Dagstuhl Report No. 300

Deduction

Ulrich Furbach
Universität Koblenz-Landau
Fachbereich Informatik
Rheinau 1
56075 Koblenz
Germany
E-mail: uli@uni-koblenz.de

Harald Ganzinger
Max-Planck-Institut für Informatik
Programming Logics Group
Im Stadtwald
D-66123 Saarbrücken
Germany
E-mail: hg@mpi-sb.mpg.de

Ryuzo Hasegawa
Department of Electronics
Kyushu University 36,
Fukuoka 812,
Japan
E-mail: hasegawa@ele.kyuhsu-u.ac.jp

Deepak Kapur
University of New Mexico
Dept. of Computer Science
Farris Eng. Center # 339
NM87131 Albuquerque, USA
E-mail: kapur@cs.unm.edu

This report[1] covers the seminar no. 01101 on *Deduction*, held at Dagstuhl, Germany during March 4–March 9, 2001. This seminar was organized by U. Furbach (Koblenz, Germany), H. Ganzinger (Saarbrücken, Germany) and D. Kapur (Albuquerque, USA). It brought together about 50 researchers from various countries.

Dagstuhl, a place being developed exclusively for research activities in Computer Science, provides an excellent atmosphere for researchers to meet and exchange ideas. During this seminar we had 40 talks and a discussion session on an 'open source software repository'.

[1]Compiled by Peter Baumgartner and Jan Murray, Universität Koblenz-Landau.

Contents

1 A Note from the Organizers

Logic has become a prominent formal language and formalism for all of computer science. It serves in many applications such as in problem specification, program development, program transformation, program verification, hardware design and verification, consistency checking of databases, theorem proving, expert systems, logic programming, and so on and so forth. Its strength derives from the universality of the language as well as from the fundamental logical operations and relations. Logical manipulations as needed in all these applications are realized by mechanisms developed in the field of deduction which has produced a variety of techniques of great importance in all these applications.

During the last years successful research has led to the development of high performance deduction systems, and to laying a broad basis for various applications. This success of deduction can be observed within the international AI and computer science scene as well. Deduction systems recently have achieved considerable successes and even public press: it was a first-order theorem prover which first proved the Robbins algebra conjecture and even reached the New York Times Science section. But not only in proofing mathematical theorems, also in various other disciplines of AI, automated deduction made substantial progress. In planning, for example, it turned out that propositional theorem provers are able to outperform special purpose planning systems. This is remarkable, since it was considered folklore that planning requires specialized algorithms, which was only recently disproved by the development of propositional satisfiability testing methods which can now handle much larger planning problem sizes. A very similar development can be observed in the field of model based diagnosis.

It is the idea of this Dagstuhl-Seminar to bring together leading scientists in the area of Deduction from all over the world. By all participants the previous seminars in 1993, 1995, 1997 and 1999 were unanimously considered great successes. The Dagstuhl-Seminar Reports No. 58, 110, 170 and 232 of these seminars, together with this one, reflect the development of the entire discipline over the last 10 years.[2]

Ulrich Furbach, Harald Ganzinger, Ryuzo Hasegawa, Deepak Kapur

[2]They are still available in the Dagstuhl Office or on the Dagstuhl Web-pages.

2 Abstracts

Path-indexing with database joins for efficient retrieval of instances and backward subsumption

Alexandre Riazanov, Andrei Voronkov, University of Manchester, Great Britain

If we represent a term as a tree with nodes marked by functional symbols, and edges marked by the corresponding argument numbers, any path in this tree is called a path in the original term. The main idea behind path-indexing for retrieving instances of a given term from a given set of terms is as follows: every instance of a query term q must contain all the paths that are present in q. The standard path-indexing technique ([1],[2],[3]) for instance retrieval does not provide perfect filtering: it simply filters out those terms from the set of indexed terms that do not contain all the necessary paths and then we have to perform the matching test on all the remaining terms.

We present an indexing technique for instance retrieval based on path-indexing and database joins. When a term is integrated into the index, its subterms are stored in special tables, called path relations, corresponding to paths in which these subterms occur. This allows us to check equality subterms assigned to different occurences of the same variable in the query term. This check is expressed as database-style joins on the corresponding path relations. This makes our indexing technique a perfect filter. Moreover, we show how the same technique can be used as a very efficient perfect filter for backward subsumption. This is done by simply adding fields in the path relations, containing numbers of the literals in the indexed clauses, and performing joins on these fields. And, finally, we show that our technique can be easily modified for efficient specialised treatment of symmetric predicates and commutative functions.

The proposed indexing technique for backward subsumption has been implemented in our theorem prover Vampire [5]. In our early experiments we discovered that the naive representation of path relations as sorted linked lists leads to unacceptably bad performance. Fortunately, there is a data structure called skip-lists ([4]) that is ideally suited for this task. With a few related optimisations, this resulted in a very efficient implementation.

References

[1] M. Stickel. The path indexing method for indexing terms. Technical Report 473, Artificial Intelligence Center, SRI International, Menlo Park, CA, 1989.

[2] P. Graf. *Term Indexing*, volume 1053 of *Lecture Notes in Computer Science*. Springer Verlag, 1996.

[3] William W. McCune. Experiments with discrimination-tree indexing and path indexing for term retrieval. *Journal of Automated Reasoning*, 9(2), 1992.

