dagstuhl-Seminar-Report; 44; 10.-13.8.92 (9233)

R. Reischuk, D. Uhlig (editors):
Complexity and Realization of Boolean Functions, Dagstuhl-Seminar-Report; 45; 24.08.-28.08.92 (9235)

Th. Lengauer, D. Schomburg, M.S. Waterman (editors):
Molecular Bioinformatics, Dagstuhl-Seminar-Report; 46; 07.09.-11.09.92 (9237)

V.R. Basili, H.D. Rombach, R.W. Selby (editors):
Experimental Software Engineering Issues, Dagstuhl-Seminar-Report; 47; 14.-18.09.92 (9238)

V. Dittrich, H. Hastedt, P. Schefe (editors):
Computer Science and Philosophy, Dagstuhl-Seminar-Report; 48; 21.09.-25.09.92 (9239)

R.P. Daley, U. Furbach, K.P. Jantke (editors):
Analogical and Inductive Inference 1992 , Dagstuhl-Seminar-Report; 49; 05.10.-09.10.92 (9241)

E. Novak, St. Smale, J.F. Traub (editors):
Algorithms and Complexity for Continuous Problems, Dagstuhl-Seminar-Report; 50; 12.10.-16.10.92 (9242)

J. Encarnação, J. Foley (editors):
Multimedia - System Architectures and Applications, Dagstuhl-Seminar-Report; 51; 02.11.-06.11.92 (9245)

F.J. Rammig, J. Staunstrup, G. Zimmermann (editors):
Self-Timed Design, Dagstuhl-Seminar-Report; 52; 30.11.-04.12.92 (9249)

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. Cellary, K. Vidyasankar , G. Vossen (editors):
Versioning in Database 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)

W. 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)

H. Kamp, J. Pustejovsky (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)

H.-D. Ehrig, F. von Henke, J. Meseguer, M. Wirsing (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)

Ch. Lengauer, P. Quinton, Y. Robert, L. Thiele (editors):
Parallelization Techniques for Uniform Algorithms, Dagstuhl-Seminar-Report; 66; 21.06.-25.06.93 (9325)

G. Farin, H. Hagen, H. Noltemeier (editors):
Geometric Modelling, Dagstuhl-Seminar-Report; 67; 28.06.-02.07.93 (9326)