Dagstuhl Report

Deduction

Wolfgang Bibel
Technische Hochschule Darmstadt
Fachbereich Informatik
Alexanderstr. 10
64283 Darmstadt
Germany
bibel@intellektik.informatik.th-darmstadt.de

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

Ryuzo Hasegawa
Department of Electronics
Kyushu University
10-1 Hakozaki 6-chome
Higashi-ku, Fukuoka 812
Japan
hasegawa@ele.kyushu-u.ac.jp

Mark Stickel
Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park, California 94025
USA
Stickel@AI.SRI.COM

This report[1] covers the seminar on *Deduction*, held at Dagstuhl, Germany during February 24–28, 1997. This seminar was organized by W. Bibel (Darmstadt, Germany), U. Furbach (Koblenz, Germnay), R. Hasegawa (Kyushu, Japan) and M. Stickel (SRI, 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 41 talks, a panel, and a discussion how system implementors could cooperate.

[1]Compiled by Peter Baumgartner, Universität Koblenz.

1 A Note from the Organizers

The Dagstuhl Seminar on Deduction, succeeding the ones in 1993 and in 1995, was convened to give international researchers on deduction the opportunity to meet and discuss techniques, applications and research directions for deduction. The seminar also provided a forum for results obtained in the German focus project (DFG Schwerpunkprogramm) on deduction.

Throughout this seminar it turned out that logic is not only an essential formalism for computer science and artificial intelligence. Moreover there is a driving force towards using logic and automated deduction within various applications and other domains of computer science. There were talks ranging from more theoretical questions on deduction systems, to applications like planning, logic programming, knowledge representation and to verification of hard- and software.

With the excellent computing facilities available at Dagstuhl it also was possible to have a number of very impressive system demonstrations.

A panel on "How to get Automated Deduction out of its Corner" was organised during this seminar by Wolfgang Bibel. The discussion resulted in an agreement that there are realistic examples of applying automated deduction and its methods within various application areas.

Ulrich Furbach

2 Abstracts of the Talks

Jürgen Avenhaus, Univ. Kaiserslautern

Inductive Theorem Proving with Partial Functions

In this talk we are interested in an algebraic specification language that allows for sufficient expessiveness, admits a well-defined semantics, and allows for formal proofs. To that end we study specifications over built-in algebras. To keep things simple, we consider built-in algebras only that are given as the initial model of a Horn clause specification. On top of this Horn clause specification new operators are (partially) defined by positive/negative conditional equations.
In the first part of the talk we define model-theoretic and operational semantics for such a hierarchical specification. We show that these semantics coincide, provided some restrictions are met. We associate a distinguished algebra $\mathcal{A}_{spec}$ to a hierachical specification *spec*. This algebra is initial in the

class of all models of *spec*. As a consequence, an equation $s = t$ is valid in $\mathcal{A}_{spec}$ iff it is valid in all term-generated models of *spec*. For this reason we define inductive validity as validity in this model $\mathcal{A}_{spec}$.
In the second part of the talk we present a proof calculus for inductive validity. Traditional inductive theorem proving restricts to the case where the specification (a) is terminating and (b) defines all new operators totally. Our proof calculus does not need these restrictions.
We give some examples to show how concrete proofs are carried out.

Franz Baader, RWTH Aachen

Combination of Compatible Reduction Orderings that are Total on Ground Terms

Reduction orderings that are compatible with an equational theory E and total on (the E-equivalence classes of) ground terms play an important rôle in automated deduction. This paper presents a general approach for combining such orderings: it shows how E_1-compatible reduction orderings total on Σ_1-ground terms and E_2-compatible reduction orderings total on Σ_2-ground terms can be used to construct an $(E_1 \cup E_2)$-compatible reduction ordering total on $(\Sigma_1 \cup \Sigma_2)$-ground terms, provided that the signatures are disjoint and some other (rather weak) restrictions are satisfied.

This work was motivated by the observation that it is often easier to construct appropriate orderings for "small" signatures and theories separately, rather than directly for their union. Well-known examples for this phenomenon are theories axiomatizing several associative-commutative function symbols.

Michael Beeson, San Jose State University

Automatic Generation of Epsilon-Delta Proofs

A computer program has been able to generate a proof of the uniform continuity of $f(x) = x^3$. This is the first time a program has been able to prove the continuity of a non-linear function.

The methods used are intended to exemplify techiques for a fruitful integration of theorem-proving and symbolic computation. A "backwards Gentzen" theorem prover is used, in which at any state, either computation or logical rules can be used. If computation is used, the computational operations may generate new assumptions (new formulae in the antecedent). These may include metavariables to be instantiated to specific terms later.

Control of the inference procedure involves symbolic computation also at the metalevel. For example, a procedure called FactorBounding is used to decide that when confronted with the goal

$$|x - y||x^2 + xy + y^2| < \epsilon$$

we should try to bound the second factor.

The success of this work rests in large part on the logically-correct symbolic computation code in my software Mathpert, which was developed to help students learn algebra, trigonometry, and calculus. To achieve the result described above, this code was combined with more or less standard proof search techniques.