[4] W. Pugh. Skip Lists: A Probabilistic Alternative to Balanced Trees. CACM, 33(6), 1990

Redundancy Elimination in Equational Theorem Proving

Jürgen Avenhaus, B. Löchner, Universität Kaiserslautern, Germany

Redundancy elimination is a key feature to make theorem provers efficient. But it is known that detecting redundancy is very expensive in practice. We describe a cheap method to detect that an equation is redundant with respect to a set of equations. It is based on testing ground joinability. Ground joinable equations can be deleted and this results in saving work for computing new consequences of this equation. But, deleting a redundant equation also weakens the power of simplification and so of pruning the search space. Hence we have sharpened the test for ground joinability and now can identify redundant equations that need not be used to compute critical pairs but can still be used for simplification. The main advantage is that this test is very cheap and so has a good cost benefit ratio. This test constitutes a main reason why our system Waldmeister is one of the strongest provers for pure equational logic and is especially efficient in handling AC-symbols.

Termination and Reduction Checking for Higher-Order Logic Programs

Brigitte Pientka, Carnegie Mellon University, USA

We present a syntax-directed termination and reduction checker for higher-order logic programs. The reduction checker verifies parametric higher-order subterm orderings describing input and output relations. These reduction orderings are exploited during termination checking to infer that a specified termination order holds. To reason about parametric higher-order subterm orderings, we introduce a deductive system as a logical foundation for proving termination. This allows the study of proof-theoretical properties, such as consistency, local soundness and completeness and decidability. We concentrate here on proving consistency of the presented inference system. The

termination and reduction checker are implemented as part of the Twelf system and enables us to verify proofs by complete induction.

Executing Higher Order Logic

Stefan Berghofer, TU München, Germany

Executing formal specifications has been a popular research topic for some decades, covering every known specification formalism. Executability is essential for validating complex specifications by runnig test cases and for generating code automatically ("rapid prototyping"). In the theorem proving community executability is no less of an issue. Two prominent examples are the Boyer-Moore system (and its successor ACL2) and constructive type theory, both of which contain a functional programming language.

We report on the design of a prototyping component for the theorem prover Isabelle/HOL. We give a precise definition of an executable subset of HOL and describe its compilation into a functional programming language. The executable subset contains datatypes and recursive functions as well as inductive relations. Inductive relations must be such that they can be executed in Prolog style but requiring only matching rather than unification. This restriction is enforced by a mode analysis. Datatypes and recursive functions compile directly into their programming language equivalents, and inductive relations are translated into functional programs using mode information and standard programming techniques for lazy lists.

Our aim has not been to reach or extend the limits of functional-logic programming but to design a lightweight and efficient execution mechanism for HOL specifications that requires only a functional programming language and is sufficient for typical applications like excution of programming language semantics or abstract machines.

Elements of Theorem Proving Intelligence

David A. Plaisted, University of North Carolina, USA

We review the history of AI, emphasizing the distinction between strong and "weak" methods. We also review the history of the speaker's own research in automated deduction. Current automated deduction programs have too few mechanisms. In order to increase the power of automated theorem provers, more mechanisms need to be present in the same theorem prover. When enough mechanisms are present, there may be a dramatic increase in the power of theorem provers. Mechanisms that are neglected in current provers

include the use of natural semantics, case analysis, relevance, and (sometimes) goal-sensitivity. The performance of resolution on logic puzzles and problems involving definition expansion dramatically illustrates the weakness of resolution on some easy problems. The OSHL (ordered semantic hyperlinking) strategy combines these neglected mechanisms and sometimes far outperforms resolution, despite being implemented in Prolog. Some recent experiments on set theory problems show that a natural semantics for set theory significantly improves the performance of OSHL, and a natural weighting of terms to prefer terms likely to occur in a proof gives a further significant improvement. Simple rules, often mechanically derivable from the axioms, give such natural term weightings. By combining these rules with custom weightings given by the user, a further improvement in the performance of OSHL is possible.

On the representation of search spaces in theorem proving: from forward to backward reasoning

Maria Paola Bonacina, University of Iowa, USA

This talk is concerned with theorem proving as a search problem, and investigates what is the search space of some of the most popular, refutational, clausal strategies for fully automated theorem proving, with the purpose of providing a unified framework for representing search space and modelling search behavior. Such a framework would offer a common basis for developing a better understanding of control issues in deduction, hence possibly better search plans, and for comparing strategies.

Strategies covered include ordering-based strategies, such as those based on (ordered) resolution, (ordered) paramodulation/ superposition, simplification and subsumption, that are synthetic in nature, and subgoal-reduction strategies, both synthetic ones, such as those based on linear resolution and model elimination with chains, and analytic ones, such as those based on tableaux.

The talk surveys the application of the concepts of state space and closure to these strategies, and discusses the limitations of these notions. A representation of search space called marked search graph is proposed as a better suited alternative, and shown to apply to all strategies above, thus succeeding in capturing both forward reasoning, synthetic, ordering-based strategies with contraction, and backward reasoning, analytic, subgoal-reduction strategies with backtracking.

Incorporating Bidirectional Relevancy into Model Generation Theorem Provers

Adnan Yahya (joint work with Donald Loveland), Duke University, Durham, USA

Model-generation theorem provers, SATCHMO-style, can explore a larger than needed search space when answering a query. Partial and total relevancy were suggested as a mechanisms to limit the search space to clauses that are needed for the refutation. SATCHMORE was an implementation of the latter. SATCHMORE relevancy, however, is driven by the entire set of negative clauses of the theory and no distinction is accorded to the query negation.

Under unfavorable circumstances, such as in the presence of large amounts of negative data, this can reduce efficiency. In this lecture we define a further refinement of that uses only the negation of the query for relevancy determination at the start. Other negative clauses are introduced on demand and only if a refutation is not possible using the current set of negative clauses. The search for the relevant negative clauses is performed in a forward chaining mode as opposed to relevancy propagation in SATCHMORE which is based on backward chaining. The approach is shown to be refutationally sound and complete. Experiments on a prototype implementation point to its potential to enhance the efficiency of the query answering process in disjunctive databases.

