MATHEMATISCHES FORSCHUNGSINSTITUT OBERWOLFACH

T a g u n g s b e r i c h t 19/1991

Deductive Systems

28. 04. – 04. 05. 1991

Die Tagung fand unter der Leitung von W. W. Bledsoe (Austin), G. Jäger (Bern) und M. M. Richter (Kaiserslautern) statt. Im Mittelpunkt stand das grosse Gebiet der deduktiven Systeme, das heute nicht nur in der Mathematik im Rahmen der mathematischen Logik sondern auch in der Informatik eine wichtige Rolle spielt.
Aus beiden Gebieten waren führende Vertreter anwesend.
Die Teilnehmer behandelten in ihren Vorträgen sowohl die mathematisch-logischen Aspekte dieses Generalthemas als auch zahlreiche Probleme, die mehr von praktischen Fragestellungen motiviert sind. Es wurde allgemein als besonders befruchtend empfunden, dass durch diese Tagung ein Forum für den Ideenaustausch zwischen zum Teil praktisch orientierten Wissenschaftlern aus dem Bereich des automatischen Beweisens und Logikern, die vor allem an den mathematischen Grundlagen dieses Gebietes interessiert sind, geschaffen wurde.
Etwas mehr präzisiert, wurden Vorträge zu folgenden Themen gehalten: Beweistheorie und Informatik, Logikprogrammierung, automatisches Beweisen, Typentheorien, nichtmonotones Schliessen, Komplexität von Beweisverfahren, modale Systeme.
Neben dem offiziellen Vortragsprogramm fanden mehrere Spezialsitzungen und Diskussionen statt, in denen Teilnehmer ihre Ergebnisse in grösserer Ausführlichkeit darstellten und allgemeine Perspektiven erörtert wurden. Gerade hier ergab sich die Möglichkeit für einen fruchtbaren Gedankenaustausch zwischen Teilnehmern, die sonst normalerweise keine Gelegenheit haben zusammenzukommen.

Abstracts

Alan Bundy:

The use of proof plans to guide automatic theorem provers

A proof plan is a computational representation of the overall structure or outline of a proof. We will give examples of proof plans and show how they are being used to assist the search for inductive proofs about recursive computer programs. A large family of inductive proofs have been analysed and common patterns extracted and expressed in the form of tactics. A tactic is a computer program that controls an automatic theorem prover, by applying rules of inference of its logic. Our tactics are Prolog programs which control OYSTER, a theorem prover for type theory. The effect of our tactics has been partially specified in a meta-logic. CLAM, an AI planning program, is used to reason with these specifications and form a customized proof plan for each conjecture input to OYSTER. These proof plans then guide OYSTER to a proof by running their constituent tactics.

Christoph Walther:

Discovering termination functions for algorithms computing normal forms

Proving the termination of a recursively defined algorithm usually requires a certain creativity of the (human or automated) theorem prover. One method is to invent a termination function τ, such that the τ-image of each argument in a recursive call is smaller (wrt. a given well-founded ordering) than the τ-image of the corresponding initial argument. Our working hypothesis is, that proving termination of algorithms can be considered as a program synthesis problem, where the algorithm is used as an incomplete and implicite specification of a termination function. Following this hypothesis, we present some first ideas for a formal framework, which allows to develop systematically termination functions for algorithms computing normal forms: Based on the given algorithm, we derive formal requirements, then we search for functions satisfying these requirements, and we finally combine these functions yielding a termination function for the given algorithm. We report on work, which has just begun and which ultimate aim is to develop a method for an automated synthesis of termination functions for this class of algorithms.

Amy Felty:

A logic programming approach to implementing higher-order rewrite systems

Term rewriting has proven to be an important technique in theorem proving. In this talk, we argue that rewrite systems and algorithms for first- and higher-order term rewriting can be naturally specified and implemented in a higher-order logic programming language. The logic programming language contains an implementation of the simply-typed lambda calculus including beta-eta conversion and higher-order unification on such terms. In addition, universal quantification in queries and the bodies of clauses is permitted. For higher-order rewriting, we show how these operations implemented at the meta-level provide elegant mechanisms for the object-level operations of descending through terms and matching terms with rewrite templates. We also discuss tactic style theorem proving in this environment and illustrate how term rewriting algorithms can be expressed as tactic-style search. The examples discussed in this talk have been implemented and tested in the logic programming language Lambda Prolog.