C. Beierle, G. Meyer, FernUniversität Hagen

Analysis and Verification of Type-Annotated Logic Programs

The technique of annotating programs can be used for the analysis and the verification of programs. The kind of annotations most widely used are type annotations. We investigate the use of types in logic programming where types are used not only for static analysis, but also as dynamic constraints in the spirit of constraint logic programming. Furthermore, types can be interpreted as describing approximations of the success set of a program. We analyze characteristics and shortcomings of different approaches to types in logic programming and propose a notion of type annotations that allows static program analysis to detect typing errors as well as useless clauses.

Wolfgang Bibel, TH Darmstadt

Reasoning about Action in CL

A resource-sensitive logic (termed linear connection method in Bibel 1986 and closely related to BCK) is embedded in classical first-order logic as a sublogic to yield a structurally simple and rather natural new computational logic CL for reasoning about theories as well as actions. Deduction in both parts of CL is nearly identical which supports the ease and efficiency of implementation and reflects reasoning in natural language where the difference is blurred anyway. CL is compared with linear logic, the STRIPS formalism, the fluent calculus, and the situation calculus. It is illustrated how planning, temporal projection, postdiction, ramification, qualification, indeterminism, synchronization, and diagnosis may be done in CL using a classical theorem prover like SETHEO or KOMET.

Jim Cunningham, Imperial College

Multi-modal Deduction: a mechanism for realising rationality

Multi-modal logic provides a framework for theories of rationality, where distinct modalities are used to separate what is Sensed ($\mathcal{S}$) to be the case from what is Believed ($\mathcal{B}$), Intended ($\mathcal{W}$), Wanted ($\mathcal{W}$), brought about by an action a ($\mathcal{E}_a$), or directly Done by a ($\mathcal{D}_a$). For example, the axiom schema: $\mathcal{S}\alpha \rightarrow \mathcal{B}\alpha$ captures the naive notion that what is sensed is believed. A slightly improved theory with updating ($\mathcal{N}$ext) and unorganised memory ($\mathcal{P}$reviously) might have the schema pair: $\mathcal{S}\alpha \rightarrow \mathcal{N}\mathcal{B}\alpha$, $\mathcal{S}\neg\alpha \wedge \mathcal{B}\alpha \rightarrow \mathcal{N}\mathcal{B}\mathcal{P}\alpha$, and so on. A similar first attempt to capture goal directed action can yield the problematic rapacious schema: $\mathcal{W}\alpha \wedge \mathcal{B}\mathcal{E}_a\alpha \rightarrow \mathcal{D}_a$. In a well known paper Cohen & Leveque provide improved multi-modal theories which demonstrate the expressive capability of multi-modal logics for AI. To describe continuous change in AI scenarios, Halpern & Shoham developed a multi-modal interval logic. In recent work, Leith & myself have shown that tense and aspect in natural language can also be represented in this logic.

To realise an abstract multi-modal logic for theories of rationality we need a programme for multi-modal deduction. One direct way is to build a network of inference engines, each working concurrently in a different modality, with appropriate communication in the common language of classical propositions. We may, for example, split each instance of a the naive sensing-is-believing schema, say $\mathcal{S}c \rightarrow \mathcal{B}c$, by introducing a communicated proposition X_c, so that $\mathcal{S}c \rightarrow X_c$, and $X_c \rightarrow \mathcal{B}c$. Of course, this requires an interpolation lemma, but it indicates a direction for multi-modal deduction which lies behind my work with Pitt in distributed modal KE Theorem Proving systems. However, for logical deduction to provide rationality we must also get away from the idea that deduction is just theorem proving. This is a tradition of Mathematical logic rather than of Philosophical logic. In AI we need to use deduction for other forms of rational enquiry too, like discovering interesting consequents, abducing causes and planning actions. (To illustrate this tradition, Lewis Carroll's School-Boy's problem is presented as a challenge).

Bernd Ingo Dahn, Univ. Berlin

Proof Presentation - Why and How / The Robbins Case

In late 1996 William McCune proved with the automated theorem provers EQP and Otter that each Robbins algebra satisfies Winker's second condition. Hence each Robbins algebra is Boolean. This solved a problem which was open for more than 60 years.

We demonstrate the automatic generation of a human readable presentation of this proof by the ILF system. It translates Otter proofs into block structured proofs. These proofs can be easily presented in any language with SGML document type definition. They can be also combined consistently with proofs from other systems.

The block structured proof is then transformed automaticaly. This eliminates trivial details and generates additional explanations. Renaming and type setting declarations introduce abbreviations for important terms. These procedures can be varied in order to present the proof from several points of view.

ILF has also a tool to visualize the logical structure of the proof. This supports the identification of important proof steps.

The analysis of different proof presentations made it possible to reveal the mathematical idea of the solution of Robbins problem by EQP and Otter. Thus the depth of terms in the proof could be reduced from 14 to 4.

Jörg Denzinger, Univ. Kaiserslautern

Learning Search Control for Automated Deduction

We present the general problems that arise, if one wants to enhance an automated prover with learning capabilities and we sketch some general solutions to these problems. Furthermore, we present a concept for an automated theorem prover that employs a search control based on ideas from several areas of artificial intelligence (AI). The combination of case-based reasoning, several similarity concepts, a cooperation concept of distributed AI and reactive planning enables a system using our concept to learn form previous successful proof attempts. In a kind of bootstrapping process easy problems are used to solve more and more complicated ones.

We provide case studies from two domains of interest in pure equational theorem proving taken from the TPTP library. These case studies show that an instantiation of our architecture achieves a high grade of automation and outperforms state-of-the-art conventional theorem provers.