Incorporating Lemmas into Consequence-finding Procedure SOL

Katsumi Inoue (joint work with Koji Iwanuma, Takashi Matsuda, Hiromasa Haneda, and Ken Satoh), Kobe University, Japan

SOL resolution is a consequence-finding procedure based on Model Elimination, and is complete for finding characteristic clauses. Previous implementation of SOL resolution often recomputes the same subgoals. In the new version of SOL resolution, the calculi are defined with the connection tableau, and avoids much redundancy in SOL resolution using mandatory operations, cutting-off rules, lemmas and folding-up, and skip pruning.

Learning Methods to Prove Theorems

Manfred Kerber (joint work with Mateja Jamnik and Christoph Benzmüller), University of Birmingham, Great Britain

A framework for automated learning within mathematical reasoning systems is presented. This framework enables proof planning systems to automatically learn new proof methods from well chosen examples of proofs which

use a similar reasoning strategy to prove related theorems. The framework consists of a representation formalism for methods and a machine learning technique which can learn methods using this representation formalism. The aim is to emulate some of the human informal mathematical reasoning, in particular the human learning capability, on machines. This work bridges two areas of research, namely it applies machine learning techniques to advance the capability of automated reasoning systems.

Deciding Fluted Logic with Resolution

Renate A. Schmidt and Ullrich Hustadt, Department of Computer Science, University of Manchester and Department of Computer Science, University of Liverpool, Great Britain

Fluted logic is a solvable fragment of first-order logic which is a by-product of Quine's predicate functor logic. In fluted logic the arguments of atoms are sequences of variables in a fixed order and a quantifier can only bind the free variable which is largest according to this ordering. We contrast the way decidability is obtained in fluted logic to other solvable first-order fragments which are more well-known, in particular, quantifier prefix classes, the guarded fragment, the two-variable fragment, the monadic class, and Maslov's class K. Fluted logic is also of interest for its relationship to non-classical logics. Just like the guarded fragment, fluted logic may be viewed as a first-order generalisation of propositional modal logic which has many of the pleasant properties that modal logics have. It turns out that fluted logic subsumes a large class of enriched modal logics. Interestingly, there is more than one way of translating propositional modal formulae into fluted formulae. In the context of resolution fluted logic can be charaterised by a class of fluted clauses. Our decision procedure for this class is based on a standard ordering refinement of resolution and an additional *separation rule*. This is a new inference rule which does a form of dynamic renaming. Formally:

$$\frac{N \cup \{C \vee D\}}{N \cup \{\neg q(x_1, \ldots, x_n) \vee C,\ q(x_1, \ldots, x_n) \vee D\}}$$

provided (N denotes a set of clauses) (i) the clause $C \vee D$ is separable into clauses C and D, that is, $\mathrm{var}(C) \not\subseteq \mathrm{var}(D)$ and $\mathrm{var}(D) \not\subseteq \mathrm{var}(C)$, (ii) $\mathrm{var}(C) \cap \mathrm{var}(D) = \{x_1, \ldots, x_n\}$ for $n \geq 0$, and (iii) q does not occur in N, C or D. The rule is sound, in general, and resolution extended by this rule remains complete, if it is applied finitely often.

Redundancy Checking

Hans de Nivelle, MPI für Informatik, Saarbrücken, Germany

Deriving and Applying Program Synthesis Calculi

David Basin, University of Freiburg, Germany

Over the last decade I have worked with colleagues on several different projects to develop, implement, and automate the use of calculi for program synthesis and transformation. These projects had different motivations and goals and differed too in the kinds of programs synthesized (e.g., functional programs, logic programs, and even circuit descriptions). However, despite their differences they were all based on three simple ideas. First, calculi can be formally derive in a rich enough logic (e.g., higher-order logic). Second, higher-order resolution is the central mechanism used to synthesize programs during proofs of their correctness. And third, synthesis proofs have a predictable form and can be partially or completely automated. In this talk I explain these ideas and illustrate the general methodology employed.

Induction and Decision Procedures

Deepak Kapur (joint work with M. Subramaniam and J. Giesl), University of New Mexico, USA

Decision procedures are considered vital for automating reasoning about system design and mobile computing, particularly hardware verification and proof-carrying code. Their power is, however, limited in the sense that most decision procedures (with Presburger arithmetic being an exception) do not incorporate induction schemes. Induction is considered central in proving properties of computation descriptions using loops, recursion and arbitrary data widths. We will show how decision procedures can be extended with induction schemes without losing automation. Using examples of Presburger arithmetic and decision procedures for the quantifier-free theory of free constructors, we will identify syntactic constraints on recursively defined functions and conjectures about them so that these conjectures can be decided automatically even though their proofs/disproofs need the use of induction schemes. For an invalid conjecture, methods for synthesizing the correctness predicate exactly characterizing the domain of values on which the conjecture is true, are developed. For an arbitrary quantifier-free conjecture, its correctness predicate can be synthesized from the correctness predicates for the equations appearing in the conjecture. For a conjecture in a subclass of formulas satisfying certain restrictions, the conjecture can be automatically decided by deciding the associated correctness predicate. This work

is done in the framework of the theorem proving approach of Rewrite Rule Laboratory (RRL), a rewrite-based induction prover. The cover set method for generating induction schemes based on terminating function definitions seems particularly effective for this approach.

Description Logics: Theory and Practice

Ian Horrocks, University of Manchester, Great Britain

There are now several promising application areas for description logics, e.g., reasoning about DB schemas and queries, and ontological engineering. Such applications generally require expressive logical languages, inevitably with high worst case complexities. However, experience has shown that by focusing on empirical tractability, practical systems are still possible.