Helmut Schwichtenberg:

Higher order arithmetic

A system HOA of higher order arithmetic is described, which has been designed and implemented with applications in hardware verification in mind. It is based on the $\rightarrow\forall$-fragment of natural deduction together with induction axioms; e. g. boolean induction is used to prove the stability of atomic formulas. Hence we get classical logic in spite of the fact that we only have rules from minimal logic. This in turn has the consequence that proofs are λ-terms with recursion operators. HOA has a fixed intended semantics, with domains D^{ρ} consisting of Scott's (partial) continuous functionals extended by some non-monotonic objects like $=_{bit}$: $bit \rightarrow bit \rightarrow boole$ (where $bit := \{0, 1, undefined\}$). However, application Fg is defined for continuous g only. The implementation is in SCHEME; the special form of HOA makes it possible to implement normalization as evaluation. However, in order to make an evaluated proof (i. e. a procedure) visible again as a λ-term, we have to "invert evaluation". A consequence of the solution to this problem is the following Completeness Theorem (obtained together with U. Berger): Let M be a model of typed λ-calculus containing all primitive recursive functions (at level 1). Then from $M \models r = s$ we can conclude $\beta\eta \vdash r = s$. This strengthens results of Friedman 1975 and Statman 1982.

Rober Stärk:

The three-valued completion of logic programs

We present a new sequent calculus in which one can derive exactly those formulas which are true in all three-valued models of the completion of a logic program. The three-valued completion of a logic program provides a suitable declarative semantics for "Negation as Failure", since one can proof soundness and completeness of SLDNF-resolution (SLD-resolution plus negation as failure) for a large class of logic programs.
Our sequent calculus is proof theoretically interesting, since it is very close to the sequent calculus for classical two-valued logic. We omit sequents of the form $A \supset A$, and we add rules for the clauses of a program and some initial equality sequents. Having this sound and complete formalization of the three-valued completion we can give more direct proofs of some known theorems of logic programming. The calculus shows also that the three valued logic of logic programming is not a real three-valued logic.
If one extends the calculus by an infinite Herbrand rule one gets completeness for Herbrand models. Thus our approach shows the difference between true in all models and true in all Herbrand models of the completion.

John C. Shepherdson:

Lapses in logic programming: the role of standardising apart

Some of the basic results in the theory of logic programming, e. g. the mgu lemma, lifting lemma and completeness theorem have been incorrectly stated in the standard texts. One source of error was pointed out a few years ago, another very recently. Although these errors are elementary, and obvious once pointed out, they are not widely known. In one sense they are unimportant because the special cases of the theorems which are used in applications are correct. But these theorems are so fundamental that it is possible they might be used in an intricate and sensitive argument where the errors wold invalidate the result. And it is desirable that the results, particularly the fundmental results, of any widely used theory should be correctly stated. In this talk I will explain how these errors arise, and how they can be avoided by suitable 'standardising apart' of the program clauses used. The significance of standardising apart for other purposes e. g. soundness, obtaining most general answer, will also be discussed.

J. Strother Moore:

A new version of the Boyer-Moore theorem prover

The Boyer-Moore theorem prover, NQTHM, supports a first order, quantifier-free logic. The logic permits extensions by a schematic axiomatization of "new" inductively defined objects and the definition of total recursive functions. When viewed as a programming language, the logic resembles pure lisp. NQTHM is implemented in Common Lisp. The system provides an automatic theorem prover for the logic (which is perhaps best known for its heuristic use of induction) and mechanizations of the extension principles. The theorem prover is driven from a database of previously proved theorems. By presenting the system with an appropriately graduated sequence of theorems the careful user can lead the system to the proofs of deep results. Among the theorems so proved (checked ?) are Gauss' law of quadratic reciprocity, the Church-Rosser theorem, and Gödel's incompleteness theorem. The main application of NQTHM, however, is in the verification of computer hardware and software. For details, see *A computational Logic Handbook* (Academic Press, 1988).

