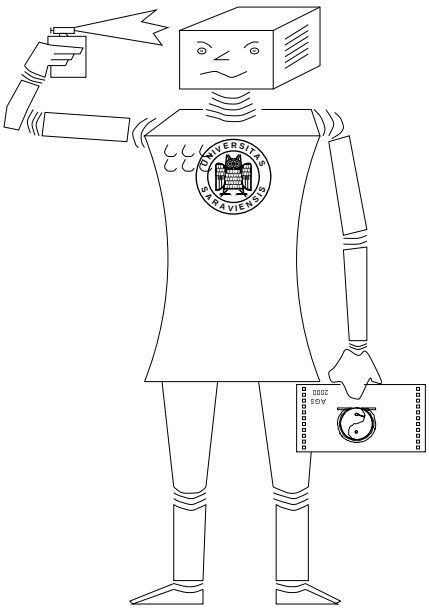
UNIVERSITÄT DES SAARLANDES
FACHBEREICH INFORMATIK
D-66041 SAARBRÜCKEN
GERMANY
WWW: http://www.ags.uni-sb.de/

SEKI Report
ISSN 1437-4447



The CALCULEMUS Autumn School 2002: Course Notes (Part III)

Christoph Benzmüller and Regine Endsuleit (Eds.)

chris@ags.uni-sb.de, endsulei@ira.uka.de

Fachbereich Informatik
Universität des Saarlandes, Saarbrücken, Germany

Fakultät für Informatik
Universität Karlsruhe (TH), Karlsruhe, Germany

SEKI Report SR-02-09

The CALCULEMUS Autumn School 2002: Course Notes (Part III)*

Christoph Benzmüller and Regine Endsuleit (Eds.)

Fachbereich Informatik
Universität des Saarlandes, Saarbrücken, Germany
chris@ags.uni-sb.de

Fakultät für Informatik
Universität Karlsruhe (TH), Karlsruhe, Germany
endsulei@ira.uka.de

Sponsors

CALCULEMUS Autumn School 2002 gratefully acknowledges the sponsorship of the following organizations and companies:

- EU CALCULEMUS Network
  http://www.eurice.de/calculemus/
- EU IST Programme, Future and Emerging Technologies (FET)
  http://www.cordis.lu/ist/fethome.htm
- Dipartimento di Matematica, Universitá di Pisa
  http://www.dm.unipi.it/
- German Research Center for Artificial Intelligence GmbH (DFKI)
  http://www.dfki.de/
- Comune di Pisa
  http://www.comune.pisa.it/doc/cittapisa.htm
- Saar Toto GmbH
  http://www.saartoto.de/
- RIACA Research Institute for Applications of Computer Algebra
  http://www.riaca.win.tue.nl/index.html

*This techreport is supported by the EU training network CALCULEMUS (HPRN-2000-00102) funded in the EU 5th framework.

Contents

1 Bruno Buchberger
RISC Linz, Austria

Formal Theory Exploration: The Lazy Thinking Paradigm

Formal Theory Exploration: The Lazy Thinking Paradigm

Bruno.Buchberger@RISC.Uni-Linz.ac.at

Calculemus Autumn School 2002, Pisa, September 2002

Overview

Formal Theory Exploration

"Theory Exploration" Paradigm as an Alternative to "Isolated Theorem Proving" Paradigm

The "Theory Exploration" paradigm has been introduced in:

B. Buchberger.
Theory Exploration Versus Theorem Proving
Proceedings of the Calculemus' 99 Workshop, Electronic Notes in Theoretical Computer Science, 23/3, Elsevier, 1999.

B. Buchberger.
Theory Exploration with Theorema.
In: Proceedings of SYNASC 2000 (2nd International Workshop on Symbolic and Numeric Algorithms in Scientific Computing), Oct.4-6,2000, Timisoara, Rumania, (T.Jebelean,V.Negru,A.Popovici eds.), Analele Universitatii Din Timisoara, Ser.Matematica-Informatica, Vol.XXXVIII, Fasc.2, 2000, pp.9-32.

Isolated Theorem Proving

knowledge base $\vdash^{?}$ theory goal formula

Prove[goal, using → knowledge–base, in → theory, by → proof–method]

More generally: Isolated Reasoning (= Proving, Solving, Simplifying, Computing)

Prove[goal, using → knowledge–base, by → proof–method]

Solve[goal, using → knowledge–base, by → solution–method]

Simplify[goal, using → knowledge–base, by → evaluation–method]

Compute[goal, using → knowledge–base, by → evaluation–method]

Theory Exploration

Given a knowledge base (in a certain theory) invent and prove "all" possible goal formulae such that

knowledge base $\vdash$ theory goal formula

More specifically,

- given a "complete knowledge base" on certain given notions,
- invent and prove a "complete" system of elementary formulae that describe the interplay of several) new notions with the given notions "completely".

"Complete" knowledge: All other knowledge can be derived by "easy" reasoning (e.g. "rewriting", "sy computation").

Theory Exploration in More Detail: Exploration Situations

An "exploration situation" is characterized by the following parameters:

- a collection *C* of known concepts
- a collection *K* of facts that (are thought to) "completely" describe the interactions (interrelations) among the known concepts
- a new concept *N*
- axioms (e.g. definitions) *A* that relate the new concept with the known concepts
- a collection *G* of goal propositions that (are thought to) "completely" explore the interaction between the new concept and the known concepts.

Theory Exploration in More Detail: Appropriate Proof (Reasoning) Method

- "Often", *K* (known facts on known concepts), *A* (axioms), and *G* (goals) of an exploration situation determine an "appropriate" proof method for proving the (basic) propositions in G.
- *K* (known facts on known concepts) may be split into a fixed theory that suggests a "special proof method" and a variable part of the knowledge base.

Example:

K ... induction axioms and equalities on zero, successor, plus
A ... inductive definition of times
G ... equalities on times

appropriate proof method: induction with simplification.

Theory Exploration in More Detail: How Does One Find "Complete" Knowledge for the New Notion?

Bottom-up: Explore "all" possible interactions of the new notion(s) with the given notions and start with the "simple" ones. (See [B. Buchberger 2000, Timisoara Proceedings].)

Top-down: Explore by the "lazy thinking paradigm".

The Lazy Thinking Paradigm

Lazy Thinking: Attempt a Proof, Analyze the Failure, and Conjecture a Lemma.

- We start from an exploration situation (*C, K, A, N, G*) with an "interesting" goal *G*.
- We attempt to find a proof of *G* (using an "appropriate" proof method P).
- If P yields a proof: ☺. (Add *G* to the knowledge base!)
- If P fails:
  - analyze the failing proof situation (current knowledge base and current goal) and
  - conjecture a lemma *L* that would make the proof possible.
- Apply the principle recursively using *L* as the new goal.

Application of this principle, in the positive case,

- does not only yield a proof of the initial goal *G*
- but also fills up the knowledge base for the new concept *N*.

The Role of Automation

The Lazy Thinking Paradigm is independent of

- whether or not the available prover P is automated
- whether or not the invention of "interesting" initial goals G is automated
- whether or not the analysis of failing proofs is automated
- whether or not invention of "useful" intermediate lemmata is automated.

However, increasing automation *is* possible in all four areas above and is a central goal of the *Theorema* pr

The principle is, however, only applicable if the (automated) prover *P*

- produces proofs in "natural style"
- and produces a proof object also in the case of failure.

These two requirements are satisfied by only a few automated provers. The two requirements are main design objectives of *Theorema.*

(The principle applies also to "solving" and "simplifying" in place of "proving".)

References on Theorema

www.theorema.org

B. Buchberger, C.Dupre, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru, W.Windsteiger.
The Theorema Project: A Progress Report.
In: Symbolic Computation and Automated Reasoning (Proceedings of CALCULEMUS 2000 Symposium on the Integration of Symbolic Computation and Mechanized Reasoning, August 6-7, 2000, St.Andrews, Scotland, M.Kerber and M.Kohlhase eds.), A.K.Peters, Natick, Massachusetts, pp.98-113.

Automation of Proving

Is a key issue in the community since many years.

Note that, by the lazy thinking paradigm, any (weak) prover *P* may become drastically more powerful (in case failure analysis and conjecture generation can be automated).

Syntactic structure of the formulae in *K* (knowlede base for given concepts), *A* (axioms for the new concept), and *G* (goals) may determine the choice of the "appropriate" prover *P* (to the extent that, in the future, this choice may be automated).

Automation of Inventing "Interesting" Goals

Interesting goals often come from "real life". Example: "I want to sort".

Interesting goals often come from mathematical "experience": In my view, mathematical experience can be cast in

- problem types (example: equations)
- knowledge types (example: rings)
- method / algorithm types (example: divide-and-conquer).

This view has been expressed for the first time in: B. Buchberger, Introduction to Mathematics for Computer Science: Prolem Types, Knowledge Types, Algorithm Types. Lecture Notes, University of Linz, 1982.

The choice of interesting goals, in the future, may be automated by trying goals from libraries of problem / knowledge / method types.

Automation of Failure Analysis and Conjecture Generation

A failing proof situation consists of

- current knowledge (initial knowledge plus temporary assumptions)
- temporary goal(s).

The conjectured lemma:

- $\forall_x$ (knowledge $\Rightarrow$ goal)$_{\text{fixed} \leftarrow x}$

i.e.

- collect current assumptions and goals and
- generalize variable-free terms.

Completeness of Knowledge versus Completeness of Provers

Complete provers: everything that follows semantically can be proved.

Complete knowledge: everything that follows can be proved by "simple" (typically incomplete) provers.

Incomplete provers are attractive because they may be more efficient (and, in many cases, complete prov not possible).

The Role of Functors (Not Pursued in this Talk)

A successful exploration of a new concept (using the lazy thinking paradigm or any other method) should be analyzed w.r.t. the "minimal knowledge" necessary in the exploration.

The result of the exploration should then be stored as the "truth transportation" property of a corresp functor.

"All of mathematics" can then be stored / generated / retrieved in compact form by a hierarchy of functo their truth transportation properties.

This is a long-term view and program for future "Mathematical Knowledge Management" (MKM).

(A new kind of "symbolic computation Bourbakism"!)

MKM 2001: 1st International Conference on MKM, RISC, Hagenberg, Austria.

MKM 2003: 2nd International Conference on MKM, Bologna, Italy.

Case Study: Commutativity of Addition

Exploration Situation

Here and in the subsequent examples we use the *Theorema* syntax, see [Buchberger et al., 1997, 2000].

Known concepts: '0' (zero), '$\square^+$' (successor function).

Knowledge about known concepts: induction axiom(s)

$$\forall_A \left(\left(A[0] \bigwedge \forall_x (A[x] \Longrightarrow A[x^+]) \right) \Longrightarrow \forall_x A[x] \right)$$

New concept: '+' (addition).

Axioms (definition) that relate the new concept with the known concepts:

Definition["addition", any[m, n],
$m + 0 = m$ " +0:"
$m + n^+ = (m + n)^+$ " + .:"]

Goal propositions: all formulae of the form

$$\forall_{x_1,\ldots,x_n} L = R$$

where *L* and *R* are terms in '0', '$\square^+$', '+', and the variables 'x_1', …, 'x_n'.

An Appropriate Proof Method

Transition from the Level of Knowledge to the Level of Inferencing: An Important Principle in Efficient Proving

In principle, goal propositions in the above exploration situation could be proved from the known facts and the axioms by any predicate logic proof method.

However, instead, more naturally and efficiently, the known facts (induction axioms) can be "lifted" from their level of being knowledge expressed in terms of formulae onto the level of being a specific inference method

(induction inference rule) and this method can be used for proving the goal propositions from the knowledge base. Here is a rough description of this special proof method:

A Simple Automated Induction Prover

Given: a formula of the form

$$\forall_{x_1,\ldots,x_n} L = R$$

The Prover:

Take all $x_1, \ldots, x_n$ "arbitrary but fixed" and try to prove '*L*=*R*' by rewriting both *L* and *R* mod assumed equalities (in the definition).

If this proof succeeds, report success (and show the proof).

If this proof fails, organize induction on x_1:

The induction base is a proof situation of the same type but with one variable less. Hence call the proof method recursively.

The induction step is a proof situation of the same type but with one variable less. Hence call the proof method recursively.

In *Theorema*, this proof method has the name 'NNEqIndProver'. (Note that, in fact, the power of this meth the class of formulae that can be proved by this method, heavily depends on the specific method us rewriting *L* and *R*.)

Examples of Proofs

A Very Simple Theorem

For proving the following proposition

Proposition["zero from left", any[m],
$0 + m = m$]

we execute the following *Theorema* call:

```
Prove[Proposition["zero from left"],
      using → Definition["addition"],
      by → NNEqIndProver]
```

```
- ProofObject -
```

▫ **The following proof is generated completely automatically by *Theorema:***

Prove:

(Proposition zero from left) $\forall_{m} (0 + m = m)$,

under the assumptions:

(Definition (addition): +0:) $\forall_{m} (m + 0 = m)$,

(Definition (addition): + .:) $\forall_{m,n} (m + n^{+} = (m + n)^{+})$.

We prove (Proposition (20): 20) by induction on *m*.

Induction Base:

(1) $0 + 0 = 0$.

A proof by simplification of (1) works.

Simplification of the lhs term:

$0 + 0 =$ by (Definition (addition): +0:)

$0 \rfloor$

Simplification of the rhs term:

$0 \rfloor$

Induction Step:

Induction Hypothesis:

(2) $0 + m_1 = m_1$

Induction Conclusion:

(3) $0 + m_1^{+} = m_1^{+}$.

A proof by simplification of (3) works.

Simplification of the lhs term:

$0 + m_1^{+} =$ by (Definition (addition): + .:)

$(0 + m_1)^{+} =$ by (2)

$m_1^{+} \rfloor$

Simplification of the rhs term:

$m_1^{+} \rfloor$

▫ **Other Theorems that Can be Proved by this Method**

Similarly, proofs for the following propositions can be generated automatically by the above method.

```
Proposition["addition from left", any[m, n],
      m+ + n = (m + n)+]
```

```
Prove[Proposition["addition from left"], using → Definition["addition"], by → NNEqIndProver]
```

```
- ProofObject -
```

```
Proposition["associativity of addition", any[m, n, p],
      (m + n) + p = m + (n + p)]
```

```
Prove[Proposition["associativity of addition"], using → Definition["addition"], by → NNEqIndProver]
```

```
- ProofObject -
```

▫ **A Failing Proof**

The following proposition cannot yet be proved by this (very simple) proof method.

```
Proposition["commutativity of addition", any[m, n],
        m + n = n + m   " + = "]
```

Failing proof:

```
Prove[Proposition["commutativity of addition"],
        using → ⟨Definition["addition"]⟩,
        by → NNEqIndProver,
        ProverOptions → {TermOrder → LeftToRight},
    transformBy → ProofSimplifier, TransformerOptions → {branches → {Proved, Failed}}];
```

The following proof attempt is generated by Theorema

Prove:

(Proposition (commutativity of addition): + =) $\forall_{m,n} (m + n = n + m)$,

under the assumptions:

(Definition (addition): +0:) $\forall_{m} (m + 0 = m)$,

(Definition (addition): + .:) $\forall_{m,n} (m + n^+ = (m + n)^+)$.

As there are several methods which can be applied, we have different choices to proceed with the proof.

Alternative proof 1: failed

The proof of (Proposition (commutativity of addition): + =)fails. (The prover "Simplifier" was unable to transform the proof situation.)

Alternative proof 2: failed

We try to prove (Proposition (commutativity of addition): + =) by induction on m.

Induction Base:

(1) $\forall_{n} (0 + n = n + 0)$.

As there are several methods which can be applied, we have different choices to proceed with the proof.

Alternative proof 1: proved

We take in (1) all variables arbitrary but fixed:

(4) $0 + n_1 = n_1 + 0$

and simplify it.

Simplification of the lhs term:

$0 + n_1 \rfloor$

Simplification of the rhs term:

$n_1 + 0 =$ by (Definition (addition): +0:)

$n_1 \rfloor$

Hence, it is sufficient to prove:

(5) $\forall_{n} (0 + n = n)$.

We prove (5) by induction on n.

Induction Base:

(6) $0 + 0 = 0$.

A proof by simplification of (6) works.

Simplification of the lhs term:

$0 + 0 =$ by (Definition (addition): +0:)

$0 \rfloor$

Simplification of the rhs term:

$0 \rfloor$

Induction Step:

Induction Hypothesis:

(7) $0 + n_2 = n_2$

Induction Conclusion:

(8) $0 + n_2^+ = n_2^+$.

A proof by simplification of (8) works.

Simplification of the lhs term:

$0 + n_2{}^+ =$ by (Definition (addition): $+.$:)

$(0 + n_2)^+ =$ by (7)

$n_2{}^+\rfloor$

Simplification of the rhs term:

$n_2{}^+\rfloor$

Alternative proof 2: pending

Pending proof of (1).

Induction Step:

Induction Hypothesis:

(2) $\underset{n}{\forall}\ (m_1 + n = n + m_1)$

Induction Conclusion:

(3) $\underset{n}{\forall}\ (m_1{}^+ + n = n + m_1{}^+)$.

As there are several methods which can be applied, we have different choices to proceed with the proof.

Alternative proof 1: failed

We take in (3) all variables arbitrary but fixed:

(9) $m_1{}^+ + n_3 = n_3 + m_1{}^+$

and simplify it.

Simplification of the lhs term:

$m_1{}^+ + n_3\rfloor$

Simplification of the rhs term:

$n_3 + m_1{}^+ =$ by (Definition (addition): $+.$:)

$(n_3 + m_1)^+\rfloor$

Hence, it is sufficient to prove:

(10) $\underset{n}{\forall}\ (m_1{}^+ + n = (n + m_1)^+)$.

We try to prove (10) by induction on *n*.

Induction Base:

(11) $m_1{}^+ + 0 = (0 + m_1)^+$.

We simplify (11).

Simplification of the lhs term:

$m_1{}^+ + 0 =$ by (Definition (addition): $+0$:)

$m_1{}^+\rfloor$

Simplification of the rhs term:

$(0 + m_1)^+\rfloor$

Hence, it is sufficient to prove:

(14) $m_1{}^+ = (0 + m_1)^+$.

The proof of (14) fails. (No applicable inference rule was found.)

Induction Step:

Induction Hypothesis:

(12) $m_1{}^+ + n_4 = (n_4 + m_1)^+$

Induction Conclusion:

(13) $m_1{}^+ + n_4{}^+ = (n_4{}^+ + m_1)^+$.

We simplify (13).

Simplification of the lhs term:

$m_1{}^+ + n_4{}^+ =$ by (Definition (addition): $+.$:)

$(m_1{}^+ + n_4)^+ =$ by (12)

$((n_4 + m_1)^+)^+\rfloor$

Simplification of the rhs term:

$(n_4{}^+ + m_1)^+ \rfloor$

Hence, it is sufficient to prove:

(15) $((n_4 + m_1)^+)^+ = (n_4{}^+ + m_1)^+$.

The proof of (15) fails. (No applicable inference rule was found.)

Alternative proof 2: failed

We try to prove (3) by induction on *n*.

Induction Base:

(16) $m_1{}^+ + 0 = 0 + m_1{}^+$.

We simplify (16).

Simplification of the lhs term:

$m_1{}^+ + 0 =$ by (Definition (addition): +0:)

$m_1{}^+ \rfloor$

Simplification of the rhs term:

$0 + m_1{}^+ =$ by (Definition (addition): + .:)

$(0 + m_1)^+ \rfloor$

Hence, it is sufficient to prove:

(19) $m_1{}^+ = (0 + m_1)^+$.

The proof of (19) fails. (No applicable inference rule was found.)

Induction Step:

Induction Hypothesis:

(17) $m_1{}^+ + n_5 = n_5 + m_1{}^+$

Induction Conclusion:

(18) $m_1{}^+ + n_5{}^+ = n_5{}^+ + m_1{}^+$.

Pending proof of (18).

□

Now, the proof method could be made more powerful by applying a more sophisticated simpli method. However, this is not the issue in this talk.

Rather, the issue is discussing strategies how fixed proof methods should be applied in the cor exploring entire theories instead of proving isolated theorems. By good exploration strategies, weak p may drastically expand their strength.

■ Exploration by Lazy Thinking

▫ Automated Lazy Thinking

Successful proof and invention of additional knowledge by the "cascade", i.e. "lazy thinking" consisting of

- a prover
- a conjecture generator (= failure analyzer + conjecture extractor).

```
Prove[Proposition["commutativity of addition"],
      using → Definition["addition"],
   by → Cascade[NNEqIndProver, ConjectureGenerator], ProverOptions → {TermOrder → LeftToRight}];
```

▫ The following sequence of proofs and proof attempts and intermediate conjectures are now generated completely automatically by *Theorema*

Prove:

(Proposition (commutativity of addition): + =) $\forall_{m,n} (m + n = n + m)$,

under the assumptions:

(Definition (addition): +0:) $\forall_{m} (m + 0 = m)$,

(Definition (addition): + .:) $\forall_{m,n} (m + n^+ = (m + n)^+)$.

As there are several methods which can be applied, we have different choices to proceed with the proof.

Alternative proof 1: failed

The proof of (Proposition (commutativity of addition): + =)fails. (The prover "Simplifier" was unable to transform the proof situation.)

Alternative proof 2: failed

We try to prove (Proposition (commutativity of addition): + =) by induction on *m*.

Induction Base:

(1) $\forall_n (0 + n = n + 0)$.

As there are several methods which can be applied, we have different choices to proceed with the proof.

Alternative proof 1: proved

We take in (1) all variables arbitrary but fixed:

(4) $0 + n_1 = n_1 + 0$

and simplify it.

Simplification of the lhs term:

$0 + n_1 \rfloor$

Simplification of the rhs term:

$n_1 + 0 =$ by (Definition (addition): +0:)

$n_1 \rfloor$

Hence, it is sufficient to prove:

(5) $\forall_n (0 + n = n)$.

We prove (5) by induction on *n*.

Induction Base:

(6) $0 + 0 = 0$.

A proof by simplification of (6) works.

Simplification of the lhs term:

$0 + 0 =$ by (Definition (addition): +0:)

$0 \rfloor$

Simplification of the rhs term:

$0 \rfloor$

Induction Step:

Induction Hypothesis:

(7) $0 + n_2 = n_2$

Induction Conclusion:

(8) $0 + n_2^+ = n_2^+$.

A proof by simplification of (8) works.

Simplification of the lhs term:

$0 + n_2^+ =$ by (Definition (addition): + .:)

$(0 + n_2)^+ =$ by (7)

$n_2^+ \rfloor$

Simplification of the rhs term:

$n_2^+ \rfloor$

Alternative proof 2: pending

Pending proof of (1).

Induction Step:

Induction Hypothesis:

(2) $\forall_n (m_1 + n = n + m_1)$

Induction Conclusion:

(3) $\forall_n (m_1^+ + n = n + m_1^+)$.

As there are several methods which can be applied, we have different choices to proceed with the proof.

Alternative proof 1: failed

We take in (3) all variables arbitrary but fixed:

(9) $m_1{}^+ + n_3 = n_3 + m_1{}^+$

and simplify it.

Simplification of the lhs term:

$m_1{}^+ + n_3 \rfloor$

Simplification of the rhs term:

$n_3 + m_1{}^+ =$ by (Definition (addition): $+.:$)

$(n_3 + m_1)^+ \rfloor$

Hence, it is sufficient to prove:

(10) $\underset{n}{\forall}\ (m_1{}^+ + n = (n + m_1)^+)$.

We try to prove (10) by induction on n.

Induction Base:

(11) $m_1{}^+ + 0 = (0 + m_1)^+$.

We simplify (11).

Simplification of the lhs term:

$m_1{}^+ + 0 =$ by (Definition (addition): $+0:$)

$m_1{}^+ \rfloor$

Simplification of the rhs term:

$(0 + m_1)^+ \rfloor$

Hence, it is sufficient to prove:

(14) $m_1{}^+ = (0 + m_1)^+$.

The proof of (14) fails. (No applicable inference rule was found.)

Induction Step:

Induction Hypothesis:

(12) $m_1{}^+ + n_4 = (n_4 + m_1)^+$

Induction Conclusion:

(13) $m_1{}^+ + n_4{}^+ = (n_4{}^+ + m_1)^+$.

We simplify (13).

Simplification of the lhs term:

$m_1{}^+ + n_4{}^+ =$ by (Definition (addition): $+.:$)

$(m_1{}^+ + n_4)^+ =$ by (12)

$((n_4 + m_1)^+)^+ \rfloor$

Simplification of the rhs term:

$(n_4{}^+ + m_1)^+ \rfloor$

Hence, it is sufficient to prove:

(15) $((n_4 + m_1)^+)^+ = (n_4{}^+ + m_1)^+$.

The proof of (15) fails. (No applicable inference rule was found.)

Alternative proof 2: failed

We try to prove (3) by induction on n.