Melvin Fitting, City University of New York

First-Order Modal Logics

The talk consisted of a brief sketch of the complexities that must be addressed in a full treatment of first-order modal logic(s). Specifically, the following items were discussed: Existence presuppositions; that is, what assumptions can be made concerning quantifier domains. Rigid vs non-rigid

designators. Syntax issues. Skolemization. Herbrand's theorem. Equality and problems related to it, such as Frege's morning star/evening star problem. The possibility of non-designation. Definite descriptions.

The presentation was essentially semantic. Tableau systems exist that can deal with the issues raised. Implementation is a different matter, and is likely to prove quite complex. It is left as an exercise.

Ulrich Furbach, Universität Koblenz

Semantically Guided Theorem Proving for Diagnosis Applications

In this paper we demonstrate how general purpose automated theorem proving techniques can be used to solve realistic model-based diagnosis problems. For this we modify a model generating tableau calculus such that a model of a correctly behaving device can be used to guide the search for minimal n-fault diagnoses. Our experiments show that our general approach is competitive with specialized diagnosis systems.

Pascal Gribomont, University of Liège

Connection-based invariant verification for concurrent programs : Validation of Boolean conditions.

The correctness problem for hardware and software systems can often be reduced to the validity problem for logical formulas. However, the size of the logical formulas to be validated grows faster than the size of the system under investigation, and the complexity of the validation procedure makes this approach practically intractable for large programs. We introduce a strategy for dealing with this problem in the propositional case, corresponding e.g. to digital circuits and concurrent synchronization algorithms. Efficiently computable criteria are used to assess the mutual relevance of formulas and subformulas. They are based on the notions of interpolation and polarity, and allow to detect and discard provably irrelevant parts of Boolean verification conditions. These criteria lead to a simplification and validation method, whose efficiency is investigated both theoretically and practically.

Reiner Hähnle, Univ. Karlsruhe

Semantic Tableaux with Selection Functions

Recently, several different sound and complete tableau calculi were introduced, all sharing the idea to use a selection function and so-called restart clauses: A-ordered tableaux, tableaux with selection function, and strict restart model elimination. We present two new sound and complete abstract tableau calculi which generalize these on the ground level. This makes differences and similarities between the calculi clearer and, in addition, gives insight into how properties of the calculi can be transferred among them. In particular, a precise borderline separating proof confluent from non-proof confluent variants is exhibited.

Masami Hagiya and Masayuki Fujita, University of Tokio and Mitsubishi Research Institute

Specification Verification by Hybrid Reasoning System in CafeOBJ: An Algebraic Specification Language

CafeOBJ is an algebraic specification language belonging to the OBJ family of languages originated by Joseph Goguen. Among its new features are (1) introduction of rules in the sense of rewriting logic and (2) introduction of hidden sorts for describing behavioral specifications. Both features are intended to support object-orientation. This talk is a brief overview of an ongoing project developing an environment for specification and verification in CafeOBJ. The project is led by Kokichi Futatsugi and funded by IPA (The Information-technology Promotion Agency, Japan), beginning in 1996 and ending in 1998. It consists of three subsystems: (1) language processing, (2) verification and (3) editing and retrieval. As for language processing, a term writing abstract machine and a compiler to abstract code are being developed. Verification in CafeOBJ will be done in an hybrid environment consisting of various provers. A network-wide environment based on an extension of HTML is expected to support editing and retrieval of modules.

The key points in verification under the environments are: (1) No new language for describing proofs is introduced. Theorems and lemmas are written as modules or views in CafeOBJ. (2) The extension of HTML (called Forsdonnet) supports tags that represent constraints relating pieces of information on a document. Modules in CafeOBJ as well as verification traces are described in Forsdonnet. Provers are considered as constraint solvers and prover outputs are also inserted and maintained in Forsdonnet.

F. von Henke, A. Dold, H. Pfeifer, H. Rueß, Universität Ulm

Deductive Program Verification Revisited: Compiler Correctness

Compiler correctness is a problem that has been investigated for at least 30 years. In this talk we give a survey of an effort to build up the mathematical basis for specifying and verifying compiler correctness in a completely formal and mechanizable framework.

Since modeling the semantics of loops typically requires fixed points, the starting point of this development is a formal treatment of the relevant parts of domain theory and fixed point theory. Based on this, denotational semantics for elementary constructs (expressions, sequence, conditional, while loop) is developed; other forms of semantics including weakest preconditions, Hoare semantics, operational semantics in the SOS style, and appropriate algebraic laws as used in refinement calculi are derived. This development is carried out using the PVS system, thus leading to a collection of (typically parameterized) theories suitable for inclusion in a PVS library.

The formal specification of compilation for the constructs listed above proceeds in steps: each type of constructs is dealt with separately, thus isolating the essential aspects of the compilation step. The final translation of control constructs into machine code involving jumps requires two kinds of induction: fixed-point induction for handling loops, and induction over the structure of (source) programs.