We have recently undertaken the complete redevelopment of both the NQTHM logic and the implementation. Our aim is to build a practical implementation of the system in the applicative programming language the system supports. To achieve the requisite efficiency we have had to change the logic.

The new logic resembles applicative Common Lisp. It provides the data types RATIONALP (with subtype INTEGERP), CHARACTERP, STRINGP, SYMBOLP, and CONSP. It provides "packages" — thereby making it easier to develop disjoint systems of function and theorem names ("theories"). Unlike NQTHM, functions need not be defined on all inputs: "guards" may be used to restrict the domain. Functions for efficiently manipulating "property lists", "arrays", and "files" are provided in a completely applicative setting. "Multiple values" and "macros" are also supported.

All of NQTHM has been recoded in this language to support this language. Many heuristics have been been extended and improved, including type set, clausification, congruence based rewriting, and linear arithmetic. New features have been added including "functional instantiation", "encapsulation", "theories", and "books". Unlike NQTHM, it is now possible to load incrementally into the database the results of some other sessions. Online documentation is provided, along with extensive error checking and NQTHM's traditional error commentary. A "prototyping" mode permits the development and testing of large systems without the burden (or blessing) of proof. All of the new system is programmed in the logic.

The new system is not yet ready for public distribution.

Rusty Lusk:

Parallel theorem proving

Argonne theorem proving systems through the years — AURA, LMA/JTP, OTTER — have been characterized by efficient algorithms for computing (part of) the closure of a clause set under the operation of any of a set of inference rules. Here we present a parallel algorithm for closure computations and its implementation in an OTTER-compatible theorem prover called ROO. ROO runs on shared-memory multiprocessors and obtains linear (and often dramatically superlinear) speedups on large problems. We give performance results on several new, interesting problems in condensed-detachment logics, and illustrate the behavior of ROO with a graphics tool for visualizing parallel program execution.

Johann Makowsky:

On average case complexity of resolution for flat distributions

The resolution method for checking satisfiability of propositional clauses was shown to be exponential in the worst case. There are natural probability distribution an the clauses for which the satisfiability of propositional clauses is polynomial on the average, and other natural probability distributions for which a restricted form of resolution is exponential. In all these cases distributions are actually families of distributions on input of fixed size n. We study families of distributions on clauses of propositional logic af arbitrary size for which the satisfiability problem (SAT) is polynomial on the average and families of distributions for which resolution is exponential in the average. We show that there are many flat distributions for both of the cases. We also show that there are non-flat distributions for which SAT is polynomial on the average. The last result is important as SAT with a flat distribution is not DistNP complete, provided **DEXPT $\neq$ NEXPT**.

Arnon Avron:

Axiomatic systems, deduction and implication

The notions of logic, deduction, axiomatic systems and implication are investigated within the general framework of consequence relations (CRs). We distinguish between several types of CRs and define corresponding notions of deduction and of inclusion between logics. Given an axiomatic system, several CRs can naturally be associated with it according to these classifications. Each of them induces its own notion of derivable and admissible rules and of inclusion between systems. Several known systems are then identified as the minimal systems (according to some notion of inclusion) that contain an internal implication relative to the corresponding type of CR. To the same systems might naturally correspond other CRs as well. In the case of implicational linear logic, for example, these CRs have clear semantical interpretations and appropriate versions of the deduction theorem hold for them, but unlike the principal associated CR, they are not known to be decidable.

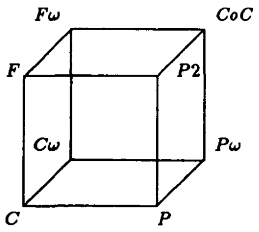
Daniel Mey:

Investigations on a predicate calculus without contractions

The predicate calculus LS is defined. It contains no rules for contraction and no cut rule. After establishing decidability for LS and other calculi without contractions, a closer look at LS-provability is taken by investigating a special order between formulas. An algebraic semantics and a game theoretical interpretation of the calculus are presented. Finally, a relation between resolution and calculi without contractions is formulated.