Induction Base:

(16) $m_1{}^+ + 0 = 0 + m_1{}^+$.

We simplify (16).

Simplification of the lhs term:

$m_1{}^+ + 0 =$ by (Definition (addition): $+0:$)

$m_1{}^+ \rfloor$

Simplification of the rhs term:

$0 + m_1{}^+ =$ by (Definition (addition): $+.:$)

$(0 + m_1)^+ \rfloor$

Hence, it is sufficient to prove:

(19) $m_1{}^+ = (0 + m_1)^+$.

The proof of (19) fails. (No applicable inference rule was found.)

Induction Step:

Induction Hypothesis:

(17) $m_1{}^+ + n_5 = n_5 + m_1{}^+$

Induction Conclusion:

(18) $m_1{}^+ + n_5{}^+ = n_5{}^+ + m_1{}^+$.

Pending proof of (18).

□

□

Prove:

(Proposition (20): 20) $\forall_m (0 + m = m)$,

under the assumptions:

(Definition (addition): +0:) $\forall_m (m + 0 = m)$,

(Definition (addition): + .:) $\forall_{m,n} (m + n^+ = (m + n)^+)$.

We prove (Proposition (20): 20) by induction on *m*.

Induction Base:

(1) $0 + 0 = 0$.

A proof by simplification of (1) works.

Simplification of the lhs term:

$0 + 0 =$ by (Definition (addition): +0:)

$0 \rfloor$

Simplification of the rhs term:

$0 \rfloor$

Induction Step:

Induction Hypothesis:

(2) $0 + m_1 = m_1$

Induction Conclusion:

(3) $0 + m_1{}^+ = m_1{}^+$.

A proof by simplification of (3) works.

Simplification of the lhs term:

$0 + m_1{}^+ =$ by (Definition (addition): + .:)

$(0 + m_1)^+ =$ by (2)

$m_1{}^+ \rfloor$

Simplification of the rhs term:

$m_1{}^+ \rfloor$

□

Prove:

(Proposition (commutativity of addition): + =) $\forall_{m,n} (m + n = n + m)$,

under the assumptions:

(Proposition (20): 20) $\forall_m (0 + m = m)$,

(Definition (addition): +0:) $\forall_m (m + 0 = m)$,

(Definition (addition): + .:) $\forall_{m,n} (m + n^+ = (m + n)^+)$.

As there are several methods which can be applied, we have different choices to proceed with the proof.

Alternative proof 1: failed

The proof of (Proposition (commutativity of addition): + =)fails. (The prover "Simplifier" was unable to transform the proof situation.)

Alternative proof 2: failed

We try to prove (Proposition (commutativity of addition): + =) by induction on *m*.

Induction Base:

(1) $\forall_n (0 + n = n + 0)$.

As there are several methods which can be applied, we have different choices to proceed with the proof.

Alternative proof 1: proved

We take in (1) all variables arbitrary but fixed and prove:

(4) $0 + n_1 = n_1 + 0$.

A proof by simplification of (4) works.

Simplification of the lhs term:

$0 + n_1 =$ by (Proposition (20): 20)

$n_1 \rfloor$

Simplification of the rhs term:

$n_1 + 0 =$ by (Definition (addition): +0:)

$n_1 \rfloor$

Alternative proof 2: pending

Pending proof of (1).

Induction Step:

Induction Hypothesis:

(2) $\forall_n (m_1 + n = n + m_1)$

Induction Conclusion:

(3) $\forall_n (m_1^+ + n = n + m_1^+)$.

As there are several methods which can be applied, we have different choices to proceed with the pro

Alternative proof 1: failed

We take in (3) all variables arbitrary but fixed:

(5) $m_1^+ + n_2 = n_2 + m_1^+$

and simplify it.

Simplification of the lhs term:

$m_1^+ + n_2 \rfloor$

Simplification of the rhs term:

$n_2 + m_1^+ =$ by (Definition (addition): + .:)

$(n_2 + m_1)^+ \rfloor$

Hence, it is sufficient to prove:

(6) $\forall_n (m_1^+ + n = (n + m_1)^+)$.

We try to prove (6) by induction on *n*.

Induction Base:

(7) $m_1^+ + 0 = (0 + m_1)^+$.

A proof by simplification of (7) works.

Simplification of the lhs term:

$m_1^+ + 0 =$ by (Definition (addition): +0:)

$m_1^+ \rfloor$

Simplification of the rhs term:

$(0 + m_1)^+ =$ by (Proposition (20): 20)

$m_1^+ \rfloor$

Induction Step:

Induction Hypothesis:

(8) $m_1^+ + n_3 = (n_3 + m_1)^+$

Induction Conclusion:

(9) $m_1^+ + n_3^+ = (n_3^+ + m_1)^+$.

We simplify (9).

Simplification of the lhs term:

$m_1^+ + n_3^+ =$ by (Definition (addition): + .:)

$(m_1^+ + n_3)^+ =$ by (8)

$((n_3 + m_1)^+)^+ \rfloor$

Simplification of the rhs term:

$(n_3^+ + m_1)^+ \rfloor$

Hence, it is sufficient to prove:

(10) $((n_3 + m_1)^+)^+ = (n_3^+ + m_1)^+$.

The proof of (10) fails. (No applicable inference rule was found.)

Alternative proof 2: failed

We try to prove (3) by induction on *n*.

Induction Base:

(11) $m_1^+ + 0 = 0 + m_1^+$.

A proof by simplification of (11) works.

Simplification of the lhs term:

$m_1^+ + 0 =$ by (Definition (addition): +0:)

$m_1^+ \rfloor$

Simplification of the rhs term:

$0 + m_1^+ =$ by (Proposition (20): 20)

$m_1^+ \rfloor$

Induction Step:

Induction Hypothesis:

(12) $m_1^+ + n_4 = n_4 + m_1^+$

Induction Conclusion:

(13) $m_1^+ + n_4^+ = n_4^+ + m_1^+$.

We simplify (13).

Simplification of the lhs term:

$m_1^+ + n_4^+ =$ by (Definition (addition): + .:)

$(m_1^+ + n_4)^+ =$ by (12)

$(n_4 + m_1^+)^+ =$ by (Definition (addition): + .:)

$((n_4 + m_1)^+)^+ \rfloor$

Simplification of the rhs term:

$n_4^+ + m_1^+ =$ by (Definition (addition): + .:)

$(n_4^+ + m_1)^+ \rfloor$

Hence, it is sufficient to prove:

(14) $((n_4 + m_1)^+)^+ = (n_4^+ + m_1)^+$.

The proof of (14) fails. (No applicable inference rule was found.)

□

Prove:

(Proposition (15): 15) $\forall_{n,m} (n^+ + m = (n + m)^+)$,

under the assumptions:

(Proposition (20): 20) $\underset{m}{\forall}\ (0 + m = m)$,

(Definition (addition): +0:) $\underset{m}{\forall}\ (m + 0 = m)$,

(Definition (addition): + .:) $\underset{m,n}{\forall}\ (m + n^{+} = (m + n)^{+})$.

We prove (Proposition (15): 15) by induction on *n*.

Induction Base:

(1) $\underset{m}{\forall}\ (0^{+} + m = (0 + m)^{+})$.

We take in (1) all variables arbitrary but fixed:

(4) $0^{+} + m_1 = (0 + m_1)^{+}$

and simplify it.

Simplification of the lhs term:

$0^{+} + m_1 \rfloor$

Simplification of the rhs term:

$(0 + m_1)^{+} =$ by (Proposition (20): 20)

$m_1{}^{+} \rfloor$

Hence, it is sufficient to prove:

(5) $\underset{m}{\forall}\ (0^{+} + m = m^{+})$.

We prove (5) by induction on *m*.

Induction Base:

(6) $0^{+} + 0 = 0^{+}$.

A proof by simplification of (6) works.

Simplification of the lhs term:

$0^{+} + 0 =$ by (Definition (addition): +0:)

$0^{+} \rfloor$

Simplification of the rhs term:

$0^{+} \rfloor$

Induction Step:

Induction Hypothesis:

(7) $0^{+} + m_2 = m_2{}^{+}$

Induction Conclusion:

(8) $0^{+} + m_2{}^{+} = (m_2{}^{+})^{+}$.

A proof by simplification of (8) works.

Simplification of the lhs term:

$0^{+} + m_2{}^{+} =$ by (Definition (addition): + .:)

$(0^{+} + m_2)^{+} =$ by (7)

$(m_2{}^{+})^{+} \rfloor$

Simplification of the rhs term:

$(m_2{}^{+})^{+} \rfloor$

Induction Step:

Induction Hypothesis:

(2) $\underset{m}{\forall}\ (n_1{}^{+} + m = (n_1 + m)^{+})$

Induction Conclusion:

(3) $\underset{m}{\forall}\ ((n_1{}^{+})^{+} + m = (n_1{}^{+} + m)^{+})$.

We take in (3) all variables arbitrary but fixed:

(9) $(n_1{}^{+})^{+} + m_3 = (n_1{}^{+} + m_3)^{+}$

and simplify it.

Simplification of the lhs term:

$(n_1{}^{+})^{+} + m_3 \rfloor$

Simplification of the rhs term:

$(n_1{}^+ + m_3)^+ =$ by (2)

$((n_1 + m_3)^+)^+ \rfloor$

Hence, it is sufficient to prove:

(10) $\underset{m}{\forall}\ ((n_1{}^+)^+ + m = ((n_1 + m)^+)^+)$.

We prove (10) by induction on m.

Induction Base:

(11) $(n_1{}^+)^+ + 0 = ((n_1 + 0)^+)^+$.

A proof by simplification of (11) works.

Simplification of the lhs term:

$(n_1{}^+)^+ + 0 =$ by (Definition (addition): +0:)

$(n_1{}^+)^+ \rfloor$

Simplification of the rhs term:

$((n_1 + 0)^+)^+ =$ by (Definition (addition): +0:)

$(n_1{}^+)^+ \rfloor$

Induction Step:

Induction Hypothesis:

(12) $(n_1{}^+)^+ + m_4 = ((n_1 + m_4)^+)^+$

Induction Conclusion:

(13) $(n_1{}^+)^+ + m_4{}^+ = ((n_1 + m_4{}^+)^+)^+$.

A proof by simplification of (13) works.

Simplification of the lhs term:

$(n_1{}^+)^+ + m_4{}^+ =$ by (Definition (addition): + .:)

$((n_1{}^+)^+ + m_4)^+ =$ by (12)

$(((n_1 + m_4)^+)^+)^+ \rfloor$

Simplification of the rhs term:

$((n_1 + m_4{}^+)^+)^+ =$ by (Definition (addition): + .:)

$(((n_1 + m_4)^+)^+)^+ \rfloor$

□

Prove:

(Proposition (commutativity of addition): + =) $\underset{m,n}{\forall}\ (m + n = n + m)$,

under the assumptions:

(Proposition (15): 15) $\underset{n,m}{\forall}\ (n^+ + m = (n + m)^+)$,

(Proposition (20): 20) $\underset{m}{\forall}\ (0 + m = m)$,

(Definition (addition): +0:) $\underset{m}{\forall}\ (m + 0 = m)$,

(Definition (addition): + .:) $\underset{m,n}{\forall}\ (m + n^+ = (m + n)^+)$.

We prove (Proposition (commutativity of addition): + =) by induction on m.

Induction Base:

(1) $\underset{n}{\forall}\ (0 + n = n + 0)$.

We take in (1) all variables arbitrary but fixed and prove:

(4) $0 + n_1 = n_1 + 0$.

A proof by simplification of (4) works.

Simplification of the lhs term:

$0 + n_1 =$ by (Proposition (20): 20)

$n_1 \rfloor$

Simplification of the rhs term:

$n_1 + 0 =$ by (Definition (addition): +0:)

$n_1 \rfloor$

Induction Step:

Induction Hypothesis:

(2) $\forall_n (m_1 + n = n + m_1)$

Induction Conclusion:

(3) $\forall_n (m_1^+ + n = n + m_1^+)$.

We take in (3) all variables arbitrary but fixed and prove:

(5) $m_1^+ + n_2 = n_2 + m_1^+$.

A proof by simplification of (5) works.

Simplification of the lhs term:

$m_1^+ + n_2 =$ by (Proposition (15): 15)

$(m_1 + n_2)^+ =$ by (2)

$(n_2 + m_1)^+ \rfloor$

Simplification of the rhs term:

$n_2 + m_1^+ =$ by (Definition (addition): + .:)

$(n_2 + m_1)^+ \rfloor$

□

Some Observations

Automated failure analysis and conjecture extraction was easy in this case.

Lazy thinking:

- easy thinking
- does not only prove but invents.

Where did first conjecture (commutativity) come from?

- "experience" (a knowledge type from a library of knowledge types: "abelian semi-group etc.
- "first interaction" (suggests the left-hand side of commutativity but cannot guess the righ side!)

Case Study: Sorting

Preliminary Remarks

Emphasis in This Case Study

In this case study, the emphasis is on the application of the lazy thinking paradigm and *not* on automation.

Some / most of the proofs could be generated automatically in the current version of *Theorema*, see the d A. Craciun.

Also, we are working on the automated proof analysis and conjecture extraction used in this case study.

In the talk, I only use *Theorema* syntax and not the *Theorema* proving facilities.

Lazy Thinking

- Throw a question / problem (on the new concept) into the available magma of ("com knowledge (on given concepts) and observe what happens!
- If you feel resistance, think (reason) a little as suggested by the nature of resistance in o create the appropriate new piece of knowledge.

Diligent Thinking

Start from the availabe knowledge (on given concepts) and systematically derive (by reasoning) new kno (on the new concept) into all directions.

Extremely Lazy Thinking

- We will not even bother to have "complete" knowledge on the given concepts readily available
- but will wait until we desperately need knowledge on the given concepts and we will keep t this knowledge; thereby, we will give account of the minimal knowledge necessary and will ma

resulting "functor" between the new and given concepts as general as possible and as widely applicable as possible for further "re-use".

(This is the analogue to "re-use" and "component-based programming".)

See also

C. Schwarzweller.
Designing Mathematical Libraries Based on Requirements for Theorems.
To appear in: "Mathematical Knowledge Management", Spec. issue of Advances in Mathematics and Artificial Intelligence (B. Buchberger, G. Gonnet, M. Hazewinkel eds.), 2003.

The Problem of Sorting

Informally: The Initial, Unstructured, Desire / Problem

We start from the desire / problem that we "want to sort".

Formally (i.e. Expressed in Predicate Logic): The Process of Sorting

sorted[X] ... "a sorted version of X", "sort X",

'S' ... a unary function constant,

'X' ... a variable ranging over objects.

Informally: The Desire / Problem of Sorting

The result of sorting X should be

a "sorted version of" the object X.

Formally: The Desire / Problem of Sorting

The first, coarse, formulation of the desire of sorting in predicate logic:

```
is–sorted–version[X, sorted[X]]
```

'is–sorted–version' ... binary predicate constant,

is–sorted–version[X, Y] ... "Y is a sorted version of X",

is–sorted–version[X, sorted[X]] ... "sorted[X] is a sorted version of X",

With some meta-information:

```
Formula["sorted yields sorted version", any[X],
  is–sorted–version[X, sorted[X]]]
```

'any[X]' ... declaration of 'X' as a free variable,

"sorted yields sorted version" ... just a label, no logical significance,
(however, very important for knowledge management!),

"Formula" ... a key word; part of the label.

An alternative formulation:

```
Formula["sorted yields sorted version",
  ∀X is–sorted–version[X, sorted[X]]]
```

The Correctness Theorem

The statement of the desire / problem can also be viewed as the "correctness theorem" for the (unk process 'sorted'.

```
Theorem["sorted yields sorted version",
  ∀X is–sorted–version[X, sorted[X]]]
```

This theorem should be proved (for the function 'sorted', which is still "unknown"). In a step-wise (lazy th attempt for proving this theorem, we will also "invent" an algorithm 'sorted'.

Formal vs. Informal

Informal is in no way better than formal.

Types

Most times we are interested only in certain types of objects:

Theorem["sorted yields sorted version", any[is–tuple[X]],
$\forall_X$ is–sorted–version[X, sorted[X]]]

is–tuple[X] ... "X is a tuple",

'is–tuple' ... a unary predicate constant (describing a "type").

The Cutting Line Between "Known" and "New" Concepts

The cutting line between "known" and "new" concepts is not an objective decision but a free-will decision.

Here, it is natural to consider

'is–sorted–version' ... known (including all auxiliary notions and "necessary" knowledge),

'sorted' ... new ("unkown").

Also, we could be interested in the "inverse problem" ("reverse engineering"):

'sorted' ... known (including all "necessary" knowledge),

'is–sorted–version' ... new ("unkown"); (what properties does the known concept 'sorted' satisfy?)

The Proof Call

Let's now try to prove the correctness theorem under the assumption that we have knowledge (a "theory") on 'is–sorted–version':

Prove[Theorem["correctness for sorting by merging"],
using → Theory["is sorted version"],
by → Prover["predicate logic"]]

Lazy Thinking Round 1: Use Problem Definition

We Prove Until We are Stuck

By a few natural predicate logic proof steps, using knowledge from the Theory["is sorted version"], the following proof attempt is easily generated:

We try to prove, for arbitry X,

is–sorted–version[X, sorted]

assuming

is–tuple[X]

In fact,

is–sorted–version[X, sorted[X]]

⇔ (by Definition["is sorted version"])

$$\bigwedge \begin{cases} \text{is–tuple[sorted[X]]} \\ \text{is–permuted–version[X, sorted[X]]} \\ \text{is–sorted[sorted[X]]} \end{cases}$$

Hence, it suffices to prove the above

is–tuple[sorted[X]],
is–permuted–version[X, sorted[X]]

and

is–sorted[sorted[X]].

Here we are stuck (because we don't have any knowledge on 'sorted' so far).

We analyze the Failure and Conjecture Lemmata

From the proof attempt we can learn that, in fact, the following three lemmata are sufficient for complet
proof. ("Collect" all temporaray assumptions and "generalize" the "arbitrary but fixed" constants into univ
bound variables!)

Lemma["sorted yields a tuple", any[is–tuple[X]],
is–tuple[sorted[X]]]

Lemma["sorted yields a permuted version", any[is–tuple[X]],
is–permuted–version[X, sorted[X]]]

Lemma["sorted yields a sorted object", any[is–tuple[X]],
is–sorted[sorted[X]]]

(Note that this easy decomposition of the proof of the correctness theorem need not really work because later, in an induction proof attempt, it may turn out that we might need the the three lemmata simultaneously in order to get over the induction step. However, since we cannnot predict what will happen, we first try this easy decomposition.)

In fact, we could now try to prove these three lemmata and would, at least, obtain some suggestions of how sorted should be defined for the empty tuples and the tuples containing only one element. (Namely how?) However, let's immediately move to the next, more interesting, step in the refinement in which we throw in an algorithm type!

Notation ≈:

Just as an abbreviation, in the sequel, we will write

$X \approx Y$

interchangeably with

is–permuted–version[X, Y].

Lazy Thinking Round 2: Unfold Algorithm Type

The Principle of Unfolding Algorithm Types

From the attempt to proof the correctness of the so far unspecified algorithm 'sorted' we already obtain a suitable definition of 'sorted' for input tuples that contain at most one element.

No more information or suggestions for a possible structure of sort can possibly be inferred from the failing proof. In other words, now we have to be creative and "invent" 'sorted'. However, as already explained above, instead of inventing 'sorted' "by individual ingenuity", we use the following "unfold algorithm types" strategy:

- choose (using a little bit of good taste and experience) one of the algorithm types from a lib
algorithm types, *)
- sketch the desired algorithm (in our case 'sorted') using the chosen algorithm type (in our ca
"divide and conquer algorithm type") with all ingredient functions unspecified,
- try to prove the desired theorems using the current sketch of the algorithm ('sorted') and
properties of the ingredient functions suggested by the failing proofs.

For obtaining more refined versions of the ingredient functions, apply this structuring principle by algorithm recursively.

*) Surprisingly, although there are thousands of algorithms, there are only a few "algorithm types"!

The Algorithm Type Divide-and-Conquer

The algorithm type divide-and-conquer suggests to define a solution function S in the following way (for any

$$S[X] = \begin{cases} special[X] & \Leftarrow \; is\text{–}basic[X] \\ composed[S[one\text{–}part[X]], S[other\text{–}part[X]]] & \Leftarrow \; otherwise \end{cases}.$$

For "guaranteeing termination" (this notion should and could be made more explicit in the context of "pr logic programming" like *Theorema* programming), left–part and right–part must satisfy

$\neg$ is–basic[X] $\Rightarrow$ is–greater[X, one–part–of[X]]
$\neg$ is–basic[X] $\Rightarrow$ is–greater[X, other–part–of[X]],

where is–greater is a Noetherian partial ordering (for which the proof technique of Noetherian induc possible) on the objects X in which we are interested. In our case, we are interested in tuples X and, he possible Noetherian ordering is the ordering $\succ$ ("longer"), which we assume here to be "known" (i.e. its de and properties are in the knowledge base in the appendix).

Algorithm: Sorted (by Merging)

Following the insight we gained from the previous proof attempt we take 'is–trivial tuple' (see knowledge b the appendix) as the function 'is–basic' and the identity as the function 'special'. Hence, we obtain the fo sketch for the desired algorithm 'sorted':

Algorithm["sorted", any[is–tuple[X]],

$$\text{sorted}[X] = \begin{cases} X & \Leftarrow \text{is–trivial–tuple}[X] \\ \text{merged}[& \Leftarrow \text{otherwise} \\ \quad \text{sorted}[\text{left–split}[X]], & \\ \quad \text{sorted}[\text{right–split}[X]]] & \end{cases}$$

where 'merged', 'left–split', and 'right–split' are just other names (that anticipate their later specification) for the unspecified functions 'composed', 'one–part', and 'other–part', respectively, in the algorithm type.

We have already seen that the proof of the correctness theorem for 'sorted' can be reduced to the proof of three lemmata. Thus, we now attempt proofs for these three lemmata using the additional knowleedge provided in the definition Algorithm["sorted"].

■ Lazy Thinking Round 2.1: Sorted Yields a Tuple

▫ Proof Start: Noetherian Induction

We want to prove:

Lemma["sorted yields a tuple", any[is–tuple[X]],
is–tuple[sorted[X]]]

We attempt a proof by Noetherian induction on $\succ$. (This proof technique is suggested by the outermost universal quantifier ranging over tuples!)

For this, we take X arbitrary but fixed and assume

(A) is–tuple[X]

and

(induction hypothesis) $\underset{\substack{\text{is–tuple}[Y] \\ X \succ Y}}{\forall}$ is–tuple[sorted[Y]].

We have to prove

is–tuple[sorted[X]].

Case : is–trivial–tuple[X].

In this case,

is–tuple[sorted[X]]

$\leftrightarrow$ by Algorithm["sorted"]

is–tuple[X]

$\leftrightarrow$ by (A)

true.

Case : $\neg$ is–trivial–tuple[X].

In this case,

is–tuple[sorted[X]]

$\leftarrow$ by Algorithm["sorted"]

is–tuple[merged[
sorted[left–split[X]],
sorted[right–split[X]]]].

Here we are stuck. However, analyzing the last proof situation, we see that we could go on - using the in
hypothesis - if we knew the following lemmata.

(Again, these lemmata can be conjectured by the method: "Collect current assumptions and goals and r
ground terms by variables.")

Conjectures for Lemmata for 'left–split', 'right–split', and 'merged'

Lemma["splits are tuples", any[is–tuple[X]], with[¬ is–trivial–tuple[X]],
is–tuple[left–split[X]]
is–tuple[right–split[X]]]

Lemma["splits are shorter", any[is–tuple[X]], with[¬ is–trivial–tuple[X]],
X ≻ left–split[X]
X ≻ right–split[X]]

Lemma["merged yields tuple", any[is–tuple[X, Y]],
is–tuple[merged[X, Y]]]

Lemma["splits are shorter"] is just what we need for the termination of Algorithm["sorted"] anyway.

The above lemmata are "desires" for the auxiliary functions 'lef–split' etc. They will be the starting point for applying the lazy thinking paradigm recursively.

Continuation of Case: ¬is–trivial–tuple[X].

Having these lemmata we could continue with the proof as follows:

is–tuple[merged[
sorted[left–split[X]],
sorted[right–split[X]]]]

← by Lemma["merged yields tuple"],
(induction hypothesis),
Lemma["splits are tuples"],
Lemma["splits are shorter"]

(A) and case assumption.

Lazy Thinking Round 2.2: Sorted Yields Permuted Version

Proof Start: Noetherian Induction

We want to prove

Lemma["sorted yields a permuted version", any[is–tuple[X]],
is–permuted–version[X, sorted[X]]]

We attempt a proof by Noetherian induction on ≻:

For this, we take X arbitrary but fixed and assume

(A) is–tuple[X]

and

(induction hypothesis) $\forall_{\substack{\text{is–tuple}[Y]\\ X\succ Y}}$ Y ≈ sorted[Y].