Alain Heuerding, IAM, University of Bern

LWB - The Logics Workbench

The Logics Workbench is an interactive system which gives a survey of a broad variety of propositional logics. One application is as a supporting tool for teaching activities at various levels, but it also offers efficient algorithms for experts. The carefully designed human interface comes with a powerful information system. Therefore it should be easy also for non-specialists to use the LWB with profit. For more information:

```
lwb@iam.unibe.ch
http://lwbwww.unibe.ch:8080/LWBinfo.html
```

Jörg Hudelmaier, Univ. Tübingen

Construction of Kripkean Countermodels for Intuitionistically Unprovable Sequents

There is a well known direct relationship between finitely failing LK-derivations of a propositional formula and its Boolean countermodels. For LJ, however,

there is no similar relationship with Kripkean semantics. Therefore we introduce a new calculus LJn obtained from LJ by adding an additional premiss to its I→-rule. The form of this new premiss is the same as the form of the (single) I→-premiss of LK. Then we show that any unprovable formula of LJ has a finitely failing LJn-derivation. Thus for LJn the required relationship with Kripkean semantics holds.

Deepak Kapur, State University of New York

Mechanical Verification of Arithmetic Circuits

The use of a rewrite-based, induction theorem prover, *Rewrite Rule Laboratory RRL*, is discussed for gate-level verifying arithmetic circuits. It will be shown that the induction scheme generation heuristic in RRL based on the cover set method, tight integration of decision procedures with rewriting, and intermediate lemma generation heuristics can help in finding verification proofs of arithmetic circuits with minimal user guidance. Particularly, *RRL* has been used to automatically verify that ripple carry, carry save as well as a more sophisticated carry lookahead adder perform addition on numbers represented by bit vectors of any length. Correctness of multiplier circuits is established generically and parametrically by showing that the family of multipliers circuits including linear multiplier, Wallace multiplier, 7-3 multiplier as well as Dadda multipler can all be done in the same way. Intermediate lemmas needed for verifying that the multipliers perform multiplication on numbers represented by bit vectors of any length can be mechanically generated from the circuit structure, repetitive use of carry-save adders and the fact that bit vectors represent numbers.

Finally, it is be shown that the invariant properties of an SRT division circuit described in Clarke et al's CAV'96 paper can also be established automatically by *RRL* using its linear arithmetic procedure, rewriting and case analysis mechanisms. This shows that verification of such a circuit does not need sophisticated capabilities of computer algebra systems such as Maple and Mathematica. This proof turned out to be straightforward, a pleasant surprise to us.

Michael Kohlhase, Univ. Saarbrücken

Deduction Techniques for Natural Language Understanding

The talk emphasizes the opportunity of using deduction methods in natural language understanding.

We start out by explaining some some of the usages of reasoning, such as common ground maintenance, discourse structure and coherence and finally during the reconstruction of underspecified material.

The technical part of the talk, takes a closer look at higher-order unification techniques for ellipsis reconstruction and corrections. In the first case we show how higher-order colored unification, a variant of HOU developed for inductive theorem proving (rippling with meta-variables) can be used as a general interface to specify extra-semantical linguistic constraints on the solutions and thereby restrict the search spaces.

In the case of corrections, we need an extension of HOU that takes logical equivalences into account to cope with the discourse relation of parallelity in the presence of background. This involves the use of a theorem prover inside higher-order unification.

The talk closes with the challenge to the community to take a closer look at the linguistic applicatio

Wolfgang Küchlin, Universität Tübingen (joint work with Reinhard Bündgen and Manfred Göbel)

Parallel Term Rewriting on a Hierarchical Multiprocessor

We report on a parallel implementation of an unfailing term completion procedure on a network of multiprocessor workstations. Our parallelization concept uses a simplified form of the Teamwork approach by Avenhaus and Denzinger to combine several instances of our parallel term-rewriting system PaReDuX. It integrates distributed search parallelism on the network, based on a master–slave approach, with the parallel execution of each strategy on a multi-processor. Both levels of parallelism are realized by a uniform fork–join paradigm using multi-threading. In many of our examples we are able to combine the benefits of distributed and shared-memory approaches for superior overall speed-ups.

A full paper on this subject has been published in Proc. DISCO'96, Springer LNCS 1128, pp. 183–194.

Alexander Leitsch, TU Vienna (joint work with Christian Fermüller)

Automated Model Building: Representation, Construction and Evaluation

Hyperresolution is presented as decision procedure, model building procedure and evaluation procedure on (classes of) clause logic. If hyperresolution

terminates on a set of clauses (e.g. on clause sets belonging to the decision classes PVD and OCC1N) without deriving a contradiction then the resulting satisfiable set of clauses $\mathcal{C}$ can be used as raw material for automated model building. If $\mathcal{C}$ is a fixed point under hyperresolution + replacement and the positive clauses are decomposed (i.e. different literals do not share variables) then there is a backtracking-free method for producing an atomic model representation; this method is based on literal selection and deductive closure. Moreover hyperresolution can be used as an evaluation algorithm for arbitrary clauses over atomic representations; this gives the first evaluation algorithm over infinite models. A method (using signature-based subsumption) is defined to decide the equivalence problem of different atomic representations. Thus we have shown that hyperresolution, the oldest refinement of resolution, is much more than just a method to derive $\square$. Finally extensions of model representation formalisms and generalizations to equational clause logic are indicated.

Neil V. Murray, SUNY Albany

A Remark on Proving Completeness

Completeness proofs that generalize the Anderson-Bledsoe excess literal argument are developed for calculi other than resolution. A simple proof of the completeness of regular, connected tableaux for formulas in conjunctive normal form (CNF) is presented. These techniques also provide completeness results for some inference mechanisms that do not rely on clause form. In particular, the completeness of regular, connected tableaux for formulas in negation normal form (NNF), and the completeness of NC-resolution for NNF formulas under a linear restriction can be proved.