Gérard Huet:

Variations on the cube



We present Barendregt's cube of λ-calculi. C is the ordinary Curry-Church simply typed λ-calculus. The representation of recursive functions over Church numerals is discussed, and the problem of type-checking the exponential motivates the introduction of type polymorphism, leading to Girard's system F, also known as Reynolds' polymorphic lambda calculus $\lambda 2$. The internalization of product formation motivates the introduction of type operators, by considering a copy $C\omega$ of C at the level of types. The union of $C\omega$ and F leads to higher-order λ-calculus $F\omega$, due to Girard. It can be seen that these systems express various dependencies: elements depend on elements in C, elements depend on types in F, types depend on types in $C\omega$. Considering the dependency of types on elements give dependent types, system P. Adding all these possibilities lead to the original Calculus of Constructions CoC, due to Coquand.
The Curry-Howard isomorphism permits to interpret these functional type systems as fragments of intuitionistic logic.
The eight systems of the cube can be uniformly presented as functional GTS over two sorts. The notion of GTS, or generalized type system, is due to Berardi and Geuvers. It gives a parametric presentation of type systems with a product operation over sorts axiomatized by axioms $s_1 : s_2$ and relations (s_1, s_2, s_3), meaning "we allow product formation $\Pi x{:}A.B(x)$ where A is a type of sort s_1 and then $B(x)$ is a type of sort s_2, obtaining type of sort s_3. In functional GTS $s_2 = s_3$. Type equality in a GTS is β-conversion. Examples of other GTS, over three sorts, are the Automath language and Church's higher order logic.

The first variation is to allow η-conversion — for instance, Edinburgh Logical Framework (LF) is equivalent to $P + \eta$. Another variation is to permit sum types, and more generally inductive types, like in Martin-Löf type theory. Adding ordered sorts allows the introduction of a cumulative hierarchy of universes.

Woody Bledsoe:

Some activities of the Bledsoe-Hines group in Austin

I will discuss our STR+VE prover (briefly) which proves theorems in general inequalities, and is complete for FOL (Hines). It is able to prove lim+ (sum of continuous functions is continuous) and lim* (extension due to Hines), and IMV-FOL, and many others, which offer a difficult challenge to automated theorem provers.
I will also discuss briefly our prover caller "TOOL" (2OL), which is used to prove certain theorems in second order logic where simple set variables are to be instantiated. IMV-HOL is such a theorem: $Cont\ f \wedge f(a) < 0 \wedge f(b) > 0 \wedge a \leq b \rightarrow \exists x(a \leq x \leq b \wedge f(x) = 0)$ (using the least upper bound axiom). IMV-FOL is a first order logic version of this.
And I show how we plan to explore further into elementary real analysis with these provers and extensions of them.

Wolfgang Bibel:

Connection Calculi

The connection method provides a general framework for the comparison and development of deductive calculi. Consolution and the connection structure calculus by Eder as well as the pool calculus by Neugebauer and Schaub are new connection calculi of this kind. These calculi also provide a new insight into resolution; for instance, resolution turns out to be a special case of consolution. The specifics of many calculi, including various refinements of resolution, can uniformly be understood as a combination of a few key features such as linear chaining, hinged loops, and factorization. This work aims not only at a better understanding of deductive systems based on any of these calculi but also at the enhancement of the performance of existing systems (such as SETHEO). Beyond these features characterizing existing systems, the principle of compression and global, higher features such as lemmas with renaming are presented as of importance for future systems. Recent results are given for both, namely cycle unification for special classes of formulas for the former and linear proofs for the latter.