We have to prove

X ≈ sorted[X].

Case : is–trivial–tuple[X].

In this case,

X ≈ sorted[X]

↔ by Algorithm["sorted"]

X ≈ X.

Here we are stuck. We could complete this proof if we knew the following lemma:

Lemma["trivial tuples are permuted versions", any[is–trivial–tuple[X]],
X ≈ X]

This is a property of ≈. Hence, we can assume that this is part of the "complete" knowledge on the known concept ≈. (We mark this as one of the "absolutely necessary" / "minimal" properties of the known concepts.)

▫ **Case: ¬is–trivial–tuple[X].**

We attempt a proof for the second case:

In this case,

X ≈ sorted[X]

↔ by Algorithm["sorted"]

X ≈ merged[
sorted[left–split[X]],
sorted[right–split[X]]]

Here we are again stuck. However, analyzing the last proof situation (by collecting assumptions and goals and generalizing ground terms by variables), we see that we could go on - using the induction hypothesis - if we knew Lemma["splits are tuples"], Lemma["splits are shorter"], and the following lemma:

Lemma["merged yields permuted version", any[is–tuple[X, Y, Z]], with[left–split[X] ≈ Y, right–split[X] ≈ Z],
X ≈ merged[Y, Z]]

This is a property on the new concepts 'merged', 'left–split', and 'right–split'. Thus, it will be a goal for the next, lower, round of lazy thinking. It cannot be assumed as part of the "absolutely necessary" knowledge on the known concepts!

▫ **Continuation of Case: ¬is–trivial–tuple[X].**

Having these lemmata we could continue with the proof as follows:

X ≈ merged[
sorted[left–split[X]],
sorted[right–split[X]]]

← by Lemma["merged yields permuted version"]

is–tuple[X],
is–tuple[sorted[left–split[X]]],
is–tuple[sorted[right–split[X]]],
left–split[X] ≈ sorted[left–split[X]],
right–split[X] ≈ sorted[right–split[X]]

← by (induction hypothesis),
Lemma["splits are tuples"],
Lemma["splits are shorter"]
Lemma["sorted by merging yields tuples"]

(A) and case assumption.

■ Lazy Thinking Round 2.2: Sorted Yields a Sorted Object

▫ **Proof Start: Noetherian Induction**

We want to prove:

Lemma["sorted yields a sorted object", any[is–tuple[X]],
is–sorted[sorted[X]]]

We attempt a proof by Noetherian induction on ≻:

For this, we take X arbitrary but fixed and assume

(A) is–tuple[X]

and

(induction hypothesis) $\forall_{\substack{\text{is–tuple[Y]} \\ X \succ Y}}$ is–sorted[sorted[Y]].

We have to prove

is–sorted[sorted[X]].

Case : is–trivial–tuple[X].

In this case,

is–sorted[sorted[X]]

↔ by Algorithm["sorted by merging"]

is–sorted[X].

↔ by Lemma["trivial tuples are sorted"]

true[X].

The Lemma['trivial tuples are sorted"] is assumed to be in the knowledge base of the known notions (see appendix). (We mark this property as "necessary".)

▫ Case: ¬is–trivial–tuple[X].

In this case, the proof continues like this:

is–sorted[sorted[X]]

← by Algorithm["sorted by merging"]

```
is–sorted[merged[
      sorted[left–split[X]],
      sorted[right–split[X]]]].
```

Here we are stuck. However, analyzing the last proof situation (by "collecting assumptions and goals and generalzing ground terms to variables"), we see that we could go on - using the induction hypothesis - if we knew Lemma["splits are tuples"], Lemma["splits are shorter"], and the following lemma:


```
Lemma["merged yields sorted", any[is–tuple[X, Y]], with[is–sorted[X, Y]],
  is–sorted[merged[X, Y] ] ]
```

This is a property of the new concept 'merged', which we have to treat in the next, lower, round of lazy th
Note, how lazy thinking generates the "natural" conjectures for the ingredient functions, e.g. 'merged'.

▫ Continuation of Case: ¬is–trivial–tuple[X].

Having these lemmata we could continue with the proof as follows:

```
is–sorted[merged[
      sorted[left–split[X]],
      sorted[right–split[X]]]].
```

← by Lemma["merged yields sorted"],
(induction hypothesis),
Lemma["splits are tuples"],
Lemma["splits are shorter"]

(A) and case assumption.

■ Summary of the Current Stage

▫ Verbal Description

We have new concept: 'sorted' that should satisfy a specification (a "desire"):

is–sorted[X, sorted[X]].

We have known concept 'is–sorted' (defined in terms of auxiliary know concepts like 'is–tuple' etc.) with knowledge available.

We tried to invent (by using an algorithm type and lazy thinking) a definition of 'sorted' in terms of new a notions (like 'merged' etc.) and attempted to prove that 'sorted' meets the spefication.

By doing this, we arrived naturally at specifications (intermediate lemmata) for the new auxiliary notio 'merged' etc.

Now, we can apply the same strategy to the refinement of the new auxiliary notions.

- **Picture**

new notion 'sorted'
with a "desired property"
that should be "materialized" (in terms of auxiliary notions and known notions)
in a "guaranteed" way

⇓ lazy thinking produces

a "materialization" of sorted and
auxiliary desired properties on 'sorted' and
the auxiliary notions (like 'merged')

known notions (like 'is–tuple', 'prepend', etc.)
with lots of knowledge
(some of it "absolutely necessary")

Now go on recursively for (the auxiliary desired properties of 'sorted' and) the auxiliary notions, like 'merged'.

- **Functors (Not Pursued in this Talk)**

$$\textbf{Algorithm}\Big["sorted",\ any[is\text{–}tuple[X]],$$
$$sorted[X] = \begin{cases} X & \Leftarrow\ is\text{–}trivial\text{–}tuple[X] \\ merged[\ sorted[left\text{–}split[X]],\ sorted[right\text{–}split[X]]\ ] & \Leftarrow\ otherwise \end{cases}\Big]$$

Every formula, in particular the above algorithm, can be viewed as a "functor":

- Given: 'is–trivial–tuple', 'merged', 'lef–split', 'right–split'.
- Produce: 'sorted'.

Functors do not only transport computations but also knowledge!

Knowledge transport can / should be studied in two directions for building up a library of "re-usable" knowledge.

Case Study: Merging etc.

Conclusions

The lazy thinking paradigm has several ingredients:

- proving
- problem / knowledge / algorithm types
- failure analysis and conjecture generation
- analysis of minimal requirements and extraction of "re-usable" knowledge.

Each of the ingredients should be studied for possible automation (computer-support).

Thereby, "doing mathematics" will be possible and necessary only on higher and higher levels.

The essence of mathematical creativity:

- Mathematics is for people who want to be "lazy".
- For this, from time to time, one must be extremely hard working.

2 Michael Kohlhase
Carnegie Mellon University, Pittsburgh, USA

Tutorial: MathWeb, MBase, and OmDoc

MATHWEB: Web-Based Mathematics

(OMDoc, MATHWEB-SB, MBASE and all that)

⚠ Mathematics ≙ Anything Formal ⚠

MICHAEL KOHLHASE

School of Computer Science — Fachbereich Informatik

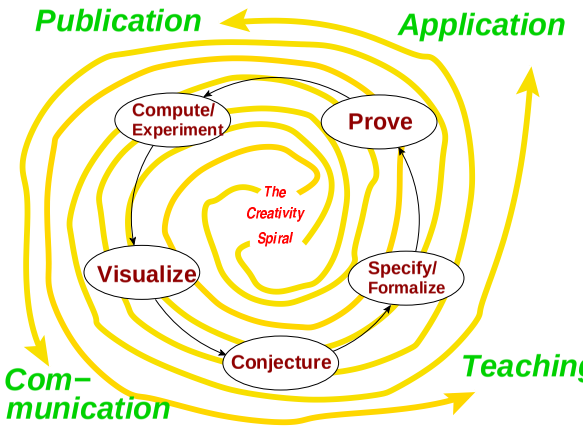
Carnegie Mellon University — Universität des Saarlandes

Pittsburgh, USA — Saarbrücken, Germany

http://www.cs.cmu.edu/~kohlhase

(supported by the German Research Council under a Heisenberg grant)

The way we do math will change dramatically



Mathematical Creativity Spiral [Buchberber '95]

- Every step will be supported by mathematical software systems
- Towards an infrastructure for web-based mathematics!

Towards an Infrastructure for Web-Based Mathematics

- Web-based math requires an open software environment that enables
  - modularization (to keep it manageable)
  - distribution (to use all those machines)
  - inter-operability (to use other peoples systems)
- the MATHWEB approach: (Plug-And-Play mathematics over the web)
  - connect math services by a common math software bus
  - obtain services by encapsulating legacy software or orig. implementation
- Observation: The value and the difficulty lies in the communication
  - we need open standards on every level ($2n$ vs. n^2translations)
  - we need to manage context and semantics (what does it all mean)

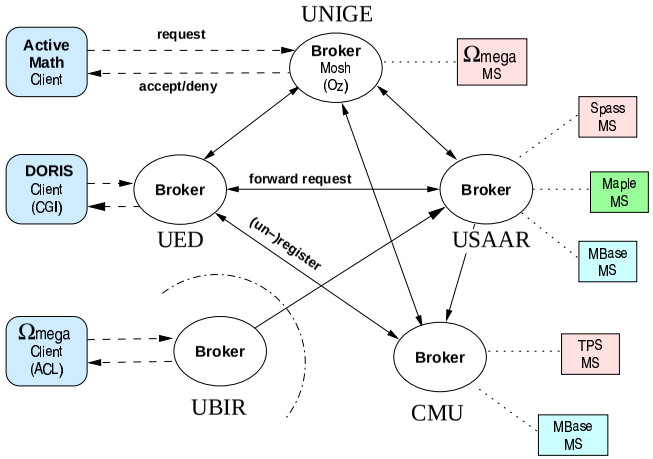
MATHWEB from the user's point of view

The Doris Discourse Representation System

(applying theorem provers without the pain)

Johan Bos

MATHWEB: The Current Architecture



- Content Language: OMDoc (Open Mathematical Documents)

Plot of the Course Capsule

- the role of communication and ontologies in MATHWEB
- (document) markup formalisms for mathematics
- OMDoc: open mathematical documents
- MATHWEB-SB: the MATHWEB Software Bus
- MBASE: a knowledge base for mathematical theories

Dimensions of Communication among Math Services

Layer	Message	Performative	Content	...
Protocol	XML-RPC	KQML	OMDoc	...

Level	formulae/objects	statements	theories	service descr.
Protocol	OpenMath/MathML	OMDoc		OMRS...

Addressee	content (machine)	presentation (human)
Protocol	OpenMath, C-MathML, Mathematica...	Html, PS, PDF, P-MathML

Why mathematics? all the problems of e.g. e-biz in a nutshell

Problem	Efficiency		Safety	
Aspect	translation	context	reference	semantics
Solution	standardize	knowledge base services		standardize

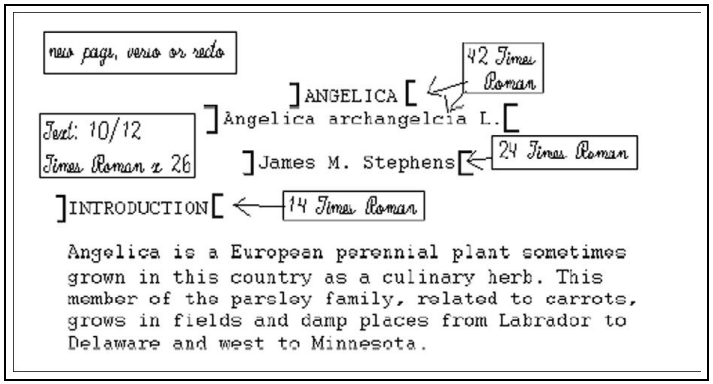
Semantics of communicated Objects?

- Why is this an issue? (preserving the meaning across transport)
  - e.g. the Ricci-Tensor $\mathcal{R}^{ij}$ differs between schools of physicists by a factor of 2!
  - Is this unit in psi or erg? (Remember the Mars orbiter †1999)
- OPENMATH/C-MATHML approach: Objects as logical formulae
  - sufficient: semantics of constants (that of var, appl, bind is well-known)
  - specify semantics by reference to joint ontology ($\Longrightarrow$ OMDoc)
- MBASE as an ontology-server for MATHWEB
  - establishes unique reference for objects (distinguished URI)
  - serves knowledge on demand (just-in-time math)
  - offers dynamic communication caches (local context)
  - seamless integration of local and global context
  - Problems: Caching, version management, distributed garbage collection

Document Markup Formalisms

Document Markup

- Definition (Document Markup)
  Document markup is the process of adding codes to a document to identify the structure of a document or the format in which it is to appear.



Styles of Markup

- Definition (Presentation Markup)
  A markup scheme that specifies document structure to aid document processing by humans
  - e.g. `*roff`, Postscript, DVI, early MS Word, low-level TEX
  - \+ simple, context-free, portable (verbatim), easy to implement/transform
  - – inflexible, possibly verbose,
- Definition (Content Markup)
  A markup scheme that specifies document structure to aid document processing by machines or with machine support.
  - e.g. LATEX (if used correctly), Programming Languages, ATP input
  - \+ flexible, portable (in spirit), unambiguous, language-independent
  - – possibly verbose, context dependent, hard to read and write

Content vs. Presentation by Example

Format	Representation	Content?
LATEX	`{\bf proof}:...\hfill\Box`	`\begin{proof}...\end{proof}`
HTML	`<font size=+2><b>...</b></font>`	`<h1>...</h1>`
LISP	$8+\sqrt{x}^3$	`(power (plus 8 (sqrt x)) 3)`
TEX	`$\{f\|f(0)> 0{\rm and}f(1)<0\}$`	$\{f\|f(0) > 0 \text{ and } f(1) < 0\}$
TEX	`$\{f\|f(0)> 0$ and $f(1)<0\}$`	$\{f\|f(0) > 0 \text{ and } f(1) < 0\}$

- We consider these to be representations of the same content (object)
- Problem: Transformations between presentation and content Markup
  - Content $\rightsquigarrow$ Pres.: usually done by styling (++ user-adaptivity)
  - Pres. $\rightsquigarrow$ Content: Heuristic Process(e.g. binomials $\binom{n}{k}$ vs. C_k^m vs. C_n^k)

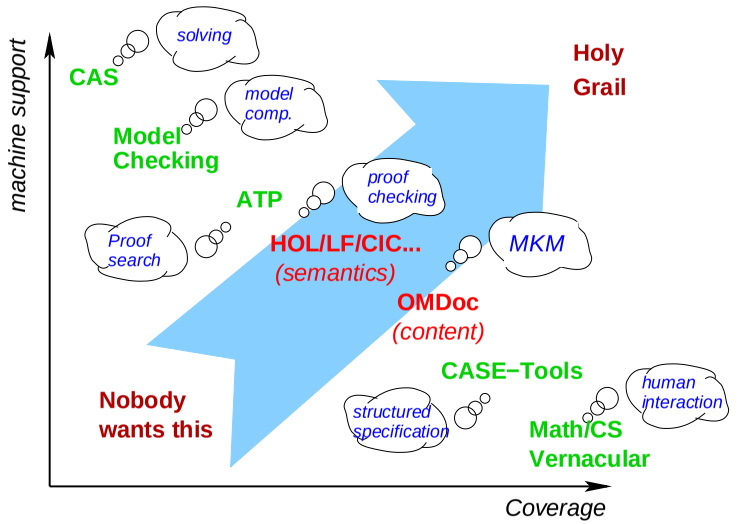
Content vs. Semantics/Formalization

- Content: logic-independent infrastructure
  Identification of abstract syntax, “semantics” by reference for symbols.

```
<apply>
  <plus/>
  <csymbol definitionURL="mbase://numbers/perfect#the-smallest"/>
  <cn>2</cn>
</apply>
```

- Semantics: establishing meaning by fixing consequences
  adds formal inference rules and axioms.
  - Mechanization in a specific system (Thm Prover or Proof Checker)
  - logical framework (specify the logic in the system itself)

Situating Content Markup: Knowledge Management



Added-value services with OMDoc/MathWeb/MBase

- cut and paste (cut output from web search engine and paste into CAS)
- automatically proof checking formal argumentations (bridge verification?)
- math explanation (e.g. specialize a proof to a simpler special case)
- semantical search for mathematical concepts (rather than keywords)
- data mining for representation theorems (find unnoticed groups out there)
- classification (given a concrete math structure, is there a general theory?)
- personalized notation (implication as $\rightarrow$ vs. $\supset$, or Ricci as $\frac{1}{1}\mathcal{R}^{ij}$ vs. $2\mathcal{R}^{ij}$)
- user-adapted documents (ActiveMath, Course Capsules)

Web Standards for Formula Markup

language	MathML	OpenMath
by	W3C Math WG	OpenMath society
origin	math for HTML	integration of CAS
coverage	cont+pres; K-14	content; extensible
status	Version 2 (II 2001)	standard (IV 2001)
activity	maintenance	maintenance $\rightsquigarrow$ OM2
Info	`http://w3c.org/Math`	`http://www.openmath.org`

C-MathML and OpenMath are equivalent (almost)

OpenMath	MathML
`<OMBIND>`	`<apply>`
`<OMS cd="quant1" name="forall"/>`	`<forall/>`
`<OMBVAR>`	`<bvar>`
`<OMATTR>`	
`<OMATP>`	
`<OMS cd="sts" name="type"/>`	
`<OMS cd="setname1" name="N"/>`	
`</OMATP>`	
`<OMV name="a"/>`	`<ci type="nat">a</ci>`
`</OMATTR>`	
`</OMBVAR>`	`</bvar>`
`<OMA>`	`<apply>`
`<OMS cd="relation1" name="geq"/>`	`<geq/>`
`<OMV name="a"/>`	`<ci type="nat">a</ci>`
`<OMI>0</OMI>`	`<cn>0</cn>`
`</OMA>`	`</apply>`
`</OMBIND>`	`</apply>`

Part of the Arithmetic CD

```
<CD>
  <CDName> arith1 </CDName>
  <CDURL>http://www.nag.co.uk/Projects/openmath/corecd/cd/arith1.ocd </CDURL>
  <CDReviewDate> 2000-09-01 </CDReviewDate>
  <CDStatus> experimental </CDStatus>
  <CDDate> 1999-07-15 </CDDate>
  <CDVersion> 1.02 </CDVersion>
  <CDUses><CDName>alg1</CDName><CDName>integer</CDName>...</CDUses>
  <Description>
    This CD defines symbols for common arithmetic functions.
  </Description>

  <CDDefinition>
    <Name> plus </Name>
    <Description>An nary commutative function plus.</Description>
    <CMP> a + b = b + a </CMP>
    <FMP>∀ a, b. a + b = b + a</FMP>
  </CDDefinition>
</CD>
```

- how can we do better? (look at data!)

Example Data (Textbook Theorem)

3.2 Naive Set Theory

⋮

Theorem 3.1.7 (Cantor):
Let S be a set, then S has a smaller cardinality than its power set $\wp(S)$.
Proof: We prove the assertion by diagonalization. Assume that there is a surjective mapping $F: S \longrightarrow \wp(S)$. Now let D be the set $\{a \mid a \notin F(a)\}$; we show that $D \notin \Im(F)$: if there were a pre-image $b \in S$ (i.e. $D = F(b)$), then assuming $b \in D$ we can obtain $b \notin D$, which is a contradiction.

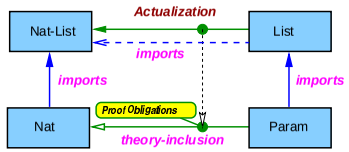
- Implicit Knowledge:
  - cardinality: $|S| < |T|$, iff there is no *surjective* mapping $F: S \longrightarrow T$.
  - Alternative: by the absence of *injective* functions $F: T \longrightarrow S$.
  - surjective: $f: S \longrightarrow T$ is, iff $\forall b \in T$, $\exists a \in S$, such that $f(a) = b$.
  - power set $\wp(S)$ of a set S is the set of all subsets of S.

OMDoc: An Open Markup Format for Mathematical Documents

OMDoc in a Nutshell (three levels of modeling)

Formula level: OpenMath/C-MathML
• Objects as logical formulae
• semantics by ref. to theory level
	`<OMA>`
	`<OMS cd="arith1" name="plus"/>`
	`<OMS cd="nat" name="zero"/>`
	`<OMV name="N"/>`
	`</OMA>`

Statement level:
• Definition, Theorem, Proof, Example
• semantics explicit forms and refs.
	`<defn for="plus" type="rec">`
	`<CMP>rec. eq. for plus</CMP>`
	`<FMP>X+0 = 0</FMP>`
	`<FMP>X+s(Y) = s(X+Y)</FMP>`
	`</defn>`

Theory level: Development Graph • inheritance via symbol-mapping • theory-inclusion by proof-obligations • local (one-step) vs. global links	

The Statement Level in OMDoc

- Text elements: `omtext`, `CMP`
- Mathematical elements: (carry the mathematical content)
  - Formulae: `FMP` (contains OpenMath repns. of logical formulae)
  - Signature: `symbol`, `definition`,
  - Properties: `assertion` (`assumption`, `conclusion`) `example`
  - Aux.: `exercise`, `omlet`, ...
- Proofs $\widehat{=}$ List of proof steps
  - Steps: `hypothesis`, `derive`, `conclude`
  - in steps: `CMP`, `FMP`, `method`, `premise`, `proof`
- (Meta data: Dublin Core Representation)

Example: an OMDoc Definition

```
<definition id="c6s1p4.d1" for="monoid">
  <CMP> A <with role="definiens">monoid</with> is a structure
    (M,*,e) where (M,*) is a semi-group and e is a unit for *. </CMP>
  <FMP>∀M,*,e.s_grp(M,*) ∧ unit(e) ⇒ monoid(M,*,e)</FMP>
</definition>
```

- Make use the co-occurrence of formal and informal representations by cross-referencing (OMDoc extends OpenMath)

```
<CMP> A <with role="definiens">monoid</with> is a structure
  <OMOBJ xref=""/> if <OMOBJ xref=""/> is a semi-group
  and <OMOBJ xref=""/> is a unit. </CMP>
<FMP>∀M,*,e.s_grp(M,*) ∧ unit(e) ⇒ monoid(M,*,e)</FMP>
<CMP xml:lang="de"> Eine Struktur <OMOBJ xref=""/> heisst
<with role="definiens">monoid</with> ist falls <OMOBJ xref=""/>
 eine Halbgruppe ist und <OMOBJ xref=""/> eine Einheit. </CMP>
```


A Piece of OMDoc Proof (also supports λ-Terms)

```
<proof id=P1>
 <hypothesis id="P1.1" discharged-in="P1.5">...</hypothesis>
 ...
 <derive id="P1.1">
  <CMP>By <ref xref="reals.A2"/> we have z+(a+(−a))=a+(−a).</CMP>
  <FMP><conclusion>(z+a)+(−a)=z+(a+(−a))</conclusion></FMP>
  <method xref="omega.omdoc#byctx(foralli*@base)">z a −a</method>
  <premise item="reals.A2"/>
 </derive>
 ...
 <conclude>...</conclude>
</proof>
```

- Infrastructure for sequent calculus (seems to generalize everything else)
- tight integration of NL and formal content (e.g. structure informal proofs)

Modular Specification of Presentation Data

- OMDoc needs presentation style sheets. (raw OMDoc illegible)
- Problem: Documents define new symbols with specialized notations
- Solution: Store presentation information locally (meta-language)
  generate document/collection-specific style sheets

```
<presentation for="binomial" parent="OMA">
 <use format="default" fixity="infix">choose</use>
 <use format="TeX" lbrack="\bigl({" rbrack="}\bigr)">\atop</use>
 <use format="pmml" element="mfrac" attributes="linethickness='0'"/>
</presentation>

<presentation for="power" parent="OMA" fixity="infix"
   crossref-symbol="no" precedence="200" bracket-style="lisp">
 <use format="html" fixity="prefix" element="sup"/>
 <use format="TeX">^</use>
 <use format="pmml" element="msup"/>
</presentation>
```


Example from the DLMF (Airy Functions)

A.7 Asymptotic Expansions

Definition (zeta-def) A local abbreviation $\zeta = \frac{2}{3} \cdot z^{\frac{3}{2}}$

Definition (AI.AS.Z.u-def)

u_0=1 u_s=$\frac{(2\cdot s+1)\cdot(2\cdot s+3)\cdot(2\cdot s+5)\cdot\cdots\cdot(6\cdot s-1)}{216^s\cdot !(s)}$

Definition (AI.AS.Z.v-def)

v_0=1 v_s=$-(\frac{6\cdot s+1}{6\cdot s-1} \cdot u_s)$

Theorem (eq:AI.AS.A)