Joachim Niehren, Univ. des Saarlandes

On Equality Up-to Constraints over Finite Trees, Context Unification, and One-Step Rewriting

We introduce equality up-to constraints over finite trees and investigate their expressiveness. Equality up-to constraints subsume equality constraints, subtree constraints, and one-step rewriting constraints. We establish a close correspondence between equality up-to constraints over finite trees and context unification. Context unification subsumes string unification and is subsumed by linear second-order unification. We obtain the following three new results. The satisfiability problem of equality up-to constraints is equivalent to context unification, which is an open problem. The positive existential fragment

of the theory of one-step rewriting is decidable. The $\exists^*\forall^*\exists^*$ fragment of the theory of context unification is undecidable.

Keywords tree constraints, subtree relation, string unification, context unification, linear second-order unification, one-step rewriting, natural language understanding.

Ilkka Niemelä, Univ. Helsinki

Disjunctive Logic Programming and First-Order Theorem Proving

It is argued that merging disjunctive logic programming (DLP) and first-order theorem proving is very natural and offers advantages to both of the fields. On one hand, each DLP system contains a full first-order theorem prover: evaluating a positive query against a positive disjunctive program amounts to testing classical consequence from a set of first-order clauses. Hence, efficient theorem proving techniques are an essential ingredient in every DLP implementation. On the other hand, DLP offers knowledge representation techniques (such as closed world assumption, default values, rules with exceptions) that can be very useful for representing domain knowledge for a theorem prover.

The disjunctive well-founded semantics of Brass and Dix (D-WFS) is put forward as a promising basis for merging DLP and theorem proving paradigms. D-WFS was originally defined for ground programs as the weakest semantics invariant under some very natural program transformations. We generalize D-WFS to the first-order case and show that D-WFS has the following property: negative queries are evaluated according to minimal model entailment which is a generalization of entailment under subset minimal Herbrand models and positive queries are answered using classical entailment. This allows a modular integration of DLP and theorem proving: answering positive queries can be done using a classical theorem prover which uses a minimal model reasoner as a background reasoner for handling the negative queries.

Tobias Nipkow, Univ. Saarbrücken

Winskel is (almost) Right
Towards a Mechanized Semantics Textbook

We present a formalization of the first 100 pages of Winskel's *The Formal Semantics of Programming Languages* in the theorem prover Isabelle/HOL:

2 operational, 2 denotational, 1 axiomatic semantics, a verification condition generator, and the necessary soundness, completeness and equivalence proofs, all for a simple imperative language.

Hans de Nivelle, CWI, Amsterdam

Verification of Mathematical Proofs in Classical Logic Using Sequent Calculus.

We present a sequent calculus for set theory which has the following features: It has direct introduction and elimination rules for the element relation, and the element relation is extended to n-tuples.

In the second part of the talk we give a formalism (called proof trees) for representing proofs in this sequent calculus, which is intended to have a simplicity and charm that can compete with the lambda-terms of type theory. We show that the proof trees have a monotonicity and relevance property, and we show how cut elimination (for the classical fragment) can be implemented using proof trees.

Jens Otten, TH Darmstadt

Proof Strategies for Intuitionistic Logic

Intuitionistic logic, due to its constructive nature, has an essential significance for the derivation of verifiably correct software. According to the "proof-as-programs" paradigm of program synthesis theorems proven in a constructive manner can be interpreted a specifications of programs which are implicitly contained in the proof. Therefore we are currently working on several proof methods for intuitionistic logic:

- a non-clausal connection method (based on Wallen's matrix characterization) [2],
- a free-variable analytic tableau calculus (also based on Wallen's characterization), and
- a proof method which decides propositional intuitionistic logic via a translation into classical propositional logic and the application of a non-clausal Davis-Putnam procedure [1,4].

In the following we will shortly explain our second approach.

In classical provers usually *term unification* and Skolemization is used to express the non-permutabilities between the quantifier rules (due to the *eigenvariable condition* in the sequent calculus). To handle the non-permutabilities

between certain intuitionistic rules in a similar way we use a specialized *string unification* and extend Skolemization accordingly.

Our analytic tableau calculus is similar to the classical one. To deal with intuitionistic logic we have to assign a prefix p to each formula in the tableau where some tableau rules add a character to the current prefix of the formula. A *prefix* (or semantically spoken a *world path*) of a formula is a string and essentially describes its position in the formula tree. We use a free-variable tableau calculus, i.e. we do not only use free variables for the terms but also for the prefixes. Similar to the Skolem-term introduced in the first-order classical case we use a Skolem-character $s(x_1, \ldots, x_n)$ for the prefixes. For the Skolem-function s we use the unique *position* of the current formula in the formula tree. The variables $x_1, \ldots, x_n$ are *only* the free variables introduced in the current branch of the formula tree not in the current branch of the tableau. This means that we use a Skolemization for the term- and prefix-variables which is similar to some liberalized δ-rule in the classical tableau calculus.