As demonstrate by the FaCT system, empirical tractability can be achieved by combining a careful choice of logic with a highly optimised implementation. As far as the logic is concerned, a key feature of FaCT is the use of transitive roles instead of transitive closure. Although the addition of role inclusion axioms results in logics of the same complexity class (ExpTime), algorithms for the SH family of logics (SH = ALC with transitive roles and role inclusions) are simple and amenable to optimisation and behave well in realistic application.

The FaCT system includes a wide range of optimisations including lexical normalisation, simplification and encoding of concepts, absorption (simplification) of axioms, Davis-Putnam style semantic branching search, Dependency directed backtracking, caching and heuristics. When combined with the choice of algorithm, this provides acceptable performance in applications, e.g., when classifying large ontologies (knowledge bases).

The logic implemented in FaCT is SHIQ: SH with the addition of inverse roles and qualified number restrictions (graded modalities). Several applications require additional features, in particular datatypes (numbers, strings etc.), nominals (extensionally defined classes) and finite model reasoning. Extending the system to include datatypes should be straightforward, but nominals seem to be problematical: even for ALC with inverse roles, adding nominals makes reasoning NExpTime hard. However, an algorithm for SHQ (i.e., SHIQ without inverse roles) is relatively straightforward. Finite model reasoning leads to similar problems and is still an open problem (for SHIQ).

E-unification: Completeness, Decidability, Complexity

Christopher Lynch and Barbara Morawska, Clarkson University, Potsdam, NY, USA

We give a general goal directed procedure for solving E-unification. Then we give a more specialized procedure for linear equational theories and goals with no repeated variables. This procedure has the advantage that Eager Variable Elimination preserves completeness. We further restrict the equational theory so there are no repeated variables, and give an algorithm that decides E-unification in those theories in linear time. This result can be used to give a fast approximation algorithm for E-unification, which can quickly rule out goals that cannot be E-unifiable. Finally we define fintely closable equational theories. For a linear finitely closable equational theory, and a goal with no repeated variables, we give an algorithm that solves E-unification in PSPACE, and in NP or quadratic time for some special cases.

Formalising The Proof Theory of Display Calculi in Logical Frameworks

Jeremy E. Dawson and Rajeev Goré, Department of Computer Science, Faculty of Engineering and Information Technology, Australian National University, and
Automated Reasoning Group, Computer Sciences Laboratory, Research School of Information Science and Engineering, Australian National University

Logical frameworks are computer systems which allow a user to formalise mathematics using specially designed languages based upon mathematical logic and Church's theory of types. They can be used to derive programs from logical specifications, thereby guaranteeing the correctness of the resulting programs. They can also be used to formalise rigorous proofs about logical systems. We compare several methods of implementing the display (sequent) calculus dRA for relation algebra in the logical frameworks Isabelle and Twelf. We aim for an implementation enabling us to formalise, within the logical framework, proof-theoretic results such as the cut-elimination theorem for dRA, and any associated increase in proof length.

Keywords: logical frameworks, higher-order logics, proof systems for relation algebra, non-classical logics, automated deduction, display logic.

A Decision Procedure for Quantified Boolean Formulas

Reinhold Letz, TU München, Germany

The language of quantified Boolean formulas (QBFs) is gaining importance. While in complexity theory the central rôle of this language is obvious from the fact that it represents one of the natural paradigms for characterising

the complexity class PSPACE, in the last few years it has been recognised that QBFs are also suitable for a natural formulation or reformulation of many problems from planning, abduction, nonmonotonic reasoning, or from intuitionistic, terminological and modal logics. This has motivated the need for efficient decision procedures for QBFs. As a consequence, recently, a number of such procedures have been developed. However, when compared with the procedures available for propositional logic, these procedures are still in their infancy. Furthermore, even for the few procedures available, there is a tendency of *divergence*, in the sense that almost every procedure contains some special adhoc techniques that apply well to some examples, but may not be useful for a generally successful approach.

In this talk we identify some techniques that are very important if not essential for *any* powerful and robust QBF procedure. One of the paradigms is *intelligent backtracking*, which can be implemented quite efficiently in different variants. We also give experimental evidence that intelligent backtracking is of general importance for deciding quantified Boolean formulas. Another paradigm is *caching* which comes in two dual variants, caching of *lemmas* and caching of *models*. Unfortunately, the efficient integration of caching methods is much more difficult. However, there exist very small formulas which are intractable for the existing QBF procedures, but which become trivial when using caching methods. This suggests that such an integration might be indispensable.

Deductive Knowledge Management for Personalized Documents

Peter Baumgartner, Universität Gießen, Germany (On leave of absence from Universität Koblenz)

The work is about a real-world application of automated deduction. The application is the management of documents (such as mathematical textbooks) that are "'sliced"' into small units. A particular task ist to assemble a new document from such units in a selective way, based on the user's current interest.

It is argued that this task can be naturally expressed as a model computation task, provided that the full first-order clausal logic (beyond Horn logic) with some default negation principle is available.

A calculus for reasoning in this logic is developed in detail. It builds on our previously developed calculi for first-order classical reasoning (Hyper Tableaux, First-order Davis-Putnam-Logeman-Loveland procedure). Distinguished features of the new calculus are inferences directly at the first-order level (not via grounding) and absence of syntactical restrictions (such as range-restrictedness). The calculus is refutationally complete for the sub-case

of classical logic. For model generation, it computes finite representations of possibly infinite supported models whenever it terminates.

DEDUCTION and EDUCATION

Jörg Siekmann, Erika Melis, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, Germany

The first part of the talk presents an overview of a current paradigm change in automated reasoning from traditional automated theorem proving – for example, based on resolution – to proof planning research. We report about psychological experiments to test the role of different instructions for proving theorems in an educational context. The results provide first evidence for the superiority of teaching proof planning methods versus teaching of examples-only or textbook-like instructions. Finally, we present ActiveMath, a web-based learning environment that integrates several external systems useful for exploratory learning – among them the proof planner of the Omega system.