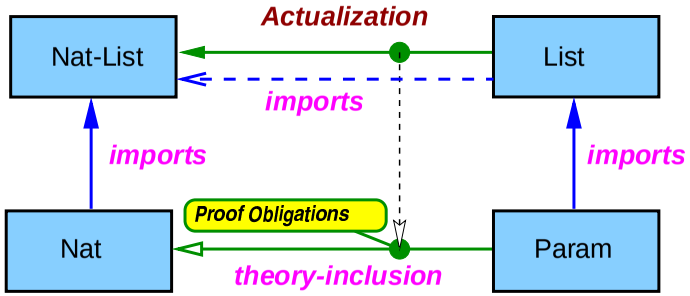
$Ai(z) \sim \frac{e^{-\zeta}}{2\cdot \sqrt[2]{\pi}\cdot z^{\frac{1}{4}}} \cdot \Sigma_{s=1}^{\infty}(-)^s \cdot \frac{u_s}{\zeta^s}$, if $|\mathbf{ph}(z)| < \pi$

Theorem (eq:AI.AS.AP)

$Ai'(z) \sim -\frac{z^{\frac{1}{4}}\cdot e^{-\zeta}}{2\cdot \sqrt[2]{\pi}} \cdot \Sigma_{s=1}^{\infty}(-)^s \cdot \frac{v_s}{\zeta^s}$, if $|\mathbf{ph}(z)| < \pi$

Theory Management in OMDoc

- Theory management follows Dieter Hutter's development graph model
- Definition and Theorem links (global and local)



- Supports a notion of theory reuse and theory change.
- theories and links represented by special OMDoc elements

A Structured Theory: Lists of Natural Numbers

```
<theory id="Param">
  <symbol id="Elem" type="sort"/>
</theory>

<assertion id="obligation"/>
```

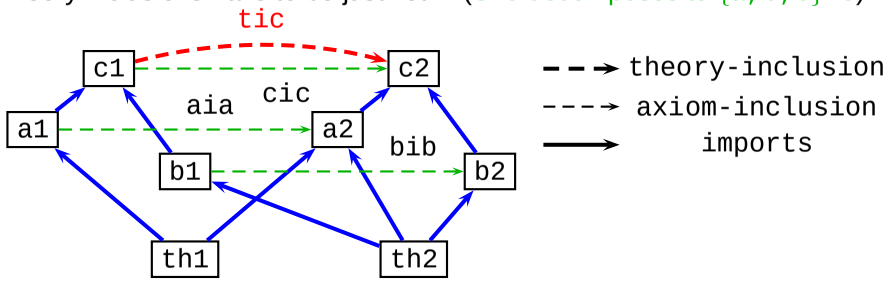
```
<theory id="List">
  <symbol id="List-sort" type="sort"/>
  <symbol id="cons"/>
  <symbol id="nil"/>
</theory>
```

```
<theory id="nat-list.thy">
  <imports id="nat-list.im-nat"     type="global" from="nat-thy"/>
  <imports id="nat-list.im-Element" type="local"  from="List">
    <morphism id="elem-nat"><requation>
      <pattern><OMOBJ><OMS cd="Param" name="Elem"/></OMOBJ></pattern>
      <value><OMOBJ><OMS cd="nat.thy" name="Nat"/></OMOBJ></value>
  </requation></morphism></imports>
  <inclusion item="elem-nat-incl"/>
</theory>

<axiom-inclusion id="elem-nat-incl" from="nat.thy" to="Element" by="obligation">
  <morphism id="elem-nat-incl-morph" base="elem-nat"/>
</axiom-inclusion>
```

Statements about Theories (Theory Inclusions)

- Theory Inclusion: All axioms in the source th. are theorems in the target
- Application: transport of theorems, change of notation. (e.g. Ricci-Tensor)
  - just generate your theory by supplying a morphism that adapts $\mathcal{R}^{ij}$
- Theory inclusions have to be justified (`tic` decomposes to `{a,b,c}ic`)



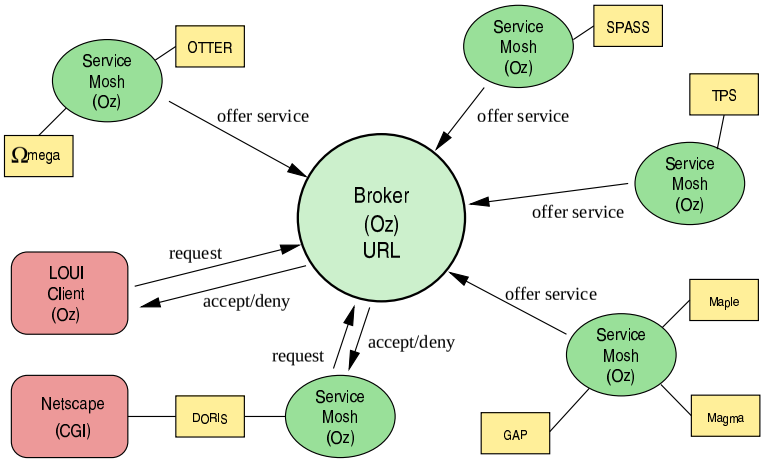
What math software systems speak OMDoc?

- ΩMEGA, *λClam*, INKA, TPS, PVS (theorem provers)
- TRAMP, P.rex (proof explanation)
- MBASE (a mathematical knowledge base)
- presentation systems into LaTeX, HTML, MATHML, (via XSL)
- migration from LaTeX, HTML, CASL, PPT, NB (⚠ heuristic process)
- transformation to and from OPENMATH CDs
- The ACTIVEMATH math tutor, CMU course capsules.
- All connected as web services by MATHWEB-SB (OMDoc as interface)

MATHWEB-SB: The MATHWEB Software Bus

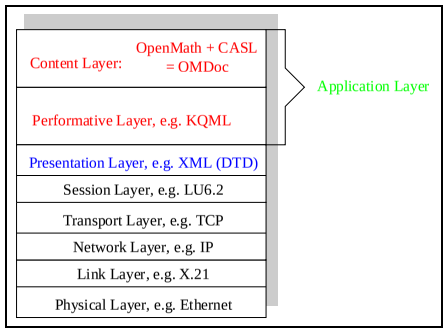
Jürgen Zimmer

MATHWEB: The Current Architecture



A Reconstruction in Terms of Agent Technology

- Use established standards for communication! ($2n$ vs. n^2translations)
- Interlingua: Knowledge Qery and Manipulation Language
- Ontolingua or Content Language: OMDoc



KQML Overview

- language and protocol for exchanging information and knowledge among software agents.
- extensible set of performatives, which defines the permissible operations that agents may attempt on each other's knowledge and goal stores.
- We use an XML encoding for KQML here. (for didactic purposes)
- KQML is being re-invented in several techologies. (Web Services)
  - UDDI (Service discovery language) ($\widehat{=}$ advertise)
  - WSDL (Web service Description Language) (@ W3C)
  - SOAP (Simple Object Access Protocol) (message layer)
  - OWL (Ontology Web Language) (@W3C)

KQML Example (Knowledge Base Lookup)

- Definition for surjectivity using the KQML `ask-one` performative.

```
<ask-one sender="kqml://goedel.ags.uni-sb.de#loui-3"
         receiver="kqml-xml://mbase.org#mbase"
         reply-with="id1" ontology="mbase" aspect="definition">
  <commonname>surjective</commonname>
</ask-one>
```

- Answer: Definition in OMDoc format using `tell`.

```
<tell sender=kqml-xml://mbase.org#mbase"
      receiver="kqml://goedel.ags.uni-sb.de#loui-3"
      reply-with="id2" in-reply-to="id1" language="OMDoc">
  <definition Ident="surjective" theory="function">
    <defmp>$\forall\alpha\beta.\lambda S_{\alpha\to o}T_{\beta\to o}F_{\alpha\to\beta}.\forall Y_\beta.SX \Rightarrow \exists X\_\alpha.FX = Y$</defmp>
  </definition>
</tell>
```

- alternatively use `ask-all` or `stream-all` for more hits.

Flexible Agent Communication in KQML

```
<advertise sender=kqml-xml://mathweb.org#mbase"
           receiver="kqml://mathweb.org#broker"
           language="kqml-xml" mvars="F">
  <ask-one sender="kqml://goedel.ags.uni-sb.de#loui-3"
       receiver="kqml-xml://mbase.org#mbase"
       reply-with="id1" ontology="mbase" aspect="definition">
   F</ask-one>
</advertise>
```

- From server to client or broker to advertise functionality (client/broker stores this in a name-DB for later reference)
- Furthermore, on-demand strategies, e.g. `standby`.
  - From client to server to initiate stream; server replies with `ready`.
  - client: `next`, `next`, `next`, `discard` (or server sends `eos`)

Theorem Proving with KQML

```
<achieve sender="A" receiver="kqml-xml://mbase.org#atp"
         reply-with="id1" language="OpenMath">
   <OMOBJ><OMA><OMS cd="reasys" name="determined">F</OMA></OMOBJ>
</achieve>

<subscribe sender="A" receiver="kqml-xml://mbase.org#atp"
           reply-with="id2" language="OpenMath">
   <ask-if sender="A" receiver="kqml-xml://mbase.org#atp"
           reply-with="id3" language="OpenMath">
     <OMOBJ><OMA><OMS cd="reasys" name="valid">F</OMA></OMOBJ>
     <OMOBJ><OMA><OMS cd="reasys" name="proper-sat">F</OMA></OMOBJ>
     <OMOBJ><OMA><OMS cd="reasys" name="unsatisfiable">F</OMA></OMOBJ>
   </ask-if>
</achieve>
```


MATHWEB Features

- Dynamic web of brokers (matchmaker agents)
  (performatives: `register,advertise,broker-one,...`)
- Mathematical Services (server agents)
  (performatives: `advertise,achieve,register,...`)
- MATHWEB clients (server agents)
  (performatives: `ask-one, ask-all, tell, ...`)
- Support for firewalls
- Service specification for first-order reasoning systems
  (to abstract from system names)

~ 30 MATHWEB Services at the moment

- FO Automated theorem provers: OTTER, SPASS, PROTEIN, BLIKSEM, VAMPIRE, E, WALDMEISTER ...
- Proof Transformers: from these to Natural Deduction: TRAMP
- Model Generators: SATCHMO, MACE, KIMBA, SEM
- HO Theorem provers: ΩMEGA, TPS, LEO, λ*Clam* ...
- Computer Algebra Systems: MAPLE, MAGMA, GAP
- User Interface: $\mathcal{LOUI}$ (runs as an agent on client machine)
- Proof Presentation: Verbalization in natural language (English)
- Contstraint Solvers: COSIE, CHORUS
- Various: RDL Rewriting Engine, HR Concept Formation System, Syntax transformers, MBASE Knowledge base

Interfaces: MATHWEB-SB offers

- ... a HTTP Server (human consumption)
  $\rightsquigarrow$ Service accesss via HTML forms
- ... XML-RPC interface (machine-consumption)
  - XML-encoded Remote Procedure Calls
  - Simple and easy to use, wide-spread
  - > 50 implementations in many programming languages
  - Internet Explorer & Mozilla 1.x as XML-RPC clients
- offering and accessing services is language-independent!

Sample XML-RPC to MATHWEB-SB

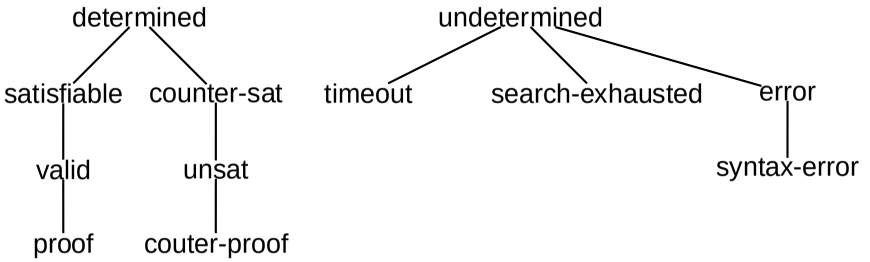
```
<methodCall><methodName>Broker.getService</methodName>
 <params><param><value><string>SPASS</string></value></param></params>
</methodCall>

<methodCall><methodName>prove</methodName>
 <params><param><struct>
  <member><name>1</name><value><string>
   include('Axioms/EQU001+0.ax').
   include('Axioms/GRP004+0.ax').
   input_formula(conjecture118,conjecture,(! [B,C,D] :
    ((equal(inverse(B),C) &amp;amp; equal(multiply(C,B),D) ) &amp;lt;=>
     (equal(multiply(B,C),D) &amp;amp; equal(multiply(C,B),D)
                             &amp;amp; equal(inverse(C),B))))).
   </string></value></member>
   <member><name>syntax</name><value><string>tptp</string>
   <member><name>timeout</name><value><int>40</int></value></member>
  </struct></param></params>
</methodCall>
```


Improved Usability: Service Specifications

All first-order ATPs . . .

- . . . offer the same interface: `prove`(ProblemString Syntax Timeout ...)
- . . . accept standard problem formats TPTP and OMDoc
- . . . return on `achieve` of specified states:



MATHWEB Future & Availability

- Standard Interfaces (SOAP)
- Service Descriptions (Web Services)

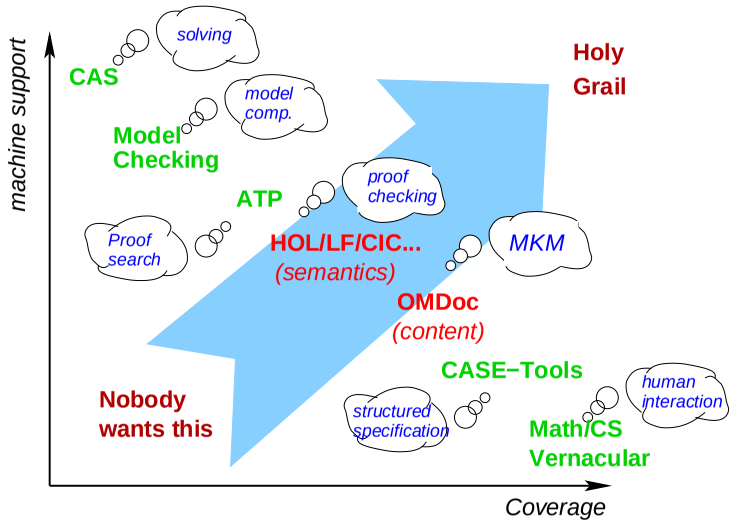
  ⇝ extensions of WSDL, UDDI

- MATHWEB-SB is implemented in MOZART (`http://www.mozart-oz.org`)

  ⇝ Unix, Linux, MacOS X, Windows

- Binary distribution (Linux, Solaris) and sources (Gnu Public License) available at `http://www.mathweb.org/mathweb`

Situating OMDoc: Math Knowledge Management



MBase: A Knowledge Base for Mathematical Theories

Andreas Franke, Markus Moschner

MBase, a knowledge base of math theories

- This has been attempted before! (Principia Math., Bourbaki,...)
- This time stress the infrastructure aspect (Open Source Model)
  - enable easy and powerful browsing (personalization, MathML)
  - high-level (semantic) search (commutativity: $X(Y,Z) = X(Z,Y)$)
  - distributed Internet support: (local working KB vs. archive KB)
  - version management & concurrent access (like CVS for cooperation)
  - offer added-value inference services (enlist MathWeb)
  - large-scale structure for navigation & reuse (theory graph, inheritance)
- MBase as a MathWeb component. (not only for human consumption)
  - situated vs. stateless communication of mathematical services

The MBase System: Architecture

- MathWeb for distribution (mozart-internal and XML-RPC)
- OMDoc for Communication (XML/OpenMath/MathML-based)
  - Precise document model allows to decouple interface development
- RDBMS for persistence (large amount of data)

  currently: MySQL (archive server) JDom impl. (scratch-pad/cache)
  - encode mathematical structure in database model
  - use Oz pickling to store mozart data structures as strings in DB.
  - use concurrent constraints in mozart to manipulate logic terms.
- InKa for management of theory change
  - management of inheritance structure, proof obligations
  - (concurrent access and update) (like CVS, but object-level diff/patch)

An Experiment in Data/Knowledge Integration

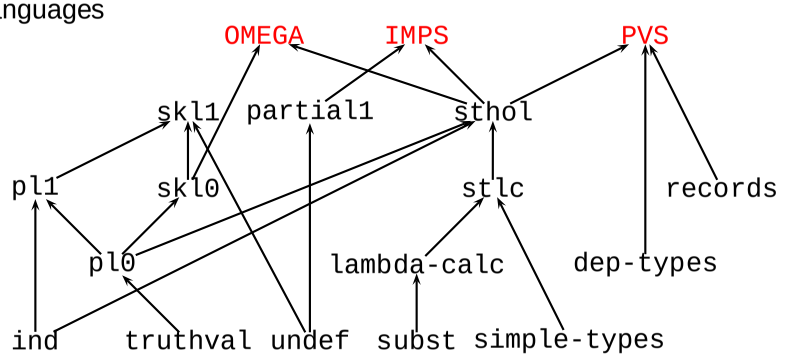
- Goal: connect various theorem proving systems to MBase
- Systems: ΩMEGA, InKa, Pvs, λ*Clam*, Tps, Imps and CoQ
- Similarities: (all descendents of AutoMath)
  - Tps, Pvs, ΩMEGA, λ*Clam*, and Imps based on simply typed λ-calculus.
  - ΩMEGA, InKa, Imps and Pvs have higher sort concepts.
  - Pvs and CoQ allow dependent- and record types
- Differences: (serving different purposes)
  - InKa, Pvs, and CoQ support inductive definitions,
    (but by very different mechanisms and on differing levels)
  - Imps supports partial functions, theory interpretations
  - CoQ is constructive, rest classical.

The Experiment (continued)

- Formula level (system-specific representation languages)
  - Use `<OMS cd="sys" name="op">` for logical operators
  - specify the representation language of sys by `<theory id="sys">`
  - use similarity among systems to define common language cores
  - communication immediate in common fragments!
- Statement level (Def,Thm,Proof,...)
  - OMDoc1.1 sufficient for ΩMEGA, INKA, PVS, λ*Clam*, TPS, IMPS
  - `<definition>` too weak for COQ (mutuality) (OMDoc2.0)
- Theory level (Development Graph)
  - OMDoc1.1 sufficient for ΩMEGA, INKA, COQ, λ*Clam*, TPS, IMPS
  - PVS has param. theories, quantification over parameters (breaks OPENMATH)

A standardized Hierarchy of logical languages

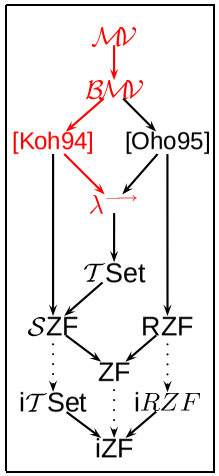
- Idea: Provide a standardized, well-documented set of "names" for logical languages



- This hierarchy is based on literal inclusion (can we wo better?)
- MBASE: Conservative Extension Principle with Logic Morphisms
  (Extend the Hierarchy with a level of proofs.)

Logic Morphisms

- Definition: Logical System $\mathcal{S} = (\mathcal{L}, \mathcal{C})$,
  - $\mathcal{L}$ language (set of well-formed formulae)
  - $\mathcal{C}$ calculus (set of inference rules)
  - $\mathcal{D}: \mathcal{H} \vdash_{\mathcal{C}} \mathbf{A}$ is a $\mathcal{C}$-derivation of $\mathbf{A}$ from $\mathcal{H}$
- Definition: Logic Morphism $\mathcal{F}: \mathcal{S} \longrightarrow \mathcal{S}'$,
  - Language Morphism $\mathcal{F}^{\mathcal{L}}: \mathcal{L} \longrightarrow \mathcal{L}'$
  - Calculus Morphism $\mathcal{F}^{\mathcal{D}}$ from $\mathcal{C}$-derivations to $\mathcal{C}'$-derivations, such that for any $\mathcal{C}$-derivation $\mathcal{D}: \mathcal{H} \vdash_{\mathcal{C}} \mathbf{A}$, we have $\mathcal{F}^{\mathcal{D}}(\mathcal{D}): \mathcal{F}^{\mathcal{L}}(\mathcal{H}) \vdash_{\mathcal{C}'} \mathcal{F}^{\mathcal{L}}(\mathbf{A})$.
- Logic morphisms transport proofs!



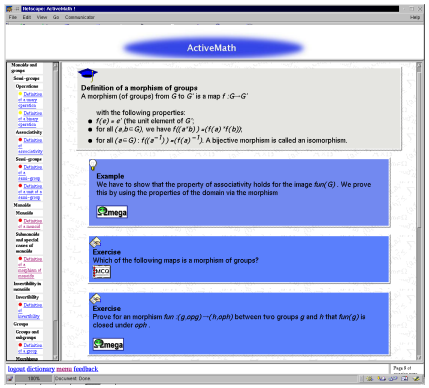
OMDoc/MBASE in Web-Based Education

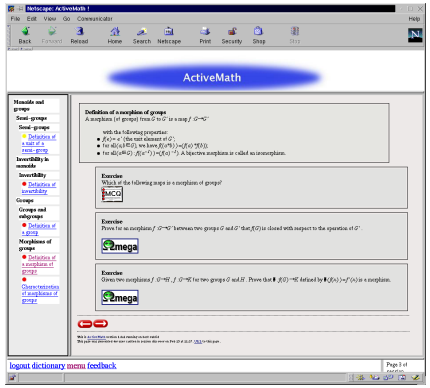
- Formal representation in OMDoc/MBASE is a lot of work, but: can be the basis for added-value services that offer
  - Efficient retrieval/searching of content (via MBASE)
  - Reuse of content, use of other external systems
- dynamic generation of mathematical documents depending on preferences and abilities of the user
  - Guided tour for a concept (explicit structure)
  - Overview on a certain subject
  - Preparation for exam (examples, problems present)
  - Follow/Review a predefined course (complex document structure)

Experiment: ACTIVEMATH/CCaps, Saarbrücken/CMU

- Content: OMDoc representation of course material
  - e.g. Chapter 6 of Algebra Interactive! ([Cohen et al. 1999])
  - Course capsules (Modeling undergraduate Comp Sci at CMU)
  - In2Math, MISS, MILCA (Math, Formal Methods, Comp. Ling)
  - OPENMATH Content Dictionaries (add didactic knowledge)
- ACTIVEMATH: personalized, flexible, presentation of knowledge
  - Guided Tours: z.B. transitive hull of the dependency relation
    (Hi, I am Michael, I am interested in the fundamental theorem of algebra)
  - Ext. systems: e.g. the ΩMEGA-System for proof exploration
  - user model: learns about user skills and preferences
    (references object structure)

Presentation for Novices and Advanced





- what can be adapted? (via user model)
  - content selection, examples, exercises; availability of systems
  - presentation: level of abstraction, notation, language and modality

Conclusions

- A mathematical infrastructure based on communication and knowledge
  - MATHWEB as a framework for distributed web-based math services
  - MBASE as a knowledge base with added-value services
  - OMDoc standardized communication languages (based on XML)
- gives us the beginning of a mathematical assistant
  - integration of external software systems (make use of the Internet)
  - a common ontology for integration (is zero a natural number?)
  - a universal math repository (data mining, semantical search)
  - bookkeeping in proofs and theories (management of change)
- Summary: A first stab at AUTOMATH/QED-NT (Emphasis on Technology)
  - You are invited to help with/share MATHWEB (http://www.mathweb.org)

3 Andreas Meier, Volker Sorge
Saarland University, Saarbrücken, Germany

Tutorial: The ΩMEGA System

System Demonstration

The ΩMEGA Group
Presentation: Andreas Meier[1] & Volker Sorge[2]

[1] University of Saarbrücken, Germany
[2] University of Birmingham, UK

Overview

I. INTRODUCTION

- Aims and Philosophy
- Background
- System

Aims

Goal: Mathematical Assistant System for proof development

- human-oriented
- abstract (top-down)
- knowledge-based
- mixed-initiative

Current status: implementation as a joint research platform for a collection of related and integrated research projects

Philosopy

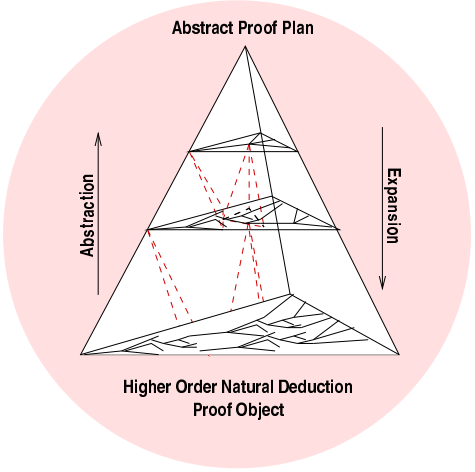
- 'Top-down' approach to theorem proving
  - Proof construction with abstract steps
  - Expansion onto a basic logic level
  - Proof checking in a small ND calculus

⇒ Proving and expansion are equivalent problems

- Do not re-invent the wheel!
  (i.e., use existing technology)

Proof Object

Proof Data Structure of ΩMEGA



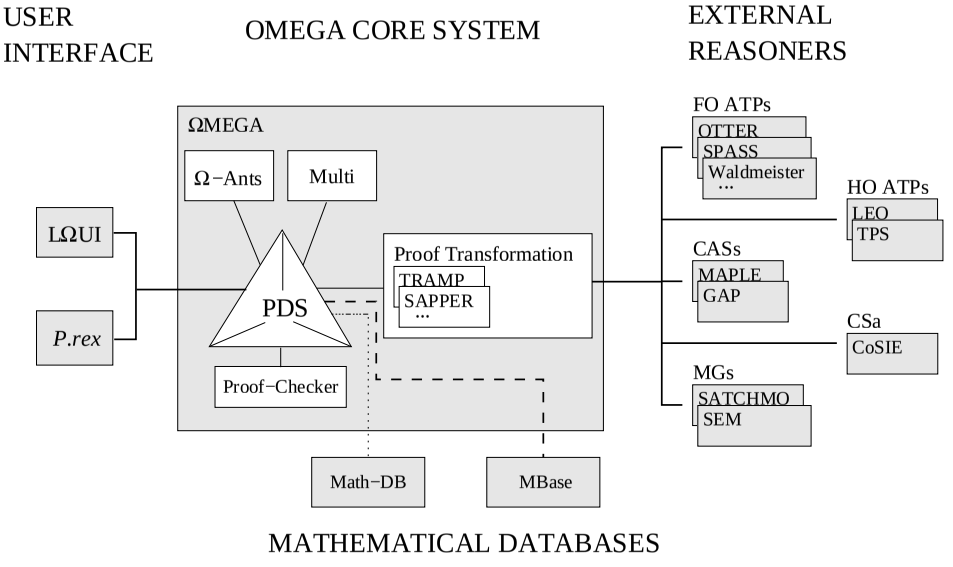
maintains simultaneously
a proof at
different levels of abstraction

Higher-order language
based on a
simply-typed λ-calculus

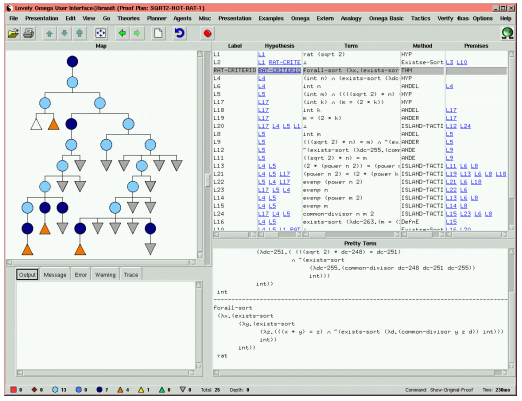
Proof Construction

- Interactive proof construction with
  - 'failing' tactics (correctness not guranteed, unlike LCF)
  - embedded external reasoners (ATP, CAS, . . .)
  - facts from knowledge base
- Usable for both proving and expanding
- Automation via proof planning and agent mechanism

Architecture



$\mathcal{LOUI}$: Graphical User Interface



multi modal:
proof tree
linearized proof
term browser

II. Interactive Theorem Proving

- Applying rules and tactics
- Using suggestion mechanism Ω-ANTS
- Proof construction with islands
- Proof expansion and proof explanation

The $\sqrt{2}$-Problem

Theorem: $\sqrt{2}$ is irrational.