We have implemented our prefix-based tableau calculus in Prolog, called ileanTAP [3]. At first a lean*TAP* like technique for *path checking* is used to prove the *classical* validity of a given first-order formula. Afterwards we try to unify the prefixes of those atoms closing the branches of the tableau proof found in the first step. If this additional string unification succeeds the formula is *intuitionistically* valid. Although the implementation of ileanTAP is comparable short (about 4 kilobytes) its performance is sometimes even better than the performance of the tableau prover ft from Sahlin et. al. (which consists of about 200 kilobytes of C source code). Due to the modular treatment of the different logical connectives the implementation can easily be adapted to deal with other non-classical logics such as the modal logics S4, D, D4, S5 and T.

[1] Daniel Korn, Christoph Kreitz. Deciding Intuitionistic Propositional Logic via Translation into Classical Logic. In W. McCune, editor, *CADE–14*, LNAI, Springer Verlag, 1997.

[2] Christoph Kreitz, Jens Otten, Stephan Schmitt. Guiding Program Development Systems by a Connection Based Proof Strategy. In M. Proietti, editor, *LOPSTR '95*, LNCS 1048, pp. 137–151, Springer Verlag, 1995.

[3] Jens Otten. ileanTAP: An intuitionistic theorem prover. *TABLEAUX '97*, LNAI, Springer Verlag, 1997.

[4] Jens Otten. On the Advantage of a Non-Clausal Davis-Putnam Procedure. *Technical Report*, AIDA-97-1, TH Darmstadt, 1997, submitted.

Larry Paulson, Univ. Saarbrücken

Formal Proofs about Cryptographic Protocols

Security protocols are formally specified in terms of traces, which may involve many interleaved protocol runs. Traces are defined inductively. Protocol descriptions model accidental key losses as well as attacks. The model spy can send spoof messages made up of components decrypted from previous traffic.

Correctness properties are verified using the proof tool Isabelle/HOL. Several symmetric-key protocols have been studied, including Needham-Schroeder, Yahalom and Otway-Rees. A new attack has been discovered in a variant of Otway-Rees (already broken by Mao and Boyd). Assertions concerning secrecy and authenticity have been proved.

The approach rests on a common theory of messages, with three operators. The operator parts denotes the components of a set of messages. The operator analz denotes those parts that can be decrypted with known keys. The operator synth denotes those messages that can be expressed in terms of given components. The three operators enjoy many algebraic laws that are invaluable in proofs.

Wolfgang Reif, Univ. Ulm

Theorem Proving in Large Theories

The motivation behind this work is the question: how can formal software verification benefit directly from automated first-order theorem proving? To answer the question we used the software verification tool, KIV ([RSS95], [Rei95], [FRSS95]) as a test environment, and did comparative experiments with four automated theorem provers as dedicated subsystems for the non-inductive first-order goals that showed up during proofs of specification- and program properties. The four provers were Otter ([WOLB92]), Setheo ([GLMS94]), $\mathcal{TAP}$ ([BHOS96]) and Spass ([WGR96]).

The challenge for the provers in this application is the large number of (up to several hundred) axioms in typical software specifications. Both the success rates and the proof times strongly depend on how good the provers are able to find out the few relevant axioms that are really needed in the proofs. We present a reduction technique for this problem. It takes the axioms of a theory and a theorem and computes a reduced axiom set by eliminating as many irrelevant axioms as possible. The proof search for the theorem then is performed in the reduced set. Comparative experiments with the four automated theorem provers showed that with the reduction technique they proved more theorems than before, and were faster for those that could be proved already without reduction.

References

[BHOS96] Bernhard Beckert, Reiner Hähnle, Peter Oel, and Martin Sulzmann. The tableau-based theorem prover $_3$TAP, version 4.0. In Michael McRobbie, editor, *Proc. 13th CADE, New Brunswick/NJ, USA*, LNCS 1104, pages 303–307. Springer, 1996.

[FRSS95] T. Fuchß, W. Reif, G. Schellhorn, and K. Stenzel. Three Selected Case Studies in Verification. In M. Broy and S. Jähnichen, editors, *KORSO: Methods, Languages, and Tools for the Construction of Correct Software – Final Report*. Springer LNCS 1009, 1995.

[GLMS94] C. Goller, R. Letz, K. Mayr, and J. Schumann. Setheo v3.2: Recent developments – system abstract. In A. Bundy, editor, *12th International Conference on Automated Deduction, CADE-12*, Springer LNCS 814. Nancy, France, 1994. for the newest version of SETHEO, see the URL: http://wwwjessen.informatik.-tu-muencen.de/forschung/reasoning/setheo.html.

[Rei95] W. Reif. The KIV-approach to Software Verification. In M. Broy and S. Jähnichen, editors, *KORSO: Methods, Languages, and Tools for the Construction of Correct Software – Final Report*. Springer LNCS 1009, 1995.

[RSS95] W. Reif, G. Schellhorn, and K. Stenzel. Interactive Correctness Proofs for Software Modules Using KIV. In *Tenth Annual Conference on Computer Assurance*, IEEE press. NIST, Gaithersburg, MD, USA, 1995.

[WGR96] C. Weidenbach, B. Gaede, and G. Rock. Spass & flotter, version 0.42. In *13th International Conference on Automated Deduction, CADE-13*, Springer LNCS, 1996.

[WOLB92] L. Wos, R. Overbeek, E. Lusk, and J. Boyle. *Automated Reasoning, Introduction and Applications (2nd ed.)*. McGraw Hill, 1992. for the newest version of OTTER, see the URL: http://www.mcs.anl.gov/home/mccune/ar/otter/#doc.

Piotr Rudnicki, University of Alberta

Mizar - an experimental data base for mathemtics

The Mizar project is a long-term effort aimed at developing software to support a working mathematician in preparing papers. A. Trybulec, the leader of the project, has designed a language—also called Mizar—for writing formal mathematics. The logical structure of the language is based on a natural deduction system developed by Jaskowski and the language mimics the ways traditional mathematics is written. The texts written in the language are called Mizar articles and are organized into a data base. The Tarski-Grothendieck set theory forms the basis of doing mathematics in Mizar. The implemented processor of the language checks the articles for logical consistency and correctness of references to other articles.

Two dacades of experience with Mizar make us believe that the organization of the database and development of the authoring language form bigger challenges for a similar project than increasing the grinding power of the logical formulae manipulator.

Mizar is a modest trial run for QED: "a project to build a computer system that effectively represents all important mathematical knowledge and techniques."

To learn more about Mizar visit:

`http://www.cs.ualberta.ca/~piotr/Mizar`, or
`http://mizar.uw.bialystok.pl/Mizar/Project.html`

Thorsten Schaub, Univ. Saarbrücken

Prolog Technology for Default Reasoning

We show how Prolog technology can be used for efficient implementation of query answering in default logics. The idea is to translate a default theory along with query into a Prolog program and a Prolog query such that the original query belongs to an extension of the default theory iff the Prolog query is derivable from the Prolog program.

Manfred Schmidt-Schauß, Univ. Saarbrücken

Deductive Analysis of functional programs

We address the analysis of programs written in a lazy functional programming language, where we focus on using the methods from automated theorem proving rather than using a theorem prover. In particular, this is an investigation of the power of abstract reduction. The idea is simple: Formulate the query syntactically, using extra constants like $\top$ or $\bot$ representing all

expressions or all undefined expression, respectively. Then reduce the expressions in the query and try to find a representation of all possible reductions. We prefer to model this as constructing a closed tableau.

The method successfully treats strictness analysis, evaluation context analysis, termination analysis and equality analysis. Prototype implementations demonstrate the power of the method.

Peter H. Schmitt, Univ. Saarbrücken

A Tableaux System for linear-time temporal logic

In this talk I present joint work with Jean Goubault-Larrecq. A full version will appear in the proceedings of TACAS'97.

We consider the propositional logic LTL_0 for linear time using the modal operators $\Box$ (for all following time points) and $\Diamond$ (at some later time point). This logic is strictly weaker than full propositional linear-time logic LTL which contains in addition the modal operator $\circ$ (at the next time point). The interest in the fragment LTL_0 comes from the fact that its satisfiability problem is NP-complete while the satisfiability problem for LTL is PSPACE-complete.
The tableau calculi known upto know for LTL, e.g. the one given by P.Wolper, use the usual tableau extension rules and in addition an expensive test for completed cycles. We present a tableau calculus based on prefixed formulas that can do without such a test. The formulas appearing in the tableau are of the following forms:

1. $[s,t]C_1,\ldots,C_k$
2. $[s,\infty[C_1,\ldots,C_k$
3. $\mid\infty\mid C_1,\ldots,C_k$

Here s,t are constants denoting time points. In addition there will appear inequalities of the form $s\leq t+n$ or $s\leq t-n$ on the tableau. The semantics relative to a temporal structure $S=(\mathbb{N},<,\pi)$ is given by:

1. $S\models[s,t]C_1,\ldots,C_k\Leftrightarrow$
   for all $s\leq i\leq t$ there is some $1\leq j\leq k$ such that $S,i\models C_j$.
2. $S\models[s,\infty[C_1,\ldots,C_k\Leftrightarrow$
   for all $s\leq i$ there is some $1\leq j\leq k$ such that $S,i\models C_j$.
3. $S\models\mid\infty\mid C_1,\ldots,C_k\Leftrightarrow$
   for infinitely many i $S,i\models C_1\wedge\ldots\wedge C_k$.

Notice the assymmetry between the interval case and the ∞-case. We present a set of invertible rules and closure conditions that consitute a sound. complete and always terminating tbaleau procedure. The set-up is modular in so far as the expansion part of the tableau procedure is independent of the closure part. We present in this talk a method for closure that uses a resolution calculus plus some graph algorithm, whioh checks unsatisfiability of the set of inequalities accumulated on a given branch.

Helmut Schwichtenberg, Univ. München

Remarks on Gentzen calculi

It is shown that permutative conversions terminate for the cut-free intuitionistic Gentzen (i.e. sequent) calculus; this proves a conjecture by Dyckhoff and Pinto. The main technical tool is a term notation for derivations in Gentzen calculi. These terms may be seen as λ-terms with explicit substitution, where the latter corresponds to the left introduction rules.

John Slaney, Australian National University

Substructural Modal Logics

This talk outlines the trasditional relevant logic R of Anderson and Belnap and the way that modal operators can be added to it to get systems which combine relevance with modality. Delicate issues arise, particularly concerning conservative extension of different versions of such logics. These observations are not new; the point of the present talk is to use them as examples to illustrate how complicated it is to combine systems, and thus to help correct the common over-optimistic impression that mixing logics is easy.