AutoBayes: A System for the Automatic Synthesis of Data Analysis Programs

Johann Schumann (joint work with Bernd Fischer), RIACS / NASA Ames, Moffett Field CA, USA

Although data analysis is an important scientific task, implementing a data analysis program is a difficult and time-consuming task, because it requires knowledge and experience in computational statistics and numerical mathematics.

In this talk, I present AutoBayes, a high-level generator for data analysis programs from statistical models. A statistic model specifies the properties for each problem variable (i.e., observation or model parameter) and its dependencies in a fully declarative way. From this model AutoBayes generates optimized and fully commented C/C++ code which can be linked dynamically into a MatLab or Octave environment.

Code is generated by schema-guided deductive synthesis. A schema consists of a code template and applicability conditions. Symbolic-algebraic computations augment schema-guided synthesis and thus can derive closed form solutions for many problems. AutoBayes has been tested on various text-book and statistical benchmark examples and is capable of synthesizing data analysis programs consisting of more than 1200 lines of optimized C++

code in roughly one minute. I also report on a recent small experiment on analyzing gamma ray burst data from the Compton Gamma Ray Observatory platform.

Process Verification using Dependency Pairs

Jürgen Giesl, RWTH Aachen, Germany

The dependency pair approach [1] is a technique which allows automated termination and innermost termination proofs for many term rewriting systems for which such proofs were not possible before. Apart from its use for termination analysis, we illustrate that the dependency pair approach is also very useful for process verification. To this end, we show how dependency pairs were applied at Ericsson Telecom in order to verify properties of a protocol for concurrent telecommunication processes.

In order to be applicable in this area, several refinements of the dependency pair technique had to be developed. We show how to extend the dependency pair approach to termination proofs of conditional rewrite systems. Moreover, we developed techniques for manipulating dependency pairs by narrowing, rewriting, and instantiations. These refinements are not only of use in the industrial application sketched in the talk, but they are generally applicable to arbitrary (conditional) rewrite systems. Thus, in this way dependency pairs can be used to prove termination of even more rewrite systems automatically.

This talk is based on joint work with Thomas Arts (Ericsson Telecom, Stockholm) [2, 3].

References

[1] T. Arts and J. Giesl, Termination of Term Rewriting Using Dependency Pairs, *Theoretical Computer Science*, 236:133-178, 2000.

[2] T. Arts and J. Giesl Applying Rewriting Techniques to the Verification of Erlang Processes, in *Proceedings of the Annual Conference of the European Association for Computer Science Logic (CSL '99)*, Madrid, Spain, Lecture Notes in Computer Science 1683, pages 96-110, 1999.

[3] J. Giesl and T. Arts, Verification of Erlang Processes by Dependency Pairs, *Applicable Algebra in Engineering, Communication and Computing*. To appear.

The KeY Approach: Integrating Object Oriented Design and Formal Verification

Reiner Hähnle, Universität Karlsruhe, Germany

The KeY project aims at bridging the gap between (a) object-oriented software engineering methods and tools and (b) deductive verification. A distinctive feature of our approach is the use of a commercial CASE tool enhanced with functionality for formal specification and deductive verification. To help users coming up with formal specifications, in the KeY system we provide design patterns that come complete with predefined OCL constraint schemata. The user needs not write formal specifications from scratch, but only to adapt and complete them.

A Program Logic for the Verification of Java Card Programs

Bernhard Beckert, Universität Karlsruhe, Germany

The work that is reported in this talk has been carried out as part of the KeY project (http:// i12www.ira.uka.de/~key). The goal of KeY is to enhance a commercial CASE tool with functionality for formal specification and deductive verification and, thus, to integrate formal methods into real-world software development processes. Accordingly, the design principles for the software verification component of the KeY system are:

- The programs that are verified should be written in a "real" object-oriented programming language (we decided to use JAVA CARD).
- The logical formalism should be as easy as possible to use for software developers (that do not have years of training in formal methods).

The ultimate goal of the KeY project is to facilitate and promote the use of formal verification as an integral part of the development process of JAVA CARD applications in an industrial context.

In this talk, I present a Dynamic Logic (a program logic that can be seen as an extension of Hoare logic) for JAVA CARD. It allows to express properties of JAVA CARD programs. The syntax and semantics of this logic is described. I present a calculus for this program logic that allows to reason about the properties of JAVA CARD programs and to verify them. The main ideas and principles of the calculus are described and some of its rules are presented. Finally, I give an example for the verification of a small JAVA CARD program.

Unified Framework for Simulation, Verification, and Testing of Formal Specifications

Olga Shumsky Matlin (joint work with L. J. Henschen), Northwestern University, Dept. of EE & CS, USA

Creating a formal design is considered an important first step in the system development cycle as doing so can eliminate significant design errors early on and thus lead to significant savings in later stages of the development. However, constructing formal designs is often omitted or the designs are essentially thrown away upon completion.

This work is concerned with building a unified framework for design, verification, and testing of system specifications. The goal is to make use of the formal specification beyond the design stage and to derive from the specifications as much benefit as possible at all stages of the development cycle. SDL, one of standardized formal description techniques, is chosen as an example formal language in which the original designs are created. Using ACL2, we build a framework that allows at earlier stages of the design to simulate and verify the SDL specification of the system and at later stages to use the specification to construct expected results for test scenarios and to automatically derive a test driver. At the current stage of the project we concentrate on building the mechanisms for simulation and verification of SDL specifications.