Donald W. Loveland:

The METEOR implementations of the Model Elimination procedure

The Model Elimination (ME) proof procedure, now 25 year old, has had several recent implementations: first, the PTTP system of Stickel, and very recently, PARTHENON (Bose, Clarke et al.), SETHO-PARTHEO (Letz, Schumann et al.) and the METEOR variants, implemented by Owen Astrachan at Duke University. There are sequential, parallel and distributed network versions of the METEOR program.
ME is a complete proof procedure for first-order logic using a (linear) input format, which allows the Warren Abstract Machine concepts developed for Prolog to be used also to implement ME. This includes parallel implementations. Although the depth-first search (using iterative deepening) allows no retention of intermediate results, the very high inference speed provides compensation. Using the ME depth-first strategy METEOR recently proved two of Bledsoe's challenge problems concerning the sum of two continuous functions, a first for general-purpose (uniform) proof procedures. We now seek to install caching and lemma use to provide METEOR with intermediate results.

David A. Plaisted:

Search duplication in theorem proving

We define an abstract concept of a theorem proving search strategy and consider the kinds of redundancies that occur. We consider only clause form refutational first-order theorem proving. The redundancies relate to how often a given instance of a given input clause is used in the search. We consider a number of well-known theorem proving strategies and show that they either have exponential redundancy or are not sensitive to the goal (theorem being proved). We also present a recent method, called clause linking, that reduces this redundancy and give experimental evidence that it is faster than other methods on propositional and near-propositional problems. This method is also sensitive to the theorem being proved. With a more efficient rule of inference, we can begin to study higher level issues such as semantics.

Egon Börger:

Formal analysis of Prolog database views and their uniform implementation

A distinguishing feature of logic programming is a realization of the notion of deduction from time dependent sets of axioms. In Prolog this is reflected by the database's being subject to change during the computation due to bips assert(c), retract(c) etc. The ISO WG 17 in its Nov '90 draft has proposed a new liberal view on dynamic code in standard Prolog — which includes immediate and logical update view, among others. We give a formal analysis of this new ISO WG 17 DB view and develop an abstract uniform implementation for it, which clarifies the concept and shows the tradeoff between logic and efficiency.
Our DB model is based on Prolog algebras (introduced in Börger, MFCS '90 and CSL '89) and on WAM algebras (defined in Börger and Rosenzweig, CSL '90, to prove the correctness of Warren's abstract machine wrt the abstract specification of full Prolog by Prolog algebras).
This work is joint work with D. Rosenzweig/Zagreb.

Markus Marzetta:

Types and names

Like Feferman's theories for explicit mathematics, to which they are closely related, theories of types and names can serve to study various kinds of type theories. In computer science they can also be used to state and prove properties of functional programs. These theories deal with a universe of computational objects which constitutes a partial combinatory algebra and includes the natural numbers. The objects of this universe are classified into types which are treated extensionally. *Names* (explicit representations in the universe) are associated in a very uniform way to types. We present several theories obtained by allowing various type constructions and forms of induction and examine their proof-theoretic strength. Furthermore we consider the role of *universes*, seen as collections of names of types satisfying certain closure conditions. The limes axioms states that every (name of a) type belongs to a universe. Adding the axioms for universes to the elementary theory of types and names gives a system of strength Γ_0.

Stan Wainer:

Ordinal analysis for recursive definitions

Classical (Gentzen-style) proof theory gives for example the reduction

$$\mathrm{PA} \vdash \forall x \exists y \mathrm{Spec}(x,y) \quad \Longrightarrow \quad \mathrm{PRA}+\mathrm{TI}(\alpha) \vdash \forall x \exists y \mathrm{Spec}(x,y)$$

Gefördert durch DFG Deutsche Forschungsgemeinschaft

where, if k, r are the size, rank of the PA-proof then $\alpha = \left.\omega^{\cdot^{\cdot^{\cdot^{\omega^{\omega k}}}}}\right\}r$. Thus α bounds the "termination complexity" of the specified program, and its "computational complexity" is bounded by the "fast-growing" function B_α, where $B_0(n) = n + 1$, $B_{\alpha+1}(n) = B_\alpha(B_\alpha(n))$, $B_\lambda(n) = B_{\lambda_n}(n)$. In this talk, the logic is stripped away in order to give a direct ordinal assignment to computations $n: N \vdash^{\alpha}_{E} f(n): N$ in Kleene's equation calculus E. The B_α's again bound complexity, and reduction to the corresponding "slow-growing" G_{α^+} provides an ordinal trade-off in reduction of computation to term-rewriting. Example (Cichon): Primitive Recursion = "Recursive Path" Termination.

Bernhard Hollunder:

Inferences in KL-ONE based knowledge representation systems

We investigate algorithms for inferences in KL-ONE based knowledge representation systems. Such systems employ two kinds of formalism: the terminological and the assertional formalism. The terminological formalism consists of a concept language to define concepts and relations between concepts for describing a terminology. On the other hand, the assertional formalism allows to introduce objects, which are instances of concepts and relations of a terminology. We present algorithms for inferences such as

- determining subsumption relations between concepts
- checking consistency of such a knowledge base
- computing the most specific concepts an object is instance of
- computing all objects that are instances of a certain concept.

Christian Horn:

Theorem proving and program synthesis in Martin-Löf type theory using Oyster2

Oyster2 is an interactive, tactic driven, backward reasoning proof editor for Martin-Löf type theory. It was developed with an eye towards easy modifyability of the underlying language and logic, so that we could try to approximate a suitable object language which combines logical simplicity and strength with the expressivity and adequacy required for practical applications, but still allows to maintain a simple and by its very nature reliable and correct system architecture. The main goal was to perform experiments to find an appropriate level of intelligent support in the theorem proving and program synthesis process, where we can't achieve full automation. The use of structural models and higher order reasoning are essential for the area of theorem proving we are interested in.
The talk presents an overview of the development and architecture of the oyster2 system, and will highlight the problematic points, i. e. necessary extensions of type theory and a technological model of the proof engineering process, where I hope to get reasonable feed back during the discussions.

H. J. Ohlbach:

Deductive systems for logics with possible world semantics

Logics with possible world semantics are built on the concept of states and state transitions. The basic logics of this kind are modal logics with the two operators $\Box$ (necessarily) and $\diamond$ (possibly). More application oriented are extensions like temporal logics, action logics, epistemic logics (logics of knowledge), doxastic logics (logics of belief) etc. Most of these logics require the modal operators to be parametrized with terms denoting actions, agents and the like.

In the talk I will first present an augmented semantics for logics with possible world semantics which emphasizes the role of the parameters of the operators and their correspondences. This semantics supports the development of new applications of the idea of possible worlds. One application is a full first order probabilistic logic where the parameters of the modal operators are interpreted as probability values (either real numbers or qualitative values). Another application is an action logic with built in facilities for hierarchical planning.
In the second part I will present a method for translating formulae of these logics into first order predicate logic such that standard predicate logic deduction methods become applicable. Moreover, the translation method permits the transformation of the characteristics of the particular logic into efficient theory unification and theory resolution algorithms.

Wolfgang Schönfeld:

Backtracking and minimal tableau proofs

Let Σ be a satisfiable set of formulas of fist-order logic, and γ a "goal" formula. A tableau proof for (the unsatisfiability of) $\Sigma \cup \{\gamma\}$ is *minimal* if no application of a formula can be removed. We show that any loop-free and connected (neighbored formulas contain a complementary pair of literals) tableau proof is minimal. Thought not every minimal proof is connected, there are enough connected proofs:
Theorem: $\Sigma \cup \{\gamma\}$ is unsatisfiable iff there is a connected tableau proof for $\Sigma \cup \{\gamma\}$. To give a constructive proof of the only-if-part (completeness), we extend the classical tableau construction to that of *alternating* tableau. They not only describe the complete search space indicating where and how to backtrack. They also indicate a potential for *intelligent* backtracking avoiding permutations of useless proof attempts. This even applies to the case of propositional logic. (For the Prolog case, it only makes sense for non-propositional formulas.)

Peter H. Schmitt:

Nonmonotonic abstract consequence operators

After a quick reminder of the first attempt to formulate axioms typical for logical consequence operators by A. Tarski, we turn to axiom systems where the monotony property $X \subseteq Y \Longrightarrow Cn(X) \subseteq Cn(Y)$ is replaced by the cumulativity property $X \subseteq Y \subseteq Cn(X) \Longrightarrow Cn(X) = Cn(Y)$. Two characterization theorems are presented: One by Helmut Thiele, where Cn among others satisfies the closure property $Cn(Cn(X)) = Cn(X)$ and the semantic model is an abstraction of Reiter's default logic. The second theorem reviewed is by Kreus, Lehmann and Magidor. The corresponding consequence operator Cn only operates on finite sets and adding the requirement that Cn be a closure operator would turn Cn into a monotonic operator.

Peter Aczel:

The notion:"A logic"

This notion and other informal notions are often used without being given mathematical definitions. But in recent years the confrontation of logic with category theory and computer science has led to new attempts to capture mathematically some of these notions, sometimes without careful attention to their history. In my talk I will focus on the following, apparently conflicting requirements:

(i) Although any given logic will take some particular approach to syntax, the general notion of a logic should not.

(ii) The schematic nature of rules of inference is fundamental. The notion of a derived rule should have a syntax free definition.

Berichterstatter: R. Stärk

Tagungsteilnehmer

Prof.Dr. Peter Aczel
Dept. of Computer Science
University of Manchester
Oxford Road

GB- Manchester M13 9PL

Prof.Dr. Alan Bundy
University of Edinburgh
Department of AI
80 South Bridge

GB- Edinburgh EH1 1HN

Prof.Dr. Arnon Avron
Computer Science Department
School of Mathematics
Tel Aviv University

Tel Aviv
ISRAEL

Prof. Dr. Martin Davis
New York University
Courant Institute of Mathematical Sciences
251 Mercer Street

New York , NY 10012
USA

Prof.Dr. Wolfgang Bibel
Fachbereich Informatik
Technische Hochschule Darmstadt
Alexanderstraße

6100 Darmstadt

Prof.Dr. Walter Felscher
Mathematisches Institut
Universität Tübingen
Auf der Morgenstelle 10

7400 Tübingen 1

Prof.Dr. Woodrow W. Bledsoe
Dept. of Mathematics
University of Texas at Austin

Austin , TX 78712
USA

Amy Felty
INRIA Rocquencourt
Domaine de Voluceau
B. P. 105

F-78153 Le Chesnay Cedex

Prof.Dr. Egon Börger
Dipartimento di Informatica
Universita di Pisa
Corso Italia, 40

I-56100 Pisa

Dr. Larry Hines
University of Texas
ICSCA

Austin , TX 78712
USA

Bernd Hollunder
Deutsches Forschungszentrum für
künstliche Intelligenz (DFKI)
Projektgruppe WINO
Postfach 2080

6750 Kaiserslautern

Dr. Ewing Lusk
Mathematics and Computer Science
Division - 221 - MCS
Argonne National Laboratory
9700 South Cass Avenue

Argonne , IL 60439-4844
USA

Dr. Christian Horn
Fachhochschule Furtwangen
Fachbereich Allgemeine Informatik
PF 28

7743 Furtwangen

Prof.Dr. J.A. Makowsky
Computer Science Department
TECHNION
Israel Institute of Technology

Haifa 32000
ISRAEL

Prof.Dr. Gerard Huet
INRIA Rocquencourt
Domaine de Voluceau
B. P. 105

F-78153 Le Chesnay Cedex

Dr. Markus Marzetta
Institut für Informatik und
angewandte Mathematik
Universität Bern
Länggasstr. 51

CH-3012 Bern

Prof.Dr. Gerhard Jäger
Institut für Informatik
und angewandte Mathematik
Länggasstraße 51

CH-3012 Bern

Dr. Daniel Mey
Institut für theoretische
Informatik
ETH-Zentrum

CH-8092 Zürich

Prof.Dr. Donald W. Loveland
Department of Computer Science
Duke University
202 North Building

Durham , NC 27706
USA

Prof.Dr. Dale Miller
Dept. of Computer and Information
Science
University of Pennsylvania
200 South 33rd Street

Philadelphia , PA 19104-6389
USA

Gefördert durch
DFG Deutsche Forschungsgemeinschaft

Dr. J. Strother Moore
Computational Logic Inc.
1717 West 6th St., Suite 290

Austin , TX 78703
USA

Prof.Dr. Michael M. Richter
Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049

6750 Kaiserslautern

Dr. Hans Jürgen Ohlbach
FB Informatik, Geb. 57
Universität Kaiserslautern

6750 Kaiserslautern

Prof.Dr. Ulf R. Schmerl
Fakultät für Informatik
Universität der Bundeswehr München
Werner-Heisenberg-Weg 39

8014 Neubiberg

Dr. Ross Overbeek
Mathematics and Computer Science
Division - 221 - MCS
Argonne National Laboratory
9700 South Cass Avenue

Argonne , IL 60439-4844
USA

Prof.Dr. Peter H. Schmitt
Fakultät f. Informatik, Institut f.
Logik,Komplexität u.Deduktionssyst.
Unviversität Karlsruhe
Postfach 6980

7500 Karlsruhe

David Plaisted
Dept. of Computer Science
University of North Carolina
352 Sitterson Hall

Chapel Hill , NC 27599-3175
USA

Prof.Dr. Wolfgang Schönfeld
IBM Deutschland GmbH
Wissenschaftliches Zentrum - IWBS
Tiergartenstraße 15

6900 Heidelberg

Prof.Dr. Wolfram Pohlers
Institut für Mathematische
Logik und Grundlagenforschung
Universität Münster
Einsteinstr. 62

4400 Münster

Prof.Dr. Helmut Schwichtenberg
Mathematisches Institut
Universität München
Theresienstr. 39

8000 München 2

Prof.Dr. John C. Shepherdson
Department of Mathematics
University of Bristol
University Walk

GB- Bristol , BS8 1TW

Robert Stärk
Institut für Informatik und
angewandte Mathematik
Universität Bern
Länggasstr. 51

CH-3012 Bern

Dr. Stan S. Wainer
School of Mathematics
University of Leeds

GB- Leeds , LS2 9JT

Prof.Dr. Christoph Walther
Institut für Theoretische
Informatik
Technische Hochschule Darmstadt
Alexanderstr. 10

6100 Darmstadt

Aczel, Peter	petera@cs.man.uk
Avron, Arnon	aa@taurus.bitnet
Bibel, Wolfgang	xiiswbib@ddathd21.bitnet
Bledsoe, Woody	bledsoe@cs.utexas.edu
Börger, Egon	boerger@dipisa.di.unipi.it
Davis, Martin	davism@cs.nyu.edu
Felscher, Walter	mife001@mailserv.zdv.uni-tuebingen.de
Felty, Amy	felty@margaux.inria.fr
Hines, Larry	hines@cs.utexas.edu
Hollunder, Bernhard	hollunde@dfki.uni-kl.de
Horn, Christian	c-horn@eis.fh-furtwangen.dbp.de
Huet, Gérard	huet@margaux.inria.fr
Jäger, Gerhard	jaeger@iam.unibe.ch
Loveland, Donald W.	dwl@cs.duke.edu
Lusk, Ewing (Rusty)	lusk@mcs.anl.gov
Makowsky, J. A.	janos@cs.technion.ac.il, janos@techsel.bitnet
Marzetta, Markus	marzetta@iam.unibe.ch
Mey, Dani	mey@inf.ethz.ch
Miller, Dale	dale@cis.upenn.edu
Moore, J. Strother	moore@cli.com
Plaisted, David A.	plaisted@cs.unc.edu
Pohlers, Wolfram	oml02@dmswwu1a.bitnet
Richter, Michael M.	mmr@informatik.uni-kl.de
Schmitt, Peter H.	pschmitt@ira.uka.de
Schönfeld, Wolfgang	schfeld@dhdibm1.bitnet
Shepherdson, John C.	John.Shepherdson@bristol.ac.uk
Stärk, Robert F.	staerk@iam.unibe.ch
Wainer, Stan	pmtbssw@uk.ac.leeds.cms1
Walther, Christoph	xiiscwal@ddathd21.bitnet

Gefördert durch
DFG Deutsche Forschungsgemeinschaft