Bruce Spencer, University of New Brunswick

Two New Restrictions of Resolution

A binary resolution derivation, represented by a binary tree in which each internal node represents a resolution of its parent clauses followed by zero or more factoring operations, can be restricted by Tseitin's well-known regularity restriction. In a regular tree, no two nodes on any branch resolve upon identical literals. An edge rotation is possible in a tree, similar to an AVL rotation, if the atom resolved upon in the child node is not merged by the parent. We call a tree minimal if it remains regular after any sequence of

rotations. Although there may be exponentially many sequences, we give a linear time algorithm to tell if two minimal trees will combine to form a minimal tree. Furthermore the rank/activity restriction is introduced which may deactivate some literals within a clause, making them unavailable for further resolutions. The combination of minimality and rank/activity preserves completeness.

Shinobu Takamatsu, Osaka Sangyo University

First-Order Logic Based Situational and Dynamic Interpretation of Natural Language Descriptions in Hardware Design Specifications

For verification, synthesis and design of hardware systems, various formal specification languages have been used in a combined form, which are temporal logics, finite state machine languages, data flow graphs and hardware description languages. Furthermore, informal natural language expressions are used for the unified descriptions of time, action, state transition and causality, and for the abstract descriptions such as functions and hierarchy of devices. We propose a method for situational and dynamic interpretation of natural language descriptions as well as formal language ones in hardware design specifications. Natural language expressions are formulated by the multi-modal predicate logic integrating logics of time, action, causality and conditional. The semantics of the logic is given by first -order logic formalization based on verification-conditional and situational semantics. We present the methods for transformation of natural language expressions to the first-order logical formulas through the multi- modal logical ones, and for description of the rules on modality, device action, register transfer and control state transition by the first-order logic. A method for dynamic interpretation using belief revision is given which is based on the inference mechanism of the first-order logic and is reinforced with default reasoning and meta level reasoning. The above processing system is implemented by Prolog language.

Andrei Voronkov, Uppsala University

Herbrand's Theorem and Equational Reasoning

Recently, a number of new results were proved in the area of equational reasoning for the connection and the tableau methods. We overview these results and their relation to problems in mathematical logic related to Herbrand's

theorem and intuitionistic provability. Many of these new results can be formulated as statements about (fragments of) simultaneous rigid E-unification (SREU).

In the talk we discuss

1. Herbrand's theorem for equational reasoning and SREU.
2. SREU and second-order unification.
3. The undecidability of SREU.
4. SREU and (tree) automata.
5. The monadic case of SREU.
6. Decidable special cases of SREU.

The results discussed in the talk were obtained by J. Gallier, A. Degtyarev, Yu. Gurevich, Yu. Matiyasevich, P. Narendran, D. Plaisted, M. Veanes and the speaker.

Christoph Weidenbach, MPI Saarbrücken

The Role of Abstraction in Automated Theorem Proving

We propose a variant of ordered resolution with semantic restrictions based on interpretations which are identified by the given atom ordering and selection function. Techniques for effectively approximating validity in these interpretations are presented. We (dynamically) abstract the given clause set into a decidable fragment of first-order logic. The models of this fragment approximate the interpretations of the original clause set, such that necessary inferences can be selected and redundant clauses can be detected. The framework is shown to be strictly more general than certain previously introduced approaches. Implementation of our techniques in the SPASS prover has lead to encouraging experimental results.

Akihiro Yamamoto, TH Darmstadt

Extensions of Deductive Logic Programming for Inductive Logic Programming

In this talk we define declarative and procedural semantics for arbitrary clausal theories, where a clausal theory is a conjunction of clauses. The declarative semantics for a clausal theory is a set of ground literals which are logical consequences of the theory. Muggleton used the set without noticing that it can be regarded as a declarative semantics for the theory. The procedural semantics is called SB-derivation, which was originally called C-derivation by Plotkin. We show that the relation between the two semantics is quite similar to the one between the least Herbrand model and SLD-resolution for definite logic programs. The relation characterizes Muggleton's inverse entailment with Plotkin's relative subsumption. We also show that the declarative semantics for a conjunction of a definite program and a goal has a special structure, and it can be generated by the combination of SOLD-derivation and bottom-up evaluation of the definite program. SOLD-derivation is an extension of SLD-resolution by adding the skip operator proposed by Inoue. Our results highlights a difference between abduction and induction.

3 Panel: How to get Automated Deduction out of its corner

A panel under this topic was held Tuesday, 25. Febr. 96, with panelists Wolfgang Bibel (Chair), Ryuzo Hasegawa, Deepak Kapur, Wolfgang Küchlin, and Mark Stickel. The title was meant to reflect the contrast between the field's amazing successes and its standing within the scientific community and the public.

In the lively discussion a healthy self-confidence was exhibited by pointing to numerous successes which have been achieved in the past, successes which range from an unprecedented growth of the accumulated body of knowledge over the automatic discovery of a proof for the longstanding open problem by Robbins (positively noted even in an article of the New York Times) to everyday use of deductive systems in the hardware (and increasingly also software) community. Areas with a promising potential for applications mentioned were mathematics (proof checking/finding), synthesis and verification of hardware and software, safety technology (protokolls), natural language understanding, planning, and so forth. Even in these times of severe money shortage a sense of optimism could be felt in almost all of the contributions.

Wolfgang Bibel