A decision Procedure for the Theory of Unary rpo

Paliath Narendran, State University of New York, USA

Representation, Administration, and Distribution of Mathematical Knowledge in the Internet Era

Michael Kohlhase, Carnegie Mellon University, USA (on leave from Saarland University, Germany)

In this talk I will survey the new opportunities for the dissemination of mathematical knowledge opening up by the Internet. It is plausible to assume that the way we publish mathematics will change radically in the next five years, and more generally that the way we do (conceive, develop, and verify) math.

Of course, this development is not restricted to mathematics itself, but will also affect other well-conceptualized and highly structured areas like formal methods or physics.

The trend towards high-quality Internet accessible math. is initiated by the availability of XML-based representation standards for mathematical formulae (MathML and OpenMath) together with corresponding browsers that allow to present formulae in LaTeX-quality, while retaining the flexibility of html.

The next step will inevitably follow: to represent the meaning of formulae, so that they can be transmitted to mathematical software systems like computer algebra systems, automated theorem provers, or proof presentation systems. The possibility of universal exchange of mathematical objects will radically change and de-centralize the way we work in mathematics, engineering and sciences.

In this talk, I want to discuss the infrastructure that is needed to conveniently and efficiently manipulate, visualize, and distribute mathematical knowledge on the basis of the OMDoc format (an extension of the OpenMath standard for the communication of mathematical objects) and the MBase system (a mathematical knowledge base).

Meta-Complexity Theorems

David McAllester (joint work with Harald Ganzinger), AT & T Labs Research, USA

We give two meta-complexity theorems. These are nontrivial theorems governing the run-time of bottom-up logic programs. We give a variety of examples of algorithms presented as logic programs and show how theorems governing run time allow a simple complexity analysis for these logic-program algorithms.

Synthesis of verifiably correct code for avionics

Michael Lowry, NASA, Ames Research Center, USA

Model Checking is easier than deduction, isn't it? Actually not!

Fabio Massacci, Università degli Studi di Siena, Italy

The appeal of model checking technology is based on its computational effectiveness: it has linear complexity in the size of the system. Yet, in practice, systems are given in high-level languages which are by far more "user-friendly" (and concise) that Kripke structures but for which the computational promises may not hold.

In talk I'll focus on the complexity of model checking when the Kripke structure is specified using SMV primitives (modules, bounded arithmetics, non-determinism, etc.).

I'll show that CTL and LTL model checking SMV-specifications is PSPACE-hard **in the size of the structure** by using a simple, linear-size reduction from QBF which uses only a constant size LTL/CTL specification.

Next we see that, for LTL, it is possible to encode synchronous SMV specifications allowing generalized set-expressions, bounded arithmetic, etc. into LTL formulae. Thus, LTL model checking using a practical language is as hard as LTL theorem proving.

One of the aim of this QBF2SMV and SMV2LTL reductions is to use the generation of QBF problems as a source for controlled generation of benchmarks to test model-checking systems.

Joint work with Francesco Donini, Paolo Liberatore, Marco Schaerf

Description Logics and Arithmetics

Hans-Jürgen Ohlbach, Ludwig-Maximilians-Universität München, Germany

In the presentation, mathematical programming and atomic decomposition was introduced as the basic modal (T-Box) inference techniques for a large class of modal and description logics. The class of description logics suitable for the proposed methods is strong on the arithmetical side. In particular there may be complex arithmetical conditions on sets of accessible worlds (role fillers).

The atomic decomposition technique can deal with set constructors for modal parameters (role terms) and parameter (role) hierarchies specified in full propositional logic. Besides the standard modal operators, a number of other constructors can be added in a relatively straightforward way. Examples are graded modalities (qualified number restrictions) and also generalized quantifiers like 'most', 'n%', 'more' and 'many'. Details can be found in [1, 2].

References

[1] Hans Jürgen Ohlbach and Jana Koehler. How to extend a formal system with a boolean algebra component. In W. Bibel P.H. Schmidt, editor, *Automated Deduction. A Basis for Applications*, volume III, pages 57–75. Kluwer Academic Publishers, 1998.

[2] Hans Jürgen Ohlbach and Jana Koehler. Modal logics, description logics and arithmetic reasoning. *Journal of Aritificial Intelligence*, (109):1–31, 1999.

New Ideas on Term Indexing

Robert Nieuwenhuis, Tech. Univ. Catalonia, Barcelona, Spain

Indexing data structures are well-known to be crucial for the efficiency of the current state-of-the-art theorem provers. In this talk we first recall discrimination trees, which are like tries where terms are seen as strings and common prefixes are shared, and substitution trees, where terms keep their tree structure and all common contexts can be shared.

Second, we describe a new indexing data structure, called context trees, where, by means of a limited kind of context variables, also common subterms can be shared, even if they occur below different function symbols.

Third, we discuss our new methodology for objectively evaluating indexing data structures. Experiments using this methodology for matching show that our preliminary implementation is already competitive with tightly coded current state-of-the-art implementations of the other main techniques. In particular space consumption of context trees is substantially less than for other index structures.

Finally, we describe current work on intelligent backtracking for retrievals on context trees, and the possibility of using flatterm queries in compiled context trees.

Polynomial-time Cost-based Hypothetical Reasoning: Propositional and Predicate Logic Cases

Mitsuru Ishizuka, Yutaka Matsuo and Helmut Prendinger, University of Tokyo, Japan

Hypothetical reasoning (or abduction) is an important framework for knowledge-based systems because it is theoretically founded and useful for many practical problems. Since the inference time of hypothetical reasoning grows exponentially with respect to problem size, its efficiency becomes the most crucial problem when applied to practical problems.

As the first topic, we present an efficient method called SL (slide-down and lift-up) method, which uses a linear programming technique, namely the simplex method, for determining an initial search point, and a non-linear programming technique for efficiently finding a near-optimal 0-1 solution. To escape from trapping into local optima, a local handler is incorporated which systematically fixes a set of variables to locally consistent values when a locally optimal point is detected. The SL method, whose behavior is very comprehensive for humans, can find a near-optimal solution for propositional cost-based hypothetical reasoning problems in polynomial time with respect to problem size.

During the above research, we have noticed that there are two major ways of transforming propositional clauses into numerical constraints. One transforms the clauses into linear inequalities, while the other transforms them into non-linear equalities; these two transformations reveal different characteristics. As the second topic, we show a method of integrating these two transformations by using the augmented Lagrangian method to effectively find better near-optimal solutions in propositional cost-based hypothetical reasoning.

As described above, these exist some efficient reasoning mechanisms in propositional-level hypothetical reasoning. However, these methods are not directly applicable to predicate-logic-version hypothetical reasoning. As the third topic, we present our approach to this problem. Specially, we first perform a knowledge-level transformation followed by instantiation of the first-order clauses into propositional ones, and then apply an efficient propositional-level reasoning mechanism to find a near-optimal solution. The knowledge-level transformation here is based on equivalency-preserving unfold/definition/fold transformations, and allows to obtain a compact propositional representation with significantly less clauses than a simple-minded instantiation of the original first-order theory.

One important message of this research is the effectiveness of using search mechanisms in continuous-value space rather than those in binary space to compute near-optimal solutions efficiently. In the continuous-value space, we can obtain the guiding information of search everywhere in the space as a gradient value of the defined objective function, whereas this is not true in binary space.

The Calculus of Structures

Kai Brünnler, Paola Bruscoli, Alessio Guglielmi, Steffen Hölldobler, Lutz Straßburger, TU Dresden, Germany

Our main goal is the development of appropriate logics and calculi for distributed computing such that we can solve existential problems like planning problems as well as universal ones like verifying deadlock freeness. Distributed computing is characterized by parallel and interacting processes as well as by sequential processes. It is well known that parallel and interacting processes can be modelled using two binary operators, which are associative, commutative and admit a unit element, but which are not idempotent. But how can sequential processes be modelled? Clearly, we need an associative, but non-commutative operator. It seems to be quite difficult if not impossible to develop a calculus with two commutative and a single non-commutative operator using a standard methodology like natural deduction

calculi, sequent-style calculi or the connection method. We have developed a new methodology based on so-called structures, which allowed us to specify appropriate calculi for our tasks. The basic calculus of structures enjoyes some nice properties like atomic interaction and atomic cut rules, a perfect top-down symmetry as well as cut elimination. We also tested our methodology by specifying calculi of structures for the multiplicative fragment of linear logic including exponentials as well as propositional logic. In both cases the calculi enjoyed the same nice properties mentioned above. It also appears that in calculi of structures derivations can be decomposed in some structured way allowing for a modulatization of cut elimination proofs.

Superposition and Chaining for Totally Ordered Divisible Abelian Groups

Uwe Waldmann, MPI für Informatik, Saarbrücken, Germany

We present a calculus for first-order theorem proving in the presence of the axioms of totally ordered divisible abelian groups. The calculus extends previous superposition or chaining calculi for divisible torsion-free abelian groups and dense total orderings without endpoints. As its predecessors, it is refutationally complete and requires neither explicit inferences with the theory axioms nor variable overlaps. It offers thus an efficient way of treating equalities and inequalities between additive terms over, e. g., the rational numbers within a first-order theorem prover.

The calculus splits into two parts: The first one is a base calculus that works on clauses in which all variables are shielded (i. e., every variable occurs at least once below a free function symbol). This calculus has the property that saturated sets of clauses are unsatisfiable if and only if they contain the empty clause, but as its rules may produce clauses with unshielded variables, it can not be used to effectively saturate a given set of clauses. The second part of the calculus is a variable elimination algorithm for totally ordered divisible abelian groups, that makes it possible to get rid of unshielded variables, and thus renders the base calculus effective.

Complexity of Logic Programming Approaches to Model Checking

Terrence Swift, SUNY at Stony Brook, USA

Annotated Reasoning

Dieter Hutter, Deutsches Forschungszentrum für Künstliche Intelligenz, Saarbrücken, Germany

The application of deduction in various domains resulted in a variety of different techniques to guide the proof search. Many of these techniques incorporate additional knowledge to restrict or select possible proof steps. In the past a large variety of approaches have been presented on how additional knowledge can improve proof search. In automated theorem the proofs that specific calculus rules can be permuted are used to cut off redundant branches of the search tree. In basic ordered paramodulation and basic superposition, for instance, paramodulation inferences are forbidden at terms introduced by substitutions from previous inference steps. To implement such a strategy, we have to maintain such knowledge for each individual subterm. In tactic based theorem proving, the problem arises to monitor parts of the problem during the proof search. Such focus mechanisms have been developed and hardwired into several calculi. Proving the invariance of a state-transition system suggests the use of a specialized proof methodology which knows about the constituents (e.g. action descriptions or instances of the invariants for various states) of the arising proof obligations and their treatments inside the proof. Rippling is also a successful example of how to use domain knowledge to guide a theorem prover. The application oriented heuristic, that the induction hypothesis should be used when proving the induction step, is translated into a syntactical requirement that in each proof step, the hypothesis should be embedded in the conclusion. In analogical reasoning, a given proof (the so-called source proof) is abstracted to serve as a proof sketch for other, so-called target problems. Typically, additional information about the source proof (besides the usual proof tree) is required to compute an abstract proof sketch for a related target problem.

In all these examples there is a need for encoding and maintaining additional knowledge, which is used to guide the proof search. While all the presented approaches developed their own individual solutions to this problem, i.e. by introducing specialized calculi, we aim at a uniform representation and maintenance of such information in a logical (and thus formal) way. We provide a methodology to augment a logic calculus by a generic mechanism to maintain such strategic knowledge. It is formally encoded into a term language and stored as annotations at the individual parts of the logic formulas. Calculus rules and necessary basic algorithms like unification are reformulated to cope with such logic annotations. Annotations are used to restrict possible deductions as the unification of two annotated terms has also to identify corresponding annotations. In contrast to labeled deduction systems, annotations are used only to maintain strategic knowledge. Each

deduction in the annotated calculus corresponds to a deduction in the original (not annotated) calculus. We obtain such a deduction simply by stripping off all annotations. Regardless of how we use annotations to encode strategic knowledge, the soundness of the underlying derivation is guaranteed. Thus an annotated calculus suggests itself as a secure kernel of a tactic based theorem prover. Annotations provide a flexible, generic mechanism to maintain strategic knowledge during proof search without jeopardizing the soundness of the underlying calculus. To instantiate this generic approach for her individual needs, a user has to provide rules how to annotate the initial problem and the given calculus rules. After these initial setting, the strategic knowledge encoded into annotations is automatically maintained by the annotated calculus and by annotated unification in particular.

Incremental Closure of Free Variable Tableaux

Martin Giese, Universität Karlsruhe, Germany

A technique for automated theorem proving with free variable tableaux is presented, that does not require backtracking.

Most existing automated proof procedures using free variable tableaux require iterative deepening and backtracking over applied instantiations to guarantee completeness. If the correct instantiation is hard to find, this can lead to a significant amount of duplicated work. Incremental Closure is a way of organizing the search for closing instantiations that avoids this inefficiency. Instead of globally applying substitutions to close branches, an instantiation that closes all branches simultaneously is incrementally calculated.

Deduction, Exploration and Abstraction

Natarajan Shankar, SRI International Computer Science Laboratory, USA

Induction as Deduction Modulo

Eric Deplagne, Claude Kirchner, LORIA - INRIA, France

Inductive proofs can be built either explicitly by making use of an induction principle or implicitly by using the so-called induction by rewriting and inductionless induction methods. When mechanizing proof construction, explicit induction is used in proof assistants and implicit induction is used in automated theorem provers. The two approaches are clearly complementary

but up to now there was no framework able to encompass and to understand uniformly the two methods. In this paper, we propose such an approach based on the general notion of deduction modulo. We extend slightly the original version of the deduction modulo framework and we provide modularity properties for it. We show how this applies to a uniform understanding of the so called induction by rewriting and inductionless induction methods and how this relates directly to the general use of an induction principle.

Symbolic Evaluation and the Use of Lemmas in the ✓eriFun System

Christoph Walther, Technische Universität Darmstadt, Germany

Symbolic evaluation is a technique to prove statements about programs, or to simplify them at least. The phrase "symbolic" stresses the fact that expressions to be evaluated usually contain uninterpreted symbols, like program variables which are not bound at verification time or calls of procedures which are not defined within the program. The term "evaluation" refers to the control regime of the theorem prover which mimics an interpreter of a programming language, e.g. when conditionals or procedure calls are encountered. We investigate how to integrate the use of lemmas, which have been proven elsewhere, into a theorem prover based on symbolic evaluation. This theorem prover then is used in the ✓eriFun System, an interactive verifier for a simple functional language.

Stratified Context Unification is in PSPACE

Manfred Schmidt-Schauss, Universität Frankfurt, Germany

Context unification is a variant of second order unification and also a generalization of string unification. Currently it is not known whether context unification is decidable. A decidable specialization of context unification is stratified context unification, which is equivalent to satisfiability of one-step rewrite constraints.

This paper contains an optimization of the decision algorithm, which shows that stratified context unification can be done in polynomial space.

Modal Normal Form Tableaux

Ulrich Furbach and Rajeev Goré, Universität Koblenz, Germany

Tableau calculi for the logics K, T and $K4$ are introduced. These calculi are based on a clausal normal form of modal logics, which has been used until now mainly for the definition of modal resolution calculi. We present the calculi such that the close relationship to clause normal form tableau procedures for the classical case become obvious. For this, a bottom-up tableau method, based on Hyper-Tableaux or SATCHMO is given. By this, the necessary information about the worlds visited during a tableau construction is implicitly coded in the branch-structure and can be used by the inference rules.

Meta-Inference in Krom-Horn Logic, an Application to the decision problem

Alexander Leitsch (joint work with Gernot Salzer and Andrei Voronkov),
TU Vienna, Austria

In computational inference systems deductions frequently show "regular" and iterative patterns. Though, in some cases, these patterns are easily describable in the mathematical meta-language, pure first-order inference systems mostly are incapable to handle them. This observation and the need to obtain shorter proofs and better termination behavior in first-order inference systems led to the investigation of meta-terms and deductive generalization around 1990. We use one of these meta-term concepts, namely R-terms (Salzer 1989, 1992), in the context of resolution inference in Krom-Horn logic; this logic allows for a particularly simple handling of cycle generation (= generation of a meta-clause C_* describing the clause powers C^n of a clause C). We define a deduction operator R_{Ms} which is based on hyperresolution, rule resolution, cycle generation and forward subsumption. It is shown that R_{Ms} terminates on an extension KH_1 of the class $\forall\exists\forall\text{Horn} \cap \text{Krom}$. This gives the first proof-theoretical decision procedure for this class, which so far resisted all attempts to decide it via a resolution refinement. R_{Ms} does not only decide the class, but yields (meta-term representations of) minimal Herbrand models in case of satisfiability. The operator R_{Ms} is not just a tool to decide the class KH_1, but also a simple and powerful calculus which might turn out useful to "real" automated deduction as well.