Proof: (by contradiction)

Assume $\sqrt{2}$ is rational, that is, there exist natural numbers m, n with no common divisor such that $\sqrt{2} = m/n$. Then $n\sqrt{2} = m$, and thus $2n^2 = m^2$. Hence m^2 is even and, since odd numbers square to odds, m is even; say $m = 2k$. Then $2n^2 = (2k)^2 = 4k^2$, that is, $n^2 = 2k^2$. Thus, n^2 is even too, and so is n. That means that both n and m are even, contradicting the fact that they do not have a common divisor.

Formalization

The Problem:

```
(th~defproblem sqrt2-not-rat
  (in real)
  (conclusion (not (rat (sqrt 2))))
  (help "sqrt 2 is not a rational number."))
```

Formalization

SQRT:

```
(th~defdef sqrt
  (in real)
  (definition
    (lam (x num)
       (that (lam (y num) (= (power y 2) x)))))
  (help "Definition of square root."))
```


Formalization

Rat-Criterion:

```
(th~deftheorem rat-criterion
  (in rational)
  (conclusion
    (forall-sort (lam (x num)
     (exists-sort (lam (y num)
      (exists-sort (lam (z num)
       (and (times x y) z)
            (not (exists-sort (lam (d num)
                    (common-divisor y z d))
                  int))))
      int)) int)) rat))
  (help "x rational implies there exist integers y,z
         which have no common divisor with x=y*z."))
```


Interactive Proof Construction

Successive proof construction by

- applying rules
- applying tactics
- using external systems
- using facts from the database

Problems:

- Which facts are needed from database?
- Which rules/tactics/external systems are applicable? How are they applicable?

Suggestion Mechanism Ω-ANTS

Goal:

- Compute possible next proof step
- Consider rules, tactics, external systems, theorems etc.
- Suggest commands + parameters to the user

Realization:

- Realized in concurrent processes
- Computations in the background
- Exhibits anytime behavior

Proof Construction with Islands

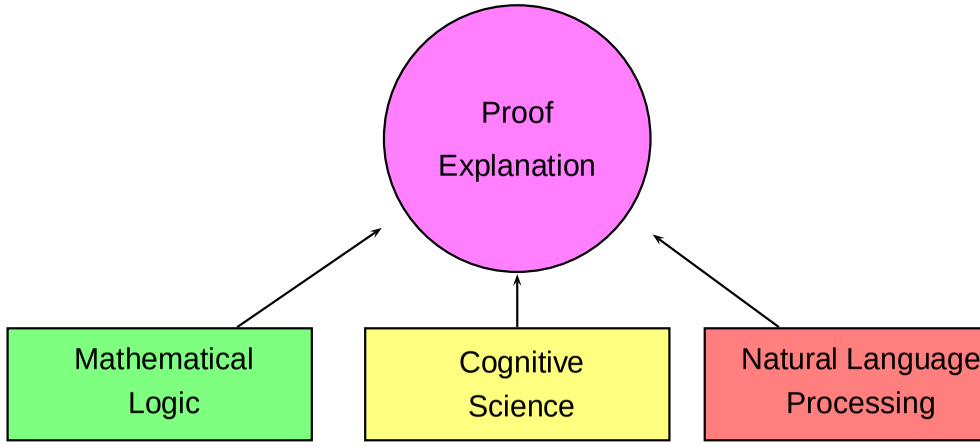
Problem: How do we get the desired proof, if the tactics do not corrspond to the necessary steps?

Insert steps as proof 'islands'

$$\dfrac{\dfrac{2 * n^2 = m^2}{Even(m^2)}\ Island}{Even(m)}\ Island$$
$$\vdots$$

Valid proof is generated by expansion.

P.rex: An Interactive Proof Explainer



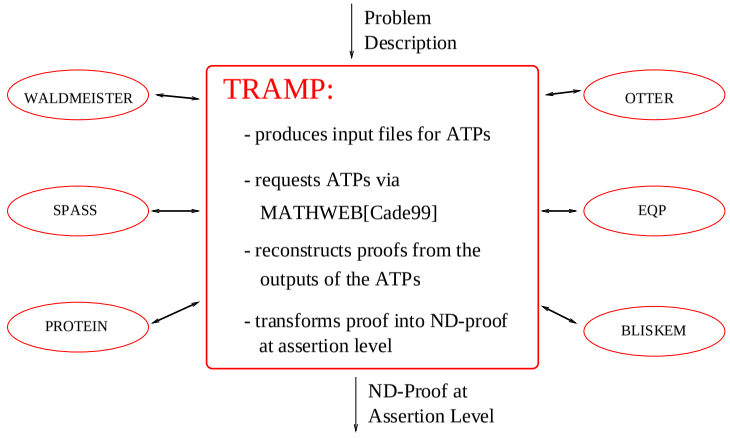
Expansion of Proofs

Recursive and interleaved process:

- Normal 'failing' tactics
  - expansion (hopefully) automatic
- Island Tactic
  - manually construct expansion
  - close gap with external reasoners
- External reasoners
  - compute automatically expansions with special systems/interfaces (e.g. TRAMP, SAPPER, . . .)

The TRAMP System

Transformation of ATP output into ND-proofs at assertion level

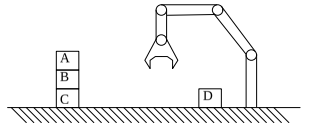


III. Proof Planning

- Automated theorem proving at abstract level
- AI planning paradigm
- Knowledge Acquisition

AI Planning

- Initial State



$on(A,B)$, $on(B,C)$, $on\_table(C)$, $on\_table(D)$, $free(A)$, ...

- Goal: $on\_table(B)$
- Operators

PUTDOWN(X)
prec: $holding(X)$
effect:
⊕ $on\_table(X)$, $hand\_empty$
⊖ $holding(X)$

- Plan $pick(A), putdown(A), pick(B), putdown(B)$

Theorem Proving as Planning

Initial State: proof assumptions
Goal: theorem
Operators: called methods

- Methods representing (abstract) proving steps
- Method = tactic + specification

Proof Plan: sequence of actions, i.e., instantiated methods
Planning Process: precondition achievement planning

Proof Planning in ΩMEGA

Knowledge-based proof planning

- domain-specific methods
- use of domain-specific external systems
- control-rules prune search space in particular domains

with multiple strategies

- strategies define different plan refinements (e.g. supply planner with different sets of methods and control rules)
- flexible combination and interleaving of strategies

Proof Planning the $\sqrt{2}$-Problem

Knowledge acquisition, e.g.

PrimeFacs-Product-m-f

prec.	L: $n * t = t'$
appl.-cond.	$n = p_1 * \ldots * p_n$ (CAS)
effect:	$\oplus$ L_1: prime-divisor(p_1,t') (Expand-PrimeFacs(L))
	$\vdots$
	$\oplus$ L_n: prime-divisor(p_n,t') (Expand-PrimeFacs(L))

Methods include specification of expansion scheme

Case Studies

- Limit Theorems

  (employs COSIE, MAPLE)
- Residue Class Domain

  (employs MAPLE, GAP, WALDMEISTER, SEM, HR)
- Homomorphism Theorems

Only with older version of ΩMEGA:

- Diagonalization proofs
- Completeness of resolution calculi

IV. Exploration

- Motivation
- Exploration with Oants
- Exploration with Multi

Motivation

Do not only re-prove known theorems, but

- Experiment with new conjectures
- Explore properties of structures

$\Rightarrow$ Postulate, prove, and refute conjectures

Set Problems

Examine different arbitrary set equations:

$$A \cup (B \cap C) = (A \cup B) \cap C$$
$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

Show validity or invalidity

Exploring with Ω-ANTS

- Consider some tactics and external reasoners
- Automated application of suggestions
- Involve complementary reasoning specialists
  - prove with automated theorem prover
  - refute with model generator
- Set examples: Proofs are simplified with some tactics and concluded by external reasoners

The Residue Class Domain

What kind of algebraic structures are ...

$$(\mathbb{Z}_4, \bar{+}), \quad (\mathbb{Z}_5, (x \bar{*} y) \bar{+} \bar{1}_5), \quad (\{\bar{0}_6, \bar{2}_6, \bar{4}_6\}, (x \bar{+} x) \bar{+} (y \bar{+} y))$$

...?

Successively check properties such as associativity, unit element, etc.

Which structures are isomorphic ?

$$(\mathbb{Z}_3, \bar{+}) \cong (\{\bar{0}_6, \bar{2}_6, \bar{4}_6\}, \bar{+}) \quad (\mathbb{Z}_2 \otimes \mathbb{Z}_2, \bar{+} \otimes \bar{+}) \ncong (\mathbb{Z}_4, \bar{+})$$

Exploration with Proof Planning

Examine properties step-by-step

- Exploration module employs Multi
  - Several strategies in Multi
  - Supported by CAS, Model Generator
- Proof plan a conjecture or its negation
- Guidance by example computation

Credits

http://www.ags.uni-sb.de/~omega

Jörg Siekmann, Christoph Benzmüller, Vladimir Brezhnev, Lassaad Cheikhrouhou, Armin Fiedler, Andreas Franke, Helmut Horacek, Michael Kohlhase, Andreas Meier, Erica Melis, Markus Moschner, Immanuel Normann, Martin Pollet, Volker Sorge, Carsten Ullrich, Claus-Peter Wirth, Jürgen Zimmer

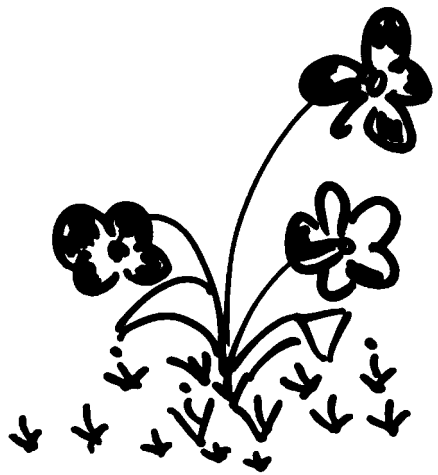
4 Jörg Siekmann, Christoph Benzmüller, Andreas Meier
Saarland Univ. and DFKI, Saarbrücken, Germany

Course: Introduction to Deduction Systems

Deduction Systems

and

Automated Reasoning



Chris Benzmüller
Jörg Siekmann

Calculemus

Pisa, September 2002

> It Is Reasonable To Expect
> That The Relationship Between
> Computation And Mathematical Logic
> Will Be As Fruitful In The Next
> Century As That Between Physics
> And Analysis In The Past.
>
> J. McCarthy, 1963

PART I:

MOTIVATION:

What is Logic?

Was ist Logik

....ist doch logisch, oder?

Was Sind Logische Schlussweisen ?

~ Luftreise? Verfahren

1. Abstraktions Schritt:

Aussagen:

A: = "Paris ist die Hauptstadt von Frankreich"

B: = "Mäuse jagen Elephanten"

falsch?
wahr?

$A \wedge B \overset{?}{\equiv} B \wedge A$

A: = "Otto wird krank"

B: = "Der Arzt verschreibt Otto eine Medizin"

→ o Verknüpfungen sind ebenfalls Abstraktionen

Beispiel: || Anomalie der materiellen Implikat

Deduktion

„Alle Katzen fressen Fische“
„Garfield ist eine Katze“

„Garfield frißt Fische“

„Alle Menschen sind sterblich“
„Sokrates ist ein Mensch“

„Sokrates ist sterblich“

„Alle 🐰 haben Eigenschaft ✿“
„🐞 ist ein 🐰“

„🐞 hat Eigenschaft ✿“

Abduktion

„Wenn es regnet, dann ist die Straße naß”
„Die Straße ist naß”

(Wahrscheinlich) „Hat es geregnet”

„Wenn 🐰 dann ❀”
„Es gilt ❀”

„Dann nehmen wir an 🐰”

WESENTLICHER UNTERSCHI
ZUR
DEDUKTION

Induktion

Fakt1: "Ein Stromkreis hat 3 Ohm Widerstand
und es fließen 2 Ampère Strom"
Beobachtung: 6 Volt Spannung

Fakt2: "Ein Stromkreis hat 10 Ohm Widerstand
und es fließen 3 Ampère Strom "
Beobachtung: 30 Volt Spannung

Induktive Schlußfolgerung: "Die Spannung ist
das Produkt aus Strom und Widerstand"
$U = R \bullet I$

- Vollständige (mathematische) Induktion
- Strukturelle Induktion

ALS SPEZIALFÄLLE

Nichtmonotones Schließen

Klassische Logik ist monoton:

Wenn: „ Aus ϕ folgt FAKTUM"
Dann: „Aus $\phi \cup \Psi$ folgt FAKTUM"

Aber:

„Alle Vögel (bis auf Pinguine) können fliegen"
„Tweety ist ein Vogel"

„Tweety kann fliegen"

!

„Alle Vögel (bis auf Pinguine) können fliegen"
„Tweety ist ein Vogel"
„Tweety ist ein Pinguin"

„Tweety kann nicht fliegen"

Probabilistisches Schließen

Wenn:

„Dozent hat Fieber"
„Dozent läuft die Nase"
„Dozent hustet"
„Dozent hat gerötete Augen"
„Dozent hat eine lausige Laune"

Dann:

„Dozent ist (wahrscheinlich) erkältet"

Folgerung:

„Nächste Vorlesungsstunde bleiben wir besser gleich zu Hause"

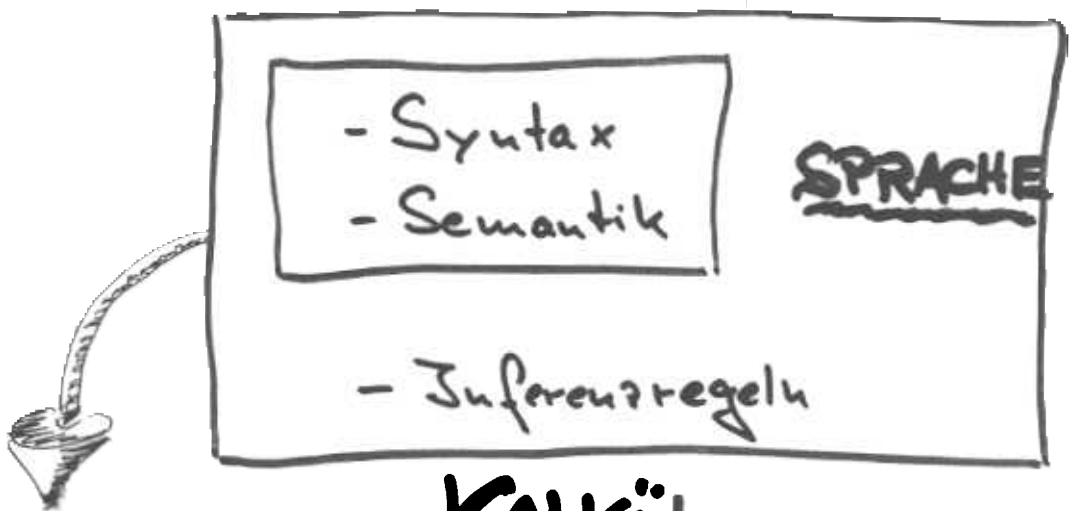
Schliessen Und Folgern:

- Deduktion
- Abduktion
- Induktion
- Nichtmonotones Schliessen
- Probabilistisches Schliessen

- Deduktionssysteme
- Deduktive Datenbanken
- Expertensysteme
- Inferenzsysteme (für WB
- Logisches Programmieren

Welche Schluss-Weisen sind hinreichend formalisiert?

BAUSTEINE EINER LOGIK:



KALKÜL

- positive Kalküle
- negative Kalküle
- Widerlegungskalkül
- natürliches Schliessen

"BEWEIS THEORIE" (proof theory)

"MODELL THEORIE" (model theory)

→ META-RESULTS

PART II:

FIRST ORDER PREDICATE CALCULUS

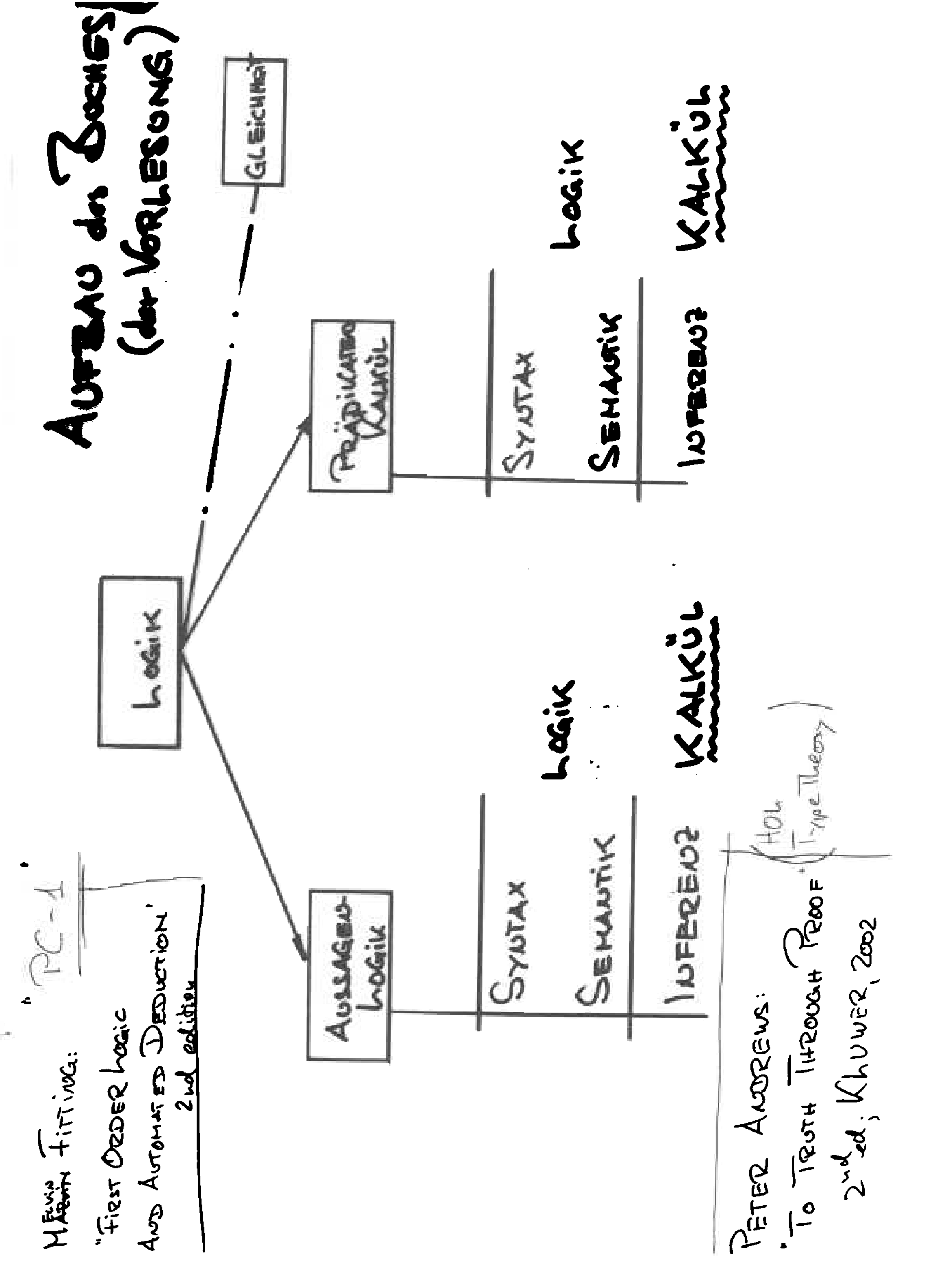
→ A: Logic, Syntax, Semantics

B: A Calculus: Inference Rules

PRÄDIKATENKALKÜL ERSTER ORDNUNG

P.K.1.

> It is reasonable to expect that the relationship between computation and mathematical Logic will be as fruitful in the next century as that between pysics and analysis in the Last
>
> J. McCarthy, 1963



LEVEL 1: THE LOGICAL LANGUAGE OF A DEDUCTION SYSTEM

- FIRST ORDER PREDICATE LOGIC:

  Constants: $a, b, c, \ldots$ Variables: $x, y, z, \ldots$

  Functions: $f, g, h, \ldots$ $f(x_1, x_2, \ldots, x_n)$

  Predicates: $P, Q, R, \ldots$ $P(x_1, x_2, \ldots, x_n)$

  Junctors: $\wedge, \vee, \neg, \Rightarrow$

  Quantifiers: $\forall, \exists$

- BUILT IN'S:

  Equality, Sets, Arithmetic ..

- SORTS:
  - many sorted Logic
  - order sorted Logic
  - Polymorphism, Overloading

2.1 Aussagenlogik: Syntax

- Ausagenlogische Variable: $P_1, P_2, P_3, \ldots\ldots$

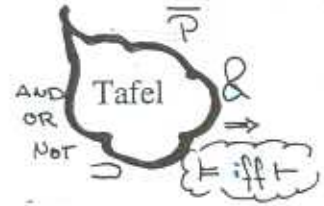
  (propositional letter)

  Konvention: $P, Q, R, \ldots$

- logische Verknüpfungen: $\wedge, \vee, \neg, \subset, \uparrow, \downarrow, \equiv, \not\equiv$

  (logical connectives)

  Konvention:
  - binäres Symbol " $\circ$ "
  - andere Schreibweisen



- atomare Formel: P_i oder $\bot$ oder $\top$
- aussagenlogische Formel:

  Konvention: $((Q \supset R) \vee P) \supset R$

  ➡ DEFINITIONS-SCHEMA (rekursiv)

TAFEL

Beispiele (für Aussagenlogische Formeln)

$$P \wedge Q \supset P \vee Q$$

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$

$$(P \supset (Q \supset R)) \equiv ((P \supset Q) \supset (P \supset R))$$

etc

TAFEL

Rekursives Definitionsschema für Formeln

Definition: Die Menge der aussagenlogischen Formeln ist die kleinste Menge P so daß:

1. Wenn A eine aussagenlogische Variable ist, dann $A \in P$
2. Wenn $X \in P$ dann $\neg X \in P$
3. Sei $\circ$ eine binäre Verknüpfung, wenn $X, Y \in P$ dann $(X \circ Y) \in P$

→ Def. kleinste Menge

→ Eindeutigkeit

Beispiel: Terme!

Prinzip der Strukturellen Induktion

Definition: Jede aussagenlogische Formel hat die Eigenschaft Q, vorausgesetzt

Basis:
- jede atomare Formel hat die Eigenschaft Q

Induktion:
- wenn X die Eigenschaft Q hat, so auch $\neg X$
- wenn X, Y die Eigenschaft Q haben, so auch $(X \circ Y)$

$\Rightarrow$
- als Beweismethode
- als Theorem über aussagenlogische Formeln

ACHTUNG: Beweisbarkeit ?

z. B. Peano Axiome

Hol

Tafel

DUALITÄT

Definition: Seien $\circ$ und $\bullet$ binäre Verknüpfungen. Dann ist $\circ$ dual zu $\bullet$, wenn

$$\neg(X \circ Y) = (\neg X \bullet \neg Y)$$

Frage: Welche dualen Verknüpfungen gibt es

Beweis ?

2.2 Aussagenlogik: SEMANTIK

→ **Definition:** Die Menge der Wahrheitsswerte ist

$$Tr = \{ t, f \}$$

→ **Bemerkung:** Unterschied zwischen t, f und $\top$, $\bot$

→ Logische Verknüpfungen als Funktionen:

$$\circ: Tr \times Tr \to Tr.$$

(FREGE)

a) einstellige Wahrheitsfunktionen:

$$id: Tr \to Tr \qquad id(t) = t,\ id(f) = f$$

$$\neg: Tr \to Tr \qquad \neg(t) = f,\ \neg(f) = t$$

DISKURS:
- materielle Implikation
- minimale Menge
- maximale Menge

b) Zweistellige Wahrheitsfunktionen: ($4^2 = 16$ Funktionen)

NAND
NOR

		Primary								Secondary	
		$\wedge$	$\vee$	$\supset$	$\subset$	$\uparrow$	$\downarrow$	$\not\supset$	$\not\subset$	$\equiv$	$\not\equiv$
t	t	t	t	t	t	f	f	f	f	t	f
t	f	f	t	f	t	t	f	t	f	f	t
f	t	f	t	t	f	t	f	f	t	f	t
f	f	f	f	t	t	t	t	f	f	t	f

TABLE 2.1. Primary and Secondary Connectives

Bool´sche Auswertung

Definition: Eine Bool´sche Valuation ist eine Abbildung

$$\nu: P \to Tr \qquad \text{mit}$$

1. $\nu(\top) = t$, $\nu(\bot) = f$
2. $\nu(\neg X) = \neg\nu(X)$
3. $\nu(X \circ Y) = \nu(X) \circ \nu(Y)$ gemäß Tabelle 2.1

Satz: Für jede Abbildung f aus der Menge der Formeln in die Menge der Wahrheitswerte existiert eine Bool´sche Valuation die mit f auf den aussagenlogischen Variablen übereinstimmt.

Satz: Wenn ν_1 und ν_2 auf den aussagenlogischen Variablen S übereinstimmen, dann stimmen sie auf allen Formeln, die aus S gebildet werden überein.

Beispiel:

Tafel

$((P \uparrow \neg Q) \supset R)$

mit $\nu(P) = t$
$\nu(Q) = f$
$\nu(R) = f$

TAFEL 2

Beispiel: $\nu(P) = t$
$\nu(Q) = f$
$\nu(R) = f$

$\nu((P \uparrow \neg Q) \supset R) = \nu(P \uparrow \neg Q) \supset \nu(R)$

$= (\nu(P) \uparrow \nu(\neg Q)) \supset \nu(R)$

$= (\nu(P) \uparrow \neg \nu(Q)) \supset \nu(R)$

$= (t \uparrow \neg f) \supset f$

$= t \uparrow t \supset f$

$= f \supset f$

$= t$

Beispiel: $(P \vee Q) \supset (P \vee Q)$

$\nu(P) = t$
$\nu(Q) = f$

alle anderen!

↓ Vereinfachen

Übung

Begriff: TAUTOLOGIE

TAFEL

Erfüllbar (satisfiable)

Unerfüllbar (unsatisfiable)

- valid, tautologisch (validity)
- falsch ~ (false)

2.4 SMULLYAN'S NOTATION

(uniform notation)

Definition: Gruppiere alle binär verknüpften Formeln in

α- Formeln: wenn sie konjunktiv wirken.

Komponenten: α_1 , α_2

β- Formeln: wenn sie disjunktiv wirken.

Komponenten: β_1 , β_2

Conjunctive			Disjunctive		
α	α_1	α_2	β	β_1	β_2
$X \wedge Y$	X	Y	$\neg(X \wedge Y)$	$\neg X$	$\neg Y$
$\neg(X \vee Y)$	$\neg X$	$\neg Y$	$X \vee Y$	X	Y
$\neg(X \supset Y)$	X	$\neg Y$	$X \supset Y$	$\neg X$	Y
$\neg(X \subset Y)$	$\neg X$	Y	$X \subset Y$	X	$\neg Y$
$\neg(X \uparrow Y)$	X	Y	$X \uparrow Y$	$\neg X$	$\neg Y$
$X \downarrow Y$	$\neg X$	$\neg Y$	$\neg(X \downarrow Y)$	X	Y
$X \not\supset Y$	X	$\neg Y$	$\neg(X \not\supset Y)$	$\neg X$	Y
$X \not\subset Y$	$\neg X$	Y	$\neg(X \not\subset Y)$	X	$\neg Y$

TABLE 2.2. α and β Formulas and Components

Satz:

(1) Für alle Bool´schen Valuationen und alle α- bzw. β- Formeln:

$$v(\alpha) = v(\alpha_1) \wedge v(\alpha_2)$$
$$v(\beta) = v(\beta_1) \vee v(\beta_2)$$

(2)

$$\alpha \equiv (\alpha_1 \wedge \alpha_2)$$
$$\beta \equiv (\beta_1 \vee \beta_2)$$

Beweis:

2.6 NORMALFORMEN

Definition: Ein Literal ist eine atomare Formel oder deren Negation oder $\top$ oder $\bot$

Beispiele: Q, $\neg Q$, $\bot$, ...

Definition: Eine aussagenlogische Formel ist in konjunktiver Normalform (KNF, Klauselform), wenn sie eine Konjunktion von Disjunktionen von Literalen ist.

Beispiele: $L_1 \vee L_2 \vee L_3 \vee .... \vee L_n \wedge$

$M_1 \vee M_2 \vee M_3 \vee .... \vee M_m \wedge$

$K_1 \vee K_2 \vee K_3 \vee .... \vee K_k$

Definition: Eine aussagenlogische Formel ist in disjunktiver Normalform (DNF), wenn sie eine Disjunktion von Konjunktion von Literalen ist.

Definition: Eine aussagenlogische Formel ist in Negationsnormalform, wenn sie

1. nur $\wedge$, $\vee$ oder $\neg$ als Verknüpfungen enthält
2. Alle Negationen so weit, wie möglich nach innen gezogen sind.

-3: CALCULUS

- (classical) HilbertCalculus
- a machine oriented logic

HILBERT-KALKÜL:

KLASSISCHER AUFBAU EINES KALKÜLS

BEISPIEL:

- SYNTAX
- SEMANTIK
- LOGISCHE AXIOME (SCHEMA)

(i) $\alpha \Rightarrow (\beta \Rightarrow \alpha)$

(ii) $\alpha \Rightarrow (\beta \Rightarrow \gamma) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma))$

(iii) $(\alpha \Rightarrow \beta) \Rightarrow (\neg\beta \Rightarrow \neg\alpha)$

- INFERENZREGELN

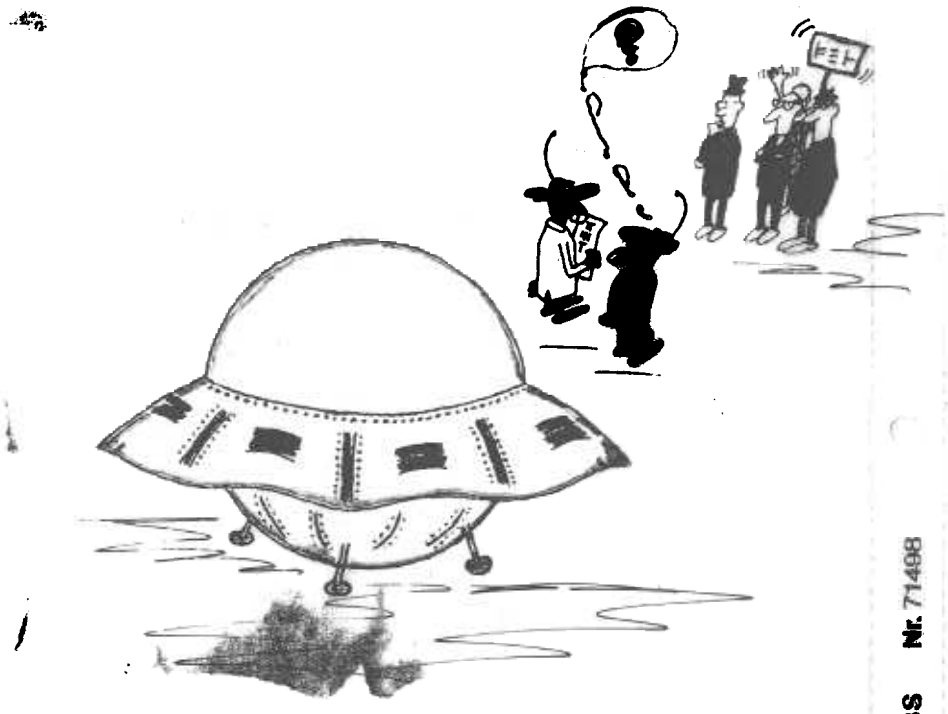
MODUS PONENS:

$$\frac{\alpha \qquad \alpha \Rightarrow \beta}{\beta}$$

INSTANTIIERUNG:

$$\frac{\forall x.\, P(x)}{P(a)}$$

HILBERT'S PROGRAM



IBS Nr. 71498

A Classical Deduction System

Wie Funktioniert Ein Deduktions-System

→ The Traditional Machine Oriented Paradigm

How Does It Work?

McCarthy 1962:

"... and the computer system has stored in it the theorems which have previously been proved. In addition, there are a number of techniques embodied in programs for generating proofs. The mathematician expresses his ideas of how a proof may be found by combining these techniques into a program. The computer carries out the program which may prove the theorem, may generate information that will guide another try, may indicate an elementary misconception, or be of no help whatsoever."

LCF-Paradigma

"... to provide an **interactive metalanguage** for conducting proofs, **in which in principle** almost every style can be programmed, but which provides the **greatest** security **against** faulty proofs"

3.2. RESOLUTION

Definition: Seien $X_1 = L_1 \vee L_2 \vee \ldots\ldots \vee L_n$

$X_2 = K_1 \vee K_2 \vee \ldots\ldots \vee K_m$

Zwei Klauseln (Disjunktionen) mit $\neg L_1 = K_1$

Dann ist

RESOLVENTE: $X = L_2 \vee L_3 \vee \ldots\ldots \vee L_n \vee K_2 \vee K_3 \vee \ldots\ldots \vee K_m$

eine binäre Resolution von X_1 und X_2.

Wenn $L_i = \bot$, dann ist

$X_i' = L_1, L_2, \ldots, L_{i-1}, L_{i+1}, \ldots, L_n$

eine triviale Resolution von X_1.

Beispiel: Tafel

GENERALISIERUNG

➡ Modus Ponens Zusammenhang

Ein kompletter Resolutionsbeweis

3.4.11 Beispiel Transitivität der Teilmengenrelation.

Definition von $\subseteq$ in Termini von $\in$:

$$\forall x,y \quad x \subseteq y \Leftrightarrow \forall w \; w \in x \Rightarrow w \in y$$

Theorem $\forall x,y,z \; x \subseteq y \wedge y \subseteq z \Rightarrow x \subseteq z$

Klauselform:

H1: $\neg x \subseteq y, \neg w \in x, w \in y$	($\Rightarrow$ Teil der Definition)
H2: $x \subseteq y, f(x,y) \in x$	(zwei $\Leftarrow$ Teile der Definition,
H3: $x \subseteq y, \neg f(x,y) \in y$	f ist die Skolem Funktion für w)
C1: $a \subseteq b$	(drei Teile der negierten Behauptung,
C2: $b \subseteq c$	a, b, c sind Skolem Konstanten für x, y, z)
C3: $\neg a \subseteq c$	

Resolutionswiderlegung:

H1,1 & C1, $\{x \mapsto a, y \mapsto b\}$	$\rightarrow$ R1: $\neg w \in a, w \in b$
H1,1 & C2, $\{x \mapsto b, y \mapsto c\}$	$\rightarrow$ R2: $\neg w \in b, w \in c$
H2,2 & R1,1, $\{x \mapsto a, w \mapsto f(a,y)\}$	$\rightarrow$ R3: $a \subseteq y, f(a,y) \in b$
H3,2 & R2,2, $\{y \mapsto c, w \mapsto f(x,c)\}$	$\rightarrow$ R4: $x \subseteq c, \neg f(x,c) \in b$
R3,2 & R4,2, $\{x \mapsto a, y \mapsto c\}$	$\rightarrow$ R5: $a \subseteq c, a \subseteq c$
R5 (Faktorisierung)	$\rightarrow$ R6: $a \subseteq c$
R6 & C3	$\rightarrow$ R7: $\lightning$

PART III : UNIFICATION

- Robinson Style
- FAST(ER) UNIFICATION ALGORITHMS

Unifikation

Unifikation: Beispiel

$f(a, g(x, b))$

$f(y, g(c, z))$

~~z~~ y

~~g(c, z)~~ x

etc

⇒ Robinson-Unifikation

⇒ 2^n quadratisch

The Race for the Fastest (the Earliest) Unification Algorithm

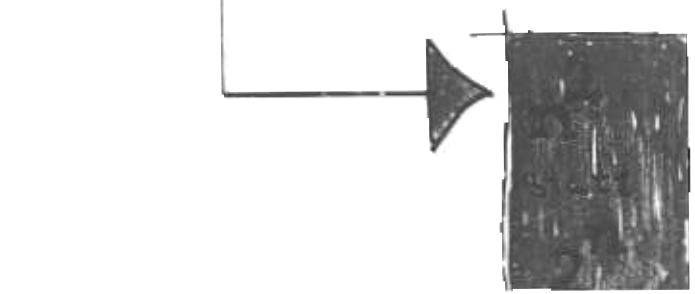
- exponential: $O(2^n)$ Robinson
- quadratic: $O(n^2)$ - - -
- logarithmic: $O(n \log n)$ Martelli - Montanari
- linear: $O(n)$ Paterson - Wegman
- quasi-linear: → today

The Martelli-Montanari-Algorithm

Quadratische (statt exponentieller)

(Decomposition)

Unifikation



MULTI-EQUATIONS:

$x_1 = x_2 = \ldots = x_n = t_1 = t_2 = \ldots = t_n$

represented as:

$M: \{\{ x_1,\ldots,x_n\}, \{t_1,\ldots,t_n\}\} = \{V,T\}$

where x_1 Variables und t_1 Terms.

σ solves $\{\{ x_1,\ldots,x_n\}, \{t_1,\ldots,t_n\}\}$, iff

$\sigma x_1 = \ldots = \sigma x_n = \sigma t_1 = \ldots = \sigma t_n$

BEISPIEL: $\{\{x,y\},\{f_2, f_2\}\}$

MARTELLI-MONTANARI ALGORITHM:

M. M-INPUT: $G = \{ s_1 = t_1,\ldots,s_n = t_n\}$.

(0) $s = t = \{\emptyset, \{s,t\}\}$

Regeln (indeterministisch)

(1) Merge:

$\{V_1, T_1\} \wedge \{V_2, T_2\} \Rightarrow$

$\Rightarrow \{ V_1 \cup V_2, T_1 \cup T_2\}$,

if $V_1 \cap V_2 \neq \emptyset$; i.e. some $x \in V_1$ and $x \in V_2$

(2) Decomposition:

$\{V_1, \{f(s_1,\ldots,s_n), f(t_1,\ldots,t_n)\} \cup T_1\} \wedge G$

$\Downarrow$

$\{V_1, \{f(s_1,\ldots,s_n)\} \cup T_1\} \wedge s_1 = t_1 \wedge \ldots \wedge s_n = t_n \wedge G$

$0 \leq n \in H$

(3) Occurs-in-check:

If Variables $x_1,\ldots,x_n$ and Terms $t_1,\ldots,t_n$ with
$x_1 \in V(t_2),\ldots, x_{n-1} \in V(t_n), x_n \in V(t_1)$,
$\Rightarrow$ STOP.FAIL

$V(t) = \{ x \mid x \text{ occurs in } t \}$

Abbruchbedingungen:

3) Zyklen-test (occurs-in-check)

Wenn Multigleichungen $M_1,\ldots,M_n$ existieren, Variablen $x_i \in M_i$, und Terme $t_i \in M_i$, so daß $x_i \in V(t_{i+1})$, $i=1,\ldots,n-1$ und $x_n \in V(t_1)$

4) Clash

Wenn eine Multigleichung M existiert, die zwei Terme $f(s_1,\ldots,s_n)$ und $g(t_1,\ldots,t_m)$ enthält mit $f \neq g$.

Output Of M.-M.-Algorithm:

A Set Of Multisets Of The Form:

$$M_i = \{\{x_{i,1}, x_{i,2}, \ldots, x_{i,n}\}, \{t_i\}\}$$

Compute The Most General Unifier!

Example: $\{\{x\},\{g(y)\}\}$ and $\{\{y\},\{h(z)\}\}$

mgu: $\sigma = \{x \mapsto g(h(z)),\ y \mapsto h(z)\}$

Ein Beispiel für Unifikation nach Martelli & Montanari

$\{k(f(x,g(a,y)), g(x,h(y)) = k(f(h(y),g(y,a)), g(z,z))\}$

$\rightarrow \{f(x,g(a,y)) = f(h(y),g(y,a)), g(x,h(y)) = g(z,z)\}$ (Dekomposition)

$\rightarrow \{x= h(y), g(a,y)=g(y,a), x=z, z=z\}$ (2 × Dekomposition)

$\rightarrow \{x= z = h(y), g(a,y)=g(y,a)\}$ (2 × Merge)

$\rightarrow \{x= z = h(y), a=y, y=a\}$ (2 × Dekomposition)

$\rightarrow \{x= z = h(y), y=a=a\}$ (Merge)

$\rightarrow \{x= z = h(y), y=a\}$ (Dekomposition)

EXTRAKTION

Allgemeinster Unifikator: $\{x \mapsto h(a), z \mapsto h(a), y \mapsto a\}$

Tel. 2 54 00 26
Fax 2 54 00 27
Dresden

THEOREM 3: Der Martelli-Montanari Algorithmus ist vollständig und korrekt.

Denn: jede Regel erhält U(G)
(wie in der üblichen Regeldarstellung

TERMINIERUNG

(von Martelli & Montanari)

↓

....und anderen!

MULTISETS AND MULTISET ORDERINGS

EXAMPLES MULTISETS:

$$\{3,3,3,4,0,0\} = \{3,0,0,3,4,3\}$$

$$\neq \{0,3,4\}$$

DEFINITION:

FOR A PARTIALLY ORDERED SET $(S,<)$ THE KORRESPONDING MULTISET ORDER IS DEFINED AS:

FOR MULTISETS M, M':

$M \gg M'$ iff

1) $M' = (M - X) \cup Y$ for Multisets X, Y

2) $\forall y \in Y$ there exists $x \in X$ with $x > y$

EXAMPLE:

$$\{3,3,4,0\} \gg \{3,4\}$$

$$\gg \{3,2,2,1,1,1,4,0\}$$

$$\gg \{3,3,3,3,2,2\}$$

THEOREM: A WELLFOUNDED ORDER ON A SET S DEFINES A WELLFOUNDED ORDER ON THE SET OF MULTISETS ON S.

TERMINATION OF MARTELLI-MONTANARI

COMPLEXITY MEASURE $\mu = (\mu_1, \mu_2)$

μ_1 := MULTISET OF DEPTH OF ALL (SUB-)TERM

μ_2 := NUMBER OF MULTISETS

$\mu = (\mu_1, \mu_2)$ IS LEXICOGRAPHICALLY WELL FOUNDED

THEOREM 1: THE MARTELLI-MONTANARI ALGORITHM TERMINATES.

PROOF: FOR EVERY PRODUCTION WE HAVE:

$$(\mu_1, \mu_2)_L < (\mu_1, \mu_2)_R$$

■

THEOREM 2: THE MARTELLI-MONTANARI ALGORITHM IS QUASI-LINEAR, i.e

$n \cdot \log(n)$

PART IV

DEDUCTION SYSTEMS

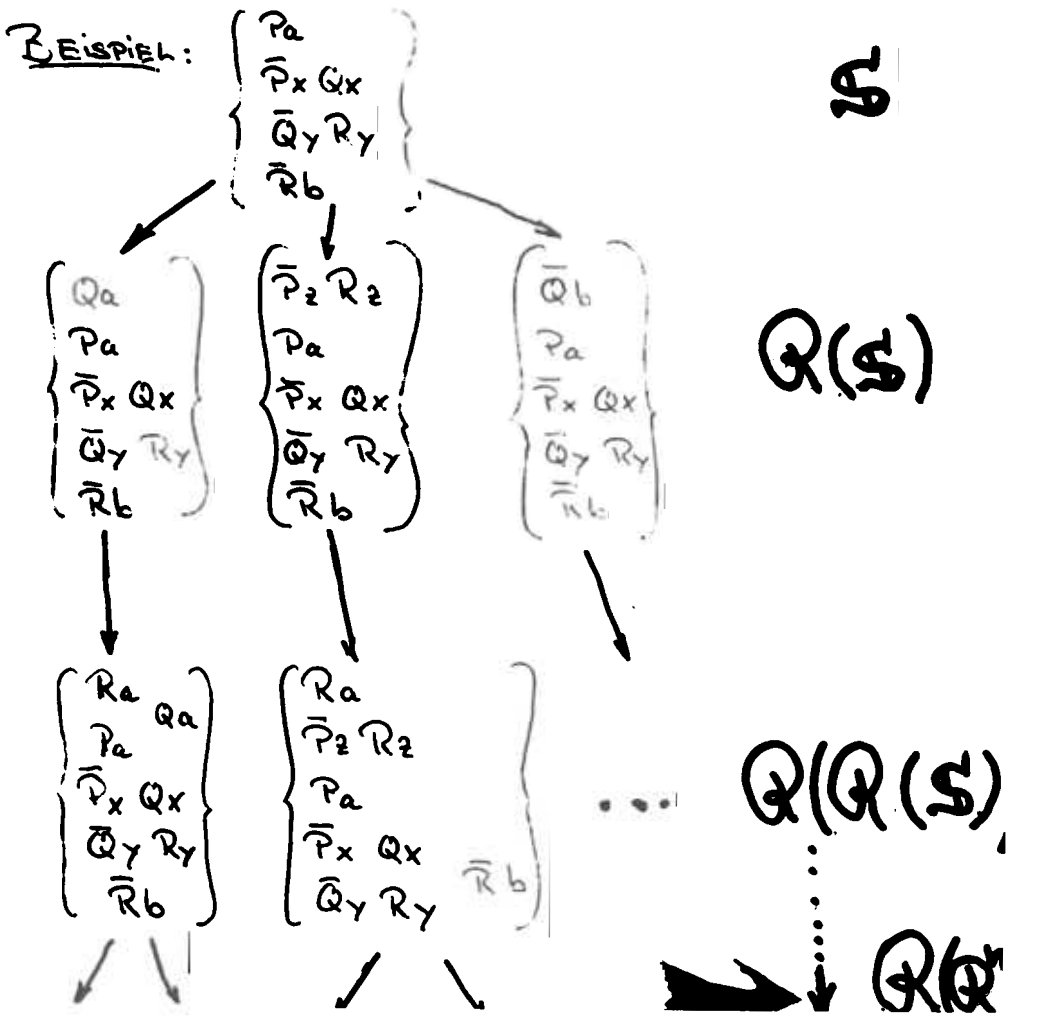
(classic systems)

LEVEL SATURATION

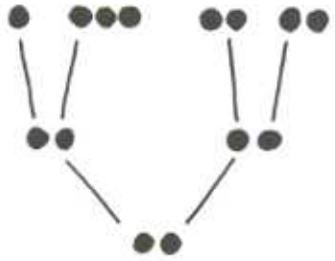
PROBLEM: • VOLLSTÄNDIGKEIT:

ES EXISTIERT EINE WIDERLEGUNG

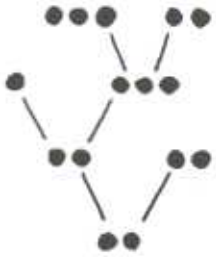
• WIE FINDET MAN SIE ?



Restriction:



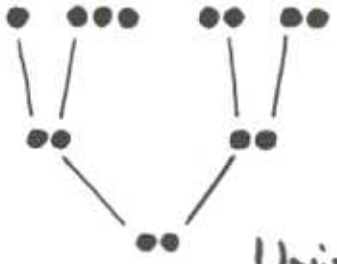
NO



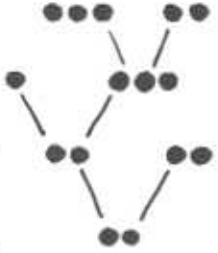
YES

INPUT

Ordering:



<



UNIT PREF.

CLASSIFICATION OF REFINEMENTS:

- RESTRICTION STRATEGIES
- ORDERING STRATEGIES

Kontrolle

Ziel: „schlechte" Schritte vermeiden
„gute" Schritte bevorzugen.

Restriktionsstrategien

Idee: Suchraumeinschränkung durch verbieten bestimmter Schritte.

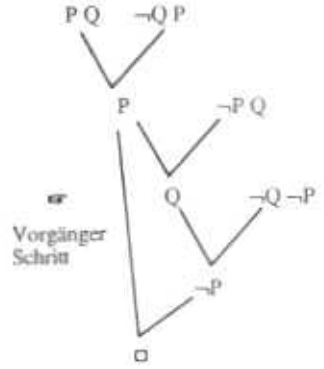
Strategie: *Set-of-Support*

SOS

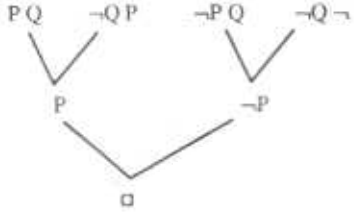
Gesamtklauselmenge = [konsistente Klauselmenge (Axiome) / Set of Support (negiertes Theorem)]

eine Elternklausel aus dem Set of Support
Resolventen → Set of Support

Beispiel: SOS = {P,Q}



SOS = {P,Q}, {¬P,Q}



SOS: Widerlegungsvollständig wenn SOS an der Widerlegung beteiligt ist.

Unit Preference

Examples:

Restriction Strategies:

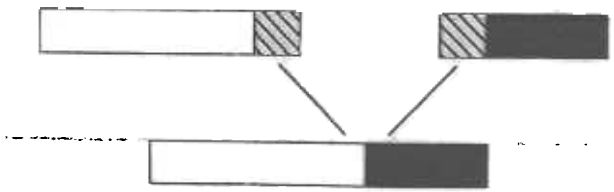
- SOS
- Unit Restriction
- Input Restriction
- Merge Restriction
- Linear Restriction
- SL-Restriction
- Factoring Restrictions

Ordering Strategies:

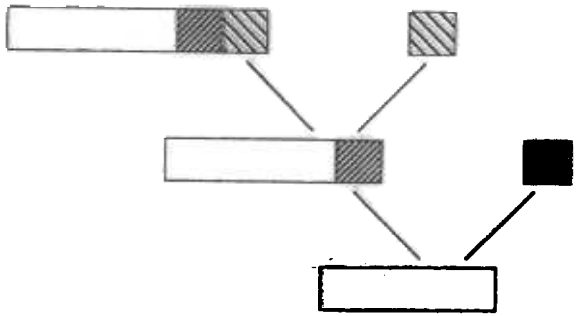
- Level Saturation
- Unit Preference
- Lock Resolution

Example: Restriction Strategy

ausgeschlossen (excluded):



zugelassen (admitted):



Unary Resolution

Example: Order Strategy

$$\frac{\begin{array}{rl} P(x,b) \vee & Q(x) \vee R(b) \vee S(c) \\ \neg P(a,y) \vee & Q(a) \vee R(y) \vee S(c) \end{array}}{Q(a) \vee R(b) \vee S(c)}$$

Resolvente kürzer als jede der Elternklauseln

Resolvent Shorter Than Each Parent

→ **Unit Preference**

EXAMPLE: ORDERING STRATEGY

$$\frac{\begin{array}{r} P(x,b) \vee Q(x) \\ \neg P(a,y) \vee R(y) \vee Q(a) \vee R(b) \end{array}}{\begin{array}{r} Q(a) \vee R(b) \vee Q(a) \vee R(b) \end{array}}$$

$$\frac{Q(a) \vee R(b) \vee Q(a) \vee R(b)}{Q(a) \vee R(b)}$$

REPLACEMENT RESOLUTION

Gesamteffekt:

ersetze eine Klausel durch eine echte Teilklausel

replace a clause by a proper subclause

REDUCTION STEP

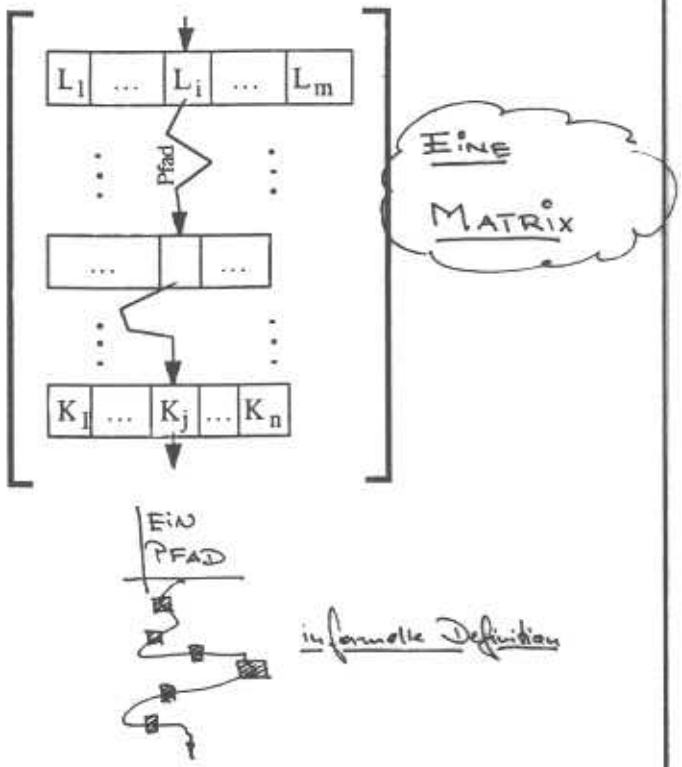
Idee: Aussagenlogische Klauselmenge:

$$((L_1 \vee \ldots \vee L_m) \wedge \ldots \wedge (K_1 \vee \ldots \vee K_n)),$$

erfüllbar gdw.
es gibt eine Interpretation, unter der in jeder Klausel mindestens ein Literal wahr ist (Pfad).

P. Andrews (CMU)

W. Bibel (Darmstadt)

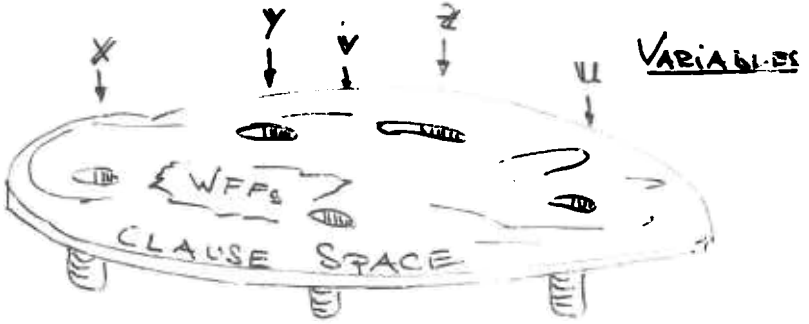


IBS Nr. 71488

Peter Andrews: (CMU)

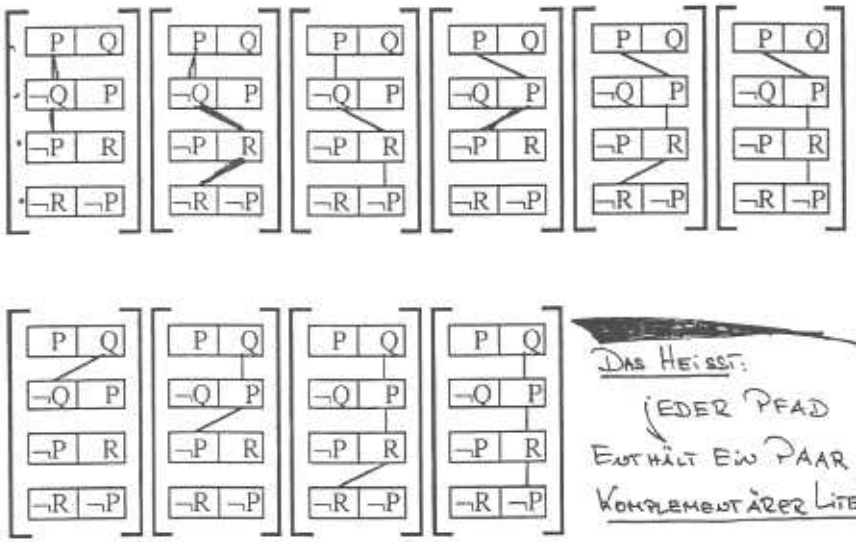
What Makes A Set Of Formulae Contradictory

4th CADE, Austin, 1979



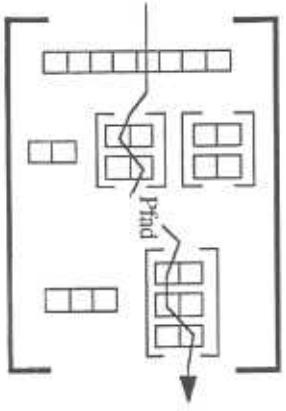
Ein Beispiel:

Test der Unerfüllbarkeit von $(P \vee Q) \wedge (\neg Q \vee P) \wedge (\neg P \vee R) \wedge (\neg R \vee \neg P)$:



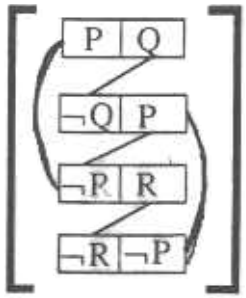
Das heißt:
Jeder Pfad enthält ein Paar komplementärer Literale

Funktioniert auch bei Nichtklauselnormalform.



Rekursive Def. von Pfad ?

Beispiel:



Konnektionen
Jeder Pfad enthält eine Konnektion

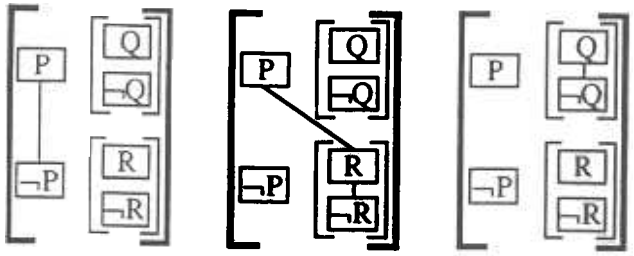
Konnektion Methode

Connection Method

Theorem: Eine Formelmenge ist genau dann unerfüllbar, wenn die zugehörige Konnektionsmatrix aufgespannt (Spaning) wird.

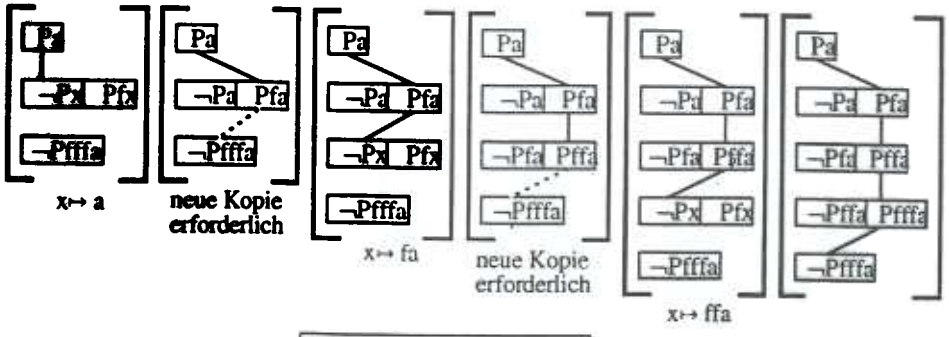
Beispiel: (NICHT KLAUSEL NORMALFORM)
$(P \vee Q) \wedge (\neg Q \vee P) \wedge (\neg P \vee R) \wedge (\neg R \vee \neg P)$ äquivalent mit
$(P \vee (Q \wedge \neg Q)) \wedge (\neg P \vee (R \wedge \neg R))$.

Das Suchverfahren benötigt jetzt nur drei Schritte:



Das Verfahren für erste Ordnung:

Beispiel:
{{P(a)}, {¬P(x), P(f(x))}, {¬P(f(f(f(a))))}}
ist unerfüllbar.

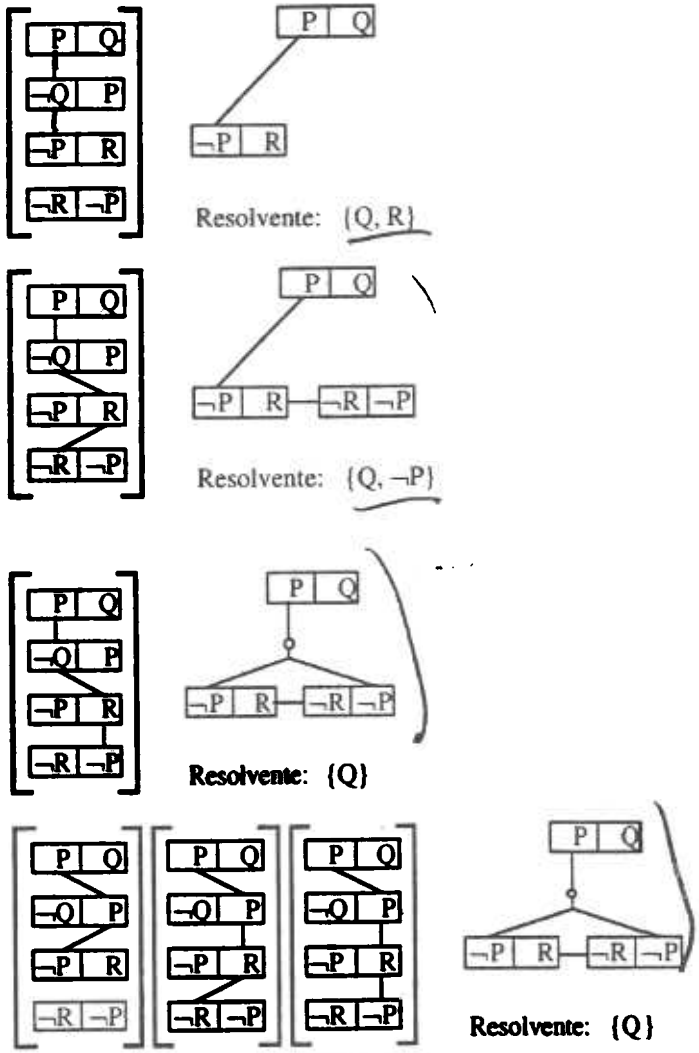


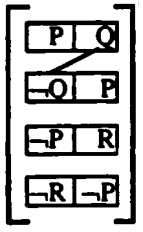
IBS Nr. 71498

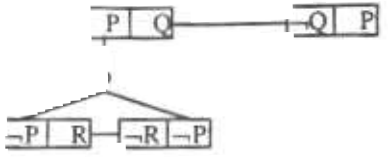
Zusammenhang zwischen Konnektions- und Resolutionsverfahren.

Pfadsuche entspricht Konstruktion eines Widerlegungsgraphen

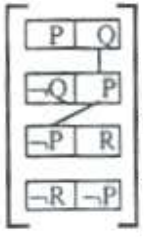
Beispiel:

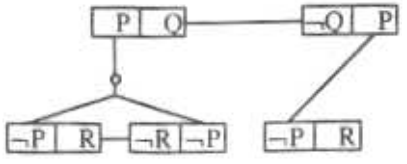




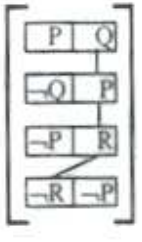


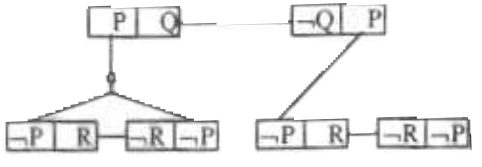
Resolvente: {P}



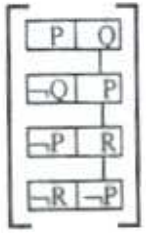


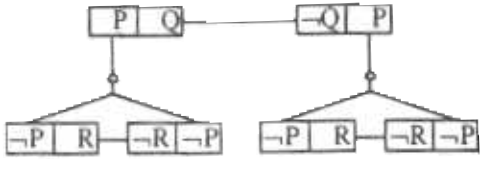
Resolvente: {R}





Resolvente: {¬P}





Resolvente: □

Introduction to Deduction Systems:

From Natural Deduction to Sequent Calculus

(and back)

Christoph Benzmüller

Reading

- F. Pfenning: Automated Theorem Proving, Course at Carnegie Mellon University. Draft. 1999.
- A.S. Troelstra and H. Schwichtenberg: Basic Proof Theory. Cambridge. 2nd Edition 2000.
- John Byrnes: Proof Search and Normal Forms in Natural Deduction. PhD Thesis. Carnegie Mellon University. 1999.
- . . . many more books on Proof Theory . . .

Natural Deduction: Motivation

Frege, Russel, Hilbert Predicate calculus and type theory as formal basis for mathematics

Gentzen ND as intuitive formulation of predicate calculus; introduction and elimination rules for each logical connective

The formalization of logical deduction, especially as it has been developed by Frege, Russel, and Hilbert, is rather far removed from the forms of deduction used in practice in mathematical proofs. . . . In contrast I intended first to set up a formal system which comes as close as possible to actual reasoning. The result was a **calculus of natural deduction** *(NJ for intuitionist, NK for classical predicate logic).* [Gentzen: Investigations into logical deduction]

Sequent Calculus: Motivation

Gentzen had a pure technical motivation for sequent calculus

- Same theorems as natural deduction
- Prove of the Hauptsatz (all sequent proofs can be found with a simple strategy)
- Corollary: Consistency of formal system(s)

The Hauptsatz says that every purely logical proof can be reduced to a definite, though not unique, normal form. Perhaps we may express the essential properties of such a normal proof by saying: it is not roundabout. . . .
In order to be able to prove the Hauptsatz in a convenient form, **I had to provide a logical calculus especially for the purpose**. *For this the natural calculus proved unsuitable.* [Gentzen: Investigations into logical deduction]

Sequent Calculus: Introduction

Sequent calculus exposes many details of fine structure of proofs in a very clear manner. Therefore it is well suited to serve as a **basic representation formalism** for many automation oriented search procedures

- Backward: tableaux, connection methods, matrix methods, some forms of resolution
- Forward: classical resolution, inverse method

Don't be afraid of the many variants of sequent calculi.

Choose the one that is most suited for you.

Natural Deduction

Natural deduction rules operate on proof trees.

Example:

- Conjunction: $\dfrac{\overset{D_1}{\mathbf{A}} \quad \overset{D_2}{\mathbf{B}}}{\mathbf{A} \wedge \mathbf{B}} \wedge I \qquad \dfrac{\overset{D_1}{\mathbf{A} \wedge \mathbf{B}}}{\mathbf{A}} \wedge E_l \qquad \dfrac{\overset{D_1}{\mathbf{A} \wedge \mathbf{B}}}{\mathbf{B}} \wedge E_r$

The presentation on the next slides treats the proof tree aspects implicit.

Example:

- Conjunction: $\dfrac{\mathbf{A} \quad \mathbf{B}}{\mathbf{A} \wedge \mathbf{B}} \wedge I \qquad \dfrac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{A}} \wedge E_l \qquad \dfrac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{B}} \wedge E_r$

Natural Deduction Rules Ia

- Conjunction: $\dfrac{\mathbf{A} \quad \mathbf{B}}{\mathbf{A} \wedge \mathbf{B}} \wedge I \qquad \dfrac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{A}} \wedge E_l \qquad \dfrac{\mathbf{A} \wedge \mathbf{B}}{\mathbf{B}} \wedge E_r$
- Disjunction:

$$\frac{\mathbf{A}}{\mathbf{A} \vee \mathbf{B}} \vee I_l \qquad \frac{\mathbf{B}}{\mathbf{A} \vee \mathbf{B}} \vee I_r \qquad \frac{\mathbf{A} \vee \mathbf{B} \quad \overset{[\mathbf{A}]_1}{\mathbf{C}} \quad \overset{[\mathbf{B}]_2}{\mathbf{C}}}{\mathbf{C}} \vee E_r^{1,2}$$

- Implication: $\dfrac{\overset{[\mathbf{A}]_1}{\mathbf{B}}}{\mathbf{A} \Rightarrow \mathbf{B}} \Rightarrow I^1 \qquad \dfrac{\mathbf{A} \Rightarrow \mathbf{B} \quad \mathbf{A}}{\mathbf{B}} \Rightarrow E$
- Truth and Falsehood: $\dfrac{}{\top} \top I \qquad \dfrac{\bot}{\mathbf{C}} \bot E$

Natural Deduction Rules IIa

- Negation: $\dfrac{\overset{[\mathbf{A}]_1}{\bot}}{\neg \mathbf{A}} \neg I^1 \qquad \dfrac{\neg \mathbf{A} \quad \mathbf{A}}{\bot} \neg E$
- Universal Quantif.: $\dfrac{\{a^*/x\}\mathbf{A}}{\forall x.\ \mathbf{A}} \forall I \qquad \dfrac{\forall x.\ \mathbf{A}}{\{t/x\}\mathbf{A}} \forall E$
- Existential Quantif.: $\dfrac{\{t/x\}\mathbf{A}}{\exists x.\ \mathbf{A}} \exists I \qquad \dfrac{\exists x.\ \mathbf{A} \quad \overset{[\{a^*/x\}\mathbf{A}]}{\mathbf{C}}}{\mathbf{C}} \exists E$

*: parameter a must be new in context

Natural Deduction Rules IIIa

For classical logic choose one of the following

- Excluded Middle $\dfrac{}{\mathbf{A} \vee \neg \mathbf{A}}\, XM$
- Double Negation $\dfrac{\neg\neg \mathbf{A}}{\mathbf{A}}\, \neg\neg C$
- Proof by Contradiction $\dfrac{\begin{array}{c}[\neg \mathbf{A}] \\ \vdots \\ \bot\end{array}}{\mathbf{A}}\, \bot_c$

Natural Deduction

Structural properties

- Exchange — hypotheses order is irrelevant
- Weakening — hypothesis need not be used
- Contraction — hypotheses can be used more than once

Natural Deduction Proofs

$$\dfrac{\dfrac{\dfrac{[\mathbf{A}]_1 \quad [\mathbf{A}]_2}{\mathbf{A} \wedge \mathbf{A}}\, \wedge I}{\mathbf{A} \Rightarrow (\mathbf{A} \wedge \mathbf{A})}\, \Rightarrow I^2}{\mathbf{A} \Rightarrow (\mathbf{A} \Rightarrow (\mathbf{A} \wedge \mathbf{A}))}\, \Rightarrow I^1$$

$$\dfrac{\dfrac{\dfrac{[\mathbf{A} \wedge \mathbf{B}]_1}{\mathbf{B}}\, \wedge E_r \quad \dfrac{\dfrac{[\mathbf{A} \wedge \mathbf{B}]_1}{\mathbf{A}}\, \wedge E_l}{\mathbf{C} \vee \mathbf{A}}\, \vee I_r}{\mathbf{B} \wedge (\mathbf{C} \vee \mathbf{A})}\, \wedge I}{(\mathbf{A} \wedge \mathbf{B}) \Rightarrow (\mathbf{B} \wedge (\mathbf{C} \vee \mathbf{A}))}\, \Rightarrow I^1$$

Natural Deduction with Contexts

Idea: Localizing hypotheses; explicit representation of the available assumptions for each formula occurrence in a ND proof:

$$\Gamma \vdash \mathbf{A}$$

Γ is a multiset of the (uncanceled) assumptions on which formula $\mathbf{A}$ depends. Γ is called context.

Example proof in context notation:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\mathbf{A}_1 \vdash \mathbf{A}} \quad \dfrac{}{\mathbf{A}_2 \vdash \mathbf{A}}}{\mathbf{A}_1, \mathbf{A}_2 \vdash \mathbf{A} \wedge \mathbf{A}}\, \wedge I}{\mathbf{A}_1 \vdash \mathbf{A} \Rightarrow (\mathbf{A} \wedge \mathbf{A})}\, \Rightarrow I_2}{\vdash \mathbf{A} \Rightarrow (\mathbf{A} \Rightarrow (\mathbf{A} \wedge \mathbf{A}))}\, \Rightarrow I_1$$

Natural Deduction with Contexts

Another Idea: Consider sets of assumptions instead of multisets.

$$\Gamma \vdash \mathbf{A}$$

Γ is now a set of (uncanceled) assumptions on which formula $\mathbf{A}$ depends.

Example proof:

$$\dfrac{\dfrac{\dfrac{\overline{\mathbf{A} \vdash \mathbf{A}} \quad \overline{\mathbf{A} \vdash \mathbf{A}}}{\mathbf{A} \vdash \mathbf{A} \wedge \mathbf{A}} \wedge I}{\mathbf{A} \vdash \mathbf{A} \Rightarrow (\mathbf{A} \wedge \mathbf{A})} \Rightarrow I}{\vdash \mathbf{A} \Rightarrow (\mathbf{A} \Rightarrow (\mathbf{A} \wedge \mathbf{A}))} \Rightarrow I$$

Natural Deduction with Contexts

Structural properties to ensure

- Exchange (hypotheses order is irrelevant)

$$\dfrac{\Gamma, \mathbf{B}, \mathbf{A} \vdash \mathbf{C}}{\Gamma, \mathbf{A}, \mathbf{B} \vdash \mathbf{C}}$$

- Weakening (hypothesis need not be used)

$$\dfrac{\Gamma \vdash \mathbf{C}}{\Gamma, \mathbf{A} \vdash \mathbf{C}}$$

- Contraction (hypotheses can be used more than once)

$$\dfrac{\Gamma, \mathbf{A}, \mathbf{A} \vdash \mathbf{C}}{\Gamma, \mathbf{A} \vdash \mathbf{C}}$$

Natural Deduction Rules Ib

- Hypotheses: $\dfrac{}{\Gamma, \mathbf{A}, \Delta \vdash \mathbf{A}}$
- Conjunction:

$$\dfrac{\Gamma \vdash \mathbf{A} \quad \Gamma \vdash \mathbf{B}}{\Gamma \vdash \mathbf{A} \wedge \mathbf{B}} \wedge I \qquad \dfrac{\Gamma \vdash \mathbf{A} \wedge \mathbf{B}}{\Gamma \vdash \mathbf{A}} \wedge E_l \qquad \dfrac{\Gamma \vdash \mathbf{A} \wedge \mathbf{B}}{\Gamma \vdash \mathbf{B}} \wedge E_r$$

- Disjunction:

$$\dfrac{\Gamma \vdash \mathbf{A}}{\Gamma \vdash \mathbf{A} \vee \mathbf{B}} \vee I_l \qquad \dfrac{\Gamma \vdash \mathbf{B}}{\Gamma \vdash \mathbf{A} \vee \mathbf{B}} \vee I_r$$

$$\dfrac{\Gamma \vdash \mathbf{A} \vee \mathbf{B} \quad \Gamma, \mathbf{A} \vdash \mathbf{C} \quad \Gamma, \mathbf{B} \vdash \mathbf{C}}{\Gamma \vdash \mathbf{C}} \vee E_r$$

- Implication:

$$\dfrac{\Gamma, \mathbf{A} \vdash \mathbf{B}}{\Gamma \vdash \mathbf{A} \Rightarrow \mathbf{B}} \Rightarrow I \qquad \dfrac{\Gamma \vdash \mathbf{A} \Rightarrow \mathbf{B} \quad \Gamma \vdash \mathbf{A}}{\Gamma \vdash \mathbf{B}} \Rightarrow E$$

Natural Deduction Rules IIb

- Truth and Falsehood:

$$\dfrac{}{\Gamma \vdash \top} \top I \qquad \dfrac{\Gamma \vdash \bot}{\Gamma \vdash \mathbf{C}} \bot E$$

- Negation:

$$\dfrac{\Gamma, \mathbf{A} \vdash \bot}{\Gamma \vdash \neg \mathbf{A}} \neg I \qquad \dfrac{\Gamma \vdash \neg \mathbf{A} \quad \Gamma \vdash \mathbf{A}}{\Gamma \vdash \bot} \neg E$$

- Universal Quantif.:

$$\dfrac{\Gamma \vdash \{a^*/x\}\mathbf{A}}{\Gamma \vdash \forall x.\, \mathbf{A}} \forall I \qquad \dfrac{\Gamma \vdash \forall x.\, \mathbf{A}}{\Gamma \vdash \{t/x\}\mathbf{A}} \forall E$$

- Existential Quantif.:

$$\dfrac{\Gamma \vdash \{t/x\}\mathbf{A}}{\Gamma \vdash \exists x.\, \mathbf{A}} \exists I \qquad \dfrac{\Gamma \vdash \exists x.\, \mathbf{A} \quad \Gamma, \{a^*/x\}\mathbf{A} \vdash \mathbf{C}}{\Gamma \vdash \mathbf{C}} \exists E$$

*: parameter a must be new in context

Natural Deduction Rules IIIb

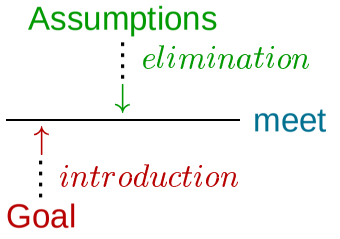
For classical logic add:

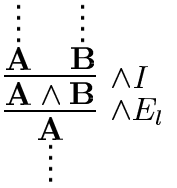
- Proof by Contradiction:

$$\frac{\Gamma, \neg \mathbf{A} \vdash \bot}{\Gamma \vdash \mathbf{A}} \bot_c$$

Intercalation

Idea (Prawitz, Sieg & Scheines, Byrnes & Sieg):
Detour free proofs: strictly use introduction rules bottom up (from proposed theorem to hypothesis) and elimination rules top down (from assumptions to proposed theorem). When they meet in the middle we have found a proof in normal form.





Intercalating Natural Deductions

New annotations:

- $\mathbf{A}\uparrow$: $\mathbf{A}$ is obtained by an introduction derivation
- $\mathbf{A}\downarrow$: $\mathbf{A}$ is extracted from a hypothesis by an elimination derivation

Example:

$$\frac{\Gamma, \mathbf{A} \models_{\mathrm{ic}} \mathbf{B}\uparrow}{\Gamma \models_{\mathrm{ic}} \mathbf{A} \Rightarrow \mathbf{B}\uparrow} \Rightarrow I \qquad \frac{\Gamma \models_{\mathrm{ic}} \mathbf{A} \Rightarrow \mathbf{B}\downarrow \quad \Gamma \models_{\mathrm{ic}} \mathbf{A}\downarrow}{\Gamma \models_{\mathrm{ic}} \mathbf{B}\downarrow} \Rightarrow E$$

ND Intercalation Rules I

- Hypotheses:

$$\frac{}{\Gamma, \mathbf{A}, \Delta \models_{\mathrm{ic}} \mathbf{A}\downarrow}$$

- Conjunction:

$$\frac{\Gamma \models_{\mathrm{ic}} \mathbf{A}\uparrow \quad \Gamma \models_{\mathrm{ic}} \mathbf{B}\uparrow}{\Gamma \models_{\mathrm{ic}} \mathbf{A} \wedge \mathbf{B}\uparrow} \wedge I \qquad \frac{\Gamma \models_{\mathrm{ic}} \mathbf{A} \wedge \mathbf{B}\downarrow}{\Gamma \models_{\mathrm{ic}} \mathbf{A}\downarrow} \wedge E_l \qquad \frac{\Gamma \models_{\mathrm{ic}} \mathbf{A} \wedge \mathbf{B}\downarrow}{\Gamma \models_{\mathrm{ic}} \mathbf{B}\downarrow} \wedge E_r$$

- Disjunction:

$$\frac{\Gamma \models_{\mathrm{ic}} \mathbf{A}\uparrow}{\Gamma \models_{\mathrm{ic}} \mathbf{A} \vee \mathbf{B}\uparrow} \vee I_l \qquad \frac{\Gamma \models_{\mathrm{ic}} \mathbf{B}\uparrow}{\Gamma \models_{\mathrm{ic}} \mathbf{A} \vee \mathbf{B}\uparrow} \vee I_r$$

$$\frac{\Gamma \models_{\mathrm{ic}} \mathbf{A} \vee \mathbf{B}\downarrow \quad \Gamma, \mathbf{A} \models_{\mathrm{ic}} \mathbf{C}\uparrow \quad \Gamma, \mathbf{B} \models_{\mathrm{ic}} \mathbf{C}\uparrow}{\Gamma \vdash \mathbf{C}\uparrow} \vee E_r$$

- Implication:

$$\frac{\Gamma, \mathbf{A} \models_{\mathrm{ic}} \mathbf{B}\uparrow}{\Gamma \models_{\mathrm{ic}} \mathbf{A} \Rightarrow \mathbf{B}\uparrow} \Rightarrow I \qquad \frac{\Gamma \models_{\mathrm{ic}} \mathbf{A} \Rightarrow \mathbf{B}\downarrow \quad \Gamma \models_{\mathrm{ic}} \mathbf{A}\downarrow}{\Gamma \models_{\mathrm{ic}} \mathbf{B}\downarrow} \Rightarrow E$$

ND Intercalation Rules II

- Truth and Falsehood: $\dfrac{}{\Gamma \models_{ic} \top \uparrow}\ \top I \qquad \dfrac{\Gamma \models_{ic} \bot \downarrow}{\Gamma \models_{ic} \mathbf{C} \uparrow}\ \bot E$
- Negation: $\dfrac{\Gamma, \mathbf{A} \models_{ic} \bot \uparrow}{\Gamma \models_{ic} \neg \mathbf{A} \uparrow}\ \neg I \qquad \dfrac{\Gamma \models_{ic} \neg \mathbf{A} \downarrow \quad \Gamma \models_{ic} \mathbf{A} \uparrow}{\Gamma \models_{ic} \bot \uparrow}\ \neg E$
- Universal Quantif.: $\dfrac{\Gamma \models_{ic} \{a^*/x\}\mathbf{A} \uparrow}{\Gamma \models_{ic} \forall x.\ \mathbf{A} \uparrow}\ \forall I \qquad \dfrac{\Gamma \models_{ic} \forall x.\ \mathbf{A} \downarrow}{\Gamma \models_{ic} \{t/x\}\mathbf{A} \downarrow}\ \forall E$
- Existential Quantif.: $\dfrac{\Gamma \models_{ic} \{t/x\}\mathbf{A} \uparrow}{\Gamma \models_{ic} \exists x.\ \mathbf{A} \uparrow}\ \exists I \qquad \dfrac{\Gamma \models_{ic} \exists x.\ \mathbf{A} \downarrow \quad \Gamma, \{a^*/x\}\mathbf{A} \models_{ic} \mathbf{C} \uparrow}{\Gamma \vdash \mathbf{C} \uparrow}\ \exists E$

*: parameter a must be new in context

ND Intercalation Rules III

For classical logic add:

- Proof by Contradiction: $\dfrac{\Gamma, \neg \mathbf{A} \models_{ic} \bot \uparrow}{\Gamma \models_{ic} \mathbf{A} \uparrow}\ \bot_c$

Intercalation and ND

Normal form proofs

Assumptions

elimination ↓

——— meet

↑ *introduction*

Goal

guaranteed by $\dfrac{\Gamma \models_{ic} \mathbf{A} \downarrow}{\Gamma \models_{ic} \mathbf{A} \uparrow}$ meet

. . . proofs without detour . . .

To model all ND proofs add $\dfrac{\Gamma \models_{ic} \mathbf{A} \uparrow}{\Gamma \models_{ic} \mathbf{A} \downarrow}$ roundabout

Example Proofs

In normal form

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{M} \wedge \mathbf{Q} \models_{ic} \mathbf{M} \wedge \mathbf{Q} \downarrow}{\mathbf{M} \wedge \mathbf{Q} \models_{ic} \mathbf{Q} \downarrow}\ \wedge E_r}{\mathbf{M} \wedge \mathbf{Q} \models_{ic} \mathbf{Q} \uparrow}\ \text{meet}}{\mathbf{M} \wedge \mathbf{Q} \models_{ic} \mathbf{Q} \vee \mathbf{S} \uparrow}\ \vee I_l}{\models_{ic} (\mathbf{M} \wedge \mathbf{Q}) \Rightarrow (\mathbf{Q} \vee \mathbf{S}) \uparrow}\ \Rightarrow I$$

With detour

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{M} \wedge \mathbf{Q} \models_{ic} \mathbf{Q} \uparrow \quad \mathbf{M} \wedge \mathbf{Q} \models_{ic} \mathbf{M} \uparrow}{\mathbf{M} \wedge \mathbf{Q} \models_{ic} \mathbf{Q} \wedge \mathbf{M} \uparrow}\ \wedge I}{\mathbf{M} \wedge \mathbf{Q} \models_{ic} \mathbf{Q} \wedge \mathbf{M} \downarrow}\ \text{roundabout}}{\mathbf{M} \wedge \mathbf{Q} \models_{ic} \mathbf{Q} \downarrow}\ \wedge E_l}{\mathbf{M} \wedge \mathbf{Q} \models_{ic} \mathbf{Q} \uparrow}\ \text{meet}$$

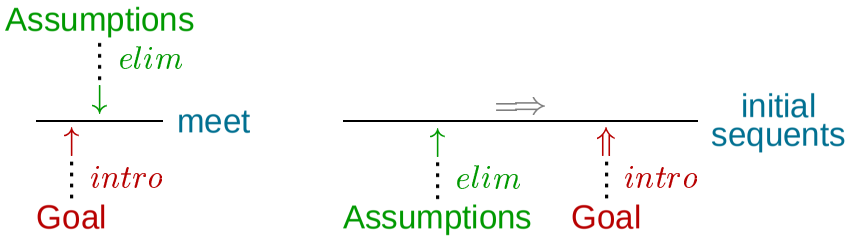
Soundness and Completeness

Let $\vdash^{\pm}_{ic}$ denote the intercalation calculus with rule roundabout and $\vdash_{ic}$ the calculus without this rule.

- Theorem 1 (Soundness):
  If $\Gamma \vdash^{\pm}_{ic} \mathbf{A} \uparrow$ then $\Gamma \vdash \mathbf{A}$.
- Theorem 2 (Completeness):
  If $\Gamma \vdash \mathbf{A}$ then $\Gamma \vdash^{\pm}_{ic} \mathbf{A} \uparrow$.
- Is normal form proof search also complete?:
  If $\Gamma \vdash^{\pm}_{ic} \mathbf{A} \uparrow$ then $\Gamma \vdash_{ic} \mathbf{A} \uparrow$?
  We will investigate this question within the **sequent calculus**.

From ND to Sequent Calculus

Normal form ND proofs **Sequent proofs**



Sequents pair <Γ, Δ> of finite lists, multisets, or sets of formulas; notation: $\Gamma \Longrightarrow \Delta$

Intuitive: a kind of implication, Δ "follows from" Γ

Sequent Calculus Rules I

- Initial Sequents: $\dfrac{}{\Gamma, \mathbf{A} \Longrightarrow \Delta, \mathbf{A}}\ init$ (**A** atomic)
- Conjunction:
  $\dfrac{\Gamma, \mathbf{A}, \mathbf{B} \Longrightarrow \Delta}{\Gamma, \mathbf{A} \wedge \mathbf{B} \Longrightarrow \Delta} \wedge L$ $\dfrac{\Gamma \Longrightarrow \Delta, \mathbf{A} \quad \Gamma \Longrightarrow \Delta, \mathbf{B}}{\Gamma \Longrightarrow \Delta, \mathbf{A} \wedge \mathbf{B}} \wedge R$
- Implication
  $\dfrac{\Gamma \Longrightarrow \Delta, \mathbf{A} \quad \Gamma, \mathbf{B} \Longrightarrow \Delta}{\Gamma, \mathbf{A} \Rightarrow \mathbf{B} \Longrightarrow \Delta} \Rightarrow L$ $\dfrac{\Gamma, \mathbf{A} \Longrightarrow \Delta, \mathbf{B}}{\Gamma \Longrightarrow \Delta, \mathbf{A} \Rightarrow \mathbf{B}} \Rightarrow R$
- Truth and Falsehood $\dfrac{}{\Gamma, \bot \Longrightarrow \Delta} \bot L$ $\dfrac{}{\Gamma \Longrightarrow \Delta, \top} \top R$

Sequent Calculus Rules II

- Negation: $\dfrac{\Gamma \Longrightarrow \Delta, \mathbf{A}}{\Gamma, \neg\mathbf{A} \Longrightarrow \Delta} \neg L$ $\dfrac{\Gamma, \mathbf{A} \Longrightarrow \Delta}{\Gamma \Longrightarrow \Delta, \neg\mathbf{A}} \neg R$
- Disjunction:
  $\dfrac{\Gamma \Longrightarrow \Delta, \mathbf{A}, \mathbf{B}}{\Gamma \Longrightarrow \Delta, \mathbf{A} \vee \mathbf{B}} \vee R$ $\dfrac{\Gamma, \mathbf{A} \Longrightarrow \Delta \quad \Gamma, \mathbf{B} \Longrightarrow \Delta}{\Gamma, \mathbf{A} \vee \mathbf{B} \Longrightarrow \Delta} \vee L$
- Universal Quantification:
  $\dfrac{\Gamma, \forall x.\ \mathbf{A}, \{t/x\}\mathbf{A} \Longrightarrow \Delta}{\Gamma, \forall x.\ \mathbf{A} \Longrightarrow \Delta} \forall L$ $\dfrac{\Gamma \Longrightarrow \Delta, \{a/x\}\mathbf{A}}{\Gamma \Longrightarrow \Delta, \forall x.\ \mathbf{A}} \forall R$
- Existential Quantification:
  $\dfrac{\Gamma, \{a/x\}\mathbf{A} \Longrightarrow \Delta}{\Gamma, \exists x.\ \mathbf{A} \Longrightarrow \Delta} \forall L$ $\dfrac{\Gamma \Longrightarrow \Delta, \exists x.\ \mathbf{A}, \{t/x\}\mathbf{A}}{\Gamma \Longrightarrow \Delta, \exists x.\ \mathbf{A}} \forall R$

Example Proof

$$\dfrac{\dfrac{\dfrac{}{\mathbf{A},\mathbf{B} \Longrightarrow \mathbf{B}}\,init}{\mathbf{A} \wedge \mathbf{B} \Longrightarrow \mathbf{B}}\,\wedge L \qquad \dfrac{\dfrac{}{\mathbf{A},\mathbf{B} \Longrightarrow \mathbf{C},\mathbf{A}}\,init}{\mathbf{A} \wedge \mathbf{B} \Longrightarrow \mathbf{C},\mathbf{A}}\,\wedge L}{\mathbf{A} \wedge \mathbf{B} \Longrightarrow \mathbf{C} \vee \mathbf{A}}\,\vee R}{\mathbf{A} \wedge \mathbf{B} \Longrightarrow \mathbf{B} \wedge (\mathbf{C} \vee \mathbf{A})}\,\wedge R$$
$$\Longrightarrow (\mathbf{A} \wedge \mathbf{B}) \Rightarrow \mathbf{B} \wedge (\mathbf{C} \vee \mathbf{A})\,\Rightarrow R$$

Sequent Calculus: Cut-rule

To map natural deductions (in $\vdash$ and $\vdash^{\pm}_{ic}$) to sequent calculus derivations we add: called **cut-rule**:

$$\dfrac{\Gamma \Longrightarrow \Delta, \mathbf{A} \quad \Gamma, \mathbf{A} \Longrightarrow \Delta}{\Gamma \Longrightarrow \Delta}\,Cut$$

The question whether normal form proof search ($\vdash_{ic}$) is complete corresponds to the question whether the cut-rule can be eliminated (is *admissible*) in sequent calculus.

Sequent Calculus

Let $\Longrightarrow^{+}$ denote the sequent calculus with cut-rule and $\Longrightarrow$ the sequent calculus without the cut-rule.

Theorem 3 (Soundness)
(a) If $\Gamma \Longrightarrow \mathbf{C}$ then $\Gamma \vdash_{ic} \mathbf{C}\uparrow$.
(b) If $\Gamma \Longrightarrow^{+} \mathbf{C}$ then $\Gamma \vdash^{\pm}_{ic} \mathbf{C}\uparrow$.

Theorem 4 (Completeness)
If $\Gamma \vdash^{\pm}_{ic} \mathbf{C}\uparrow$ then $\Gamma \Longrightarrow^{+} \mathbf{C}$.

Gentzen's Hauptsatz

Theorem 5 (Cut-Elimination): Cut-elimination holds for the sequent calculus. In other words: The cut rule is *admissible* in the sequent calculus.

$$\text{If } \Gamma \Longrightarrow^{+} \mathbf{C} \text{ then } \Gamma \Longrightarrow \mathbf{C}$$

Proof non-trivial; main means: nested inductions and case distinctions over rule applications

This result qualifies the sequent calculus as suitable for automating proof search.

Applications of Cut-Elimination

Theorem (Normalization for ND):

If $\Gamma \vdash \mathbf{C}$ then $\Gamma \vdash_{\mathrm{ic}} \mathbf{C} \uparrow$.

Proof sketch:

Assume $\Gamma \vdash \mathbf{C}$.

Then $\Gamma \vdash_{\mathrm{ic}}^{\pm} \mathbf{C} \uparrow$ by completeness of $\vdash_{\mathrm{ic}}^{\pm}$.

Then $\Gamma \Longrightarrow^{+} \mathbf{C}$ by completeness of $\Longrightarrow^{+}$.

Then $\Gamma \Longrightarrow \mathbf{C}$ by cut-elimination.

Then $\Gamma \vdash_{\mathrm{ic}} \mathbf{C} \uparrow$ by soundness of $\Longrightarrow$.

What have we done?

Natural Deduction	Intercalation	Sequent Calculus
$\vdash$ (with detours)	$\vdash_{\mathrm{ic}}^{\pm}$ (with roundabout)	$\Longrightarrow^{+}$ (with cut)
$\vdash$ (without detours)	$\vdash_{\mathrm{ic}}$ (without roundabout)	$\Longrightarrow$ (without cut)

Applications of Cut-Elimination

Theorem (Consistency of ND): There is no natural deduction derivation $\vdash \bot$.

Proof sketch:

Assume there is a proof of $\vdash \bot$.

Then $\Longrightarrow^{+} \bot$ by completeness of $\Longrightarrow^{+}$ and $\vdash_{\mathrm{ic}}^{\pm}$.

But $\Longrightarrow^{+} \bot$ cannot be the conclusion of any sequent rule.

Contradiction.

Summary

We have illustrated the connection of

- natural deduction and sequent calculus
- normal form natural deductions and cut-free sequent calculus.

Fact: Sequent calculus often employed as meta-theory for specialized proof search calculi and strategies.

Question: Can these calculi and strategies be transformed to natural deduction proof search?

How to build a (resolution) prover?

Andreas Meier

Saarland University, Saarbrücken, Germany

Proof Search by Saturation

Given: Set of clauses

+ Inferences (resolution, factorization, paramodulation)

⇒ Saturate clause set with all possible inferences

1. if empty clause in clause set, then terminate
2. select clauses
3. apply inferences to selected clauses
4. add result to clause set; goto 1

Problem?

Dealing with millions of clauses . . .

Efficient automated theorem prover by

good theory

\+ efficient implementation

\+ clever heuristics

References: [Voronkov, IJCAR, 2001]

Theory

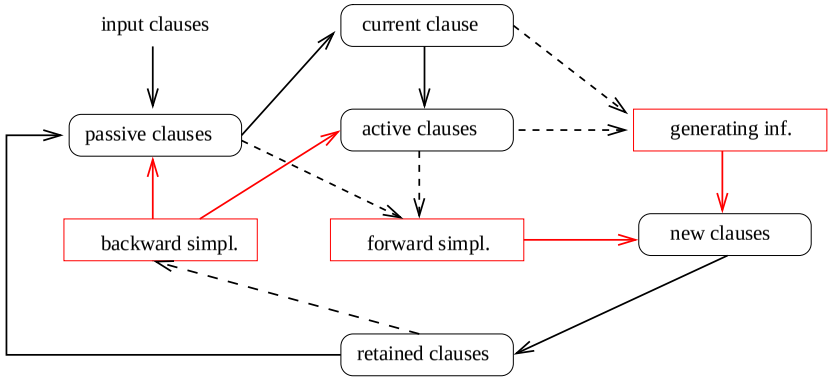
Progress in the theory of resolution-based systems:

- reduction of possible inferences
  without impair of completeness
  (e.g., superposition calculus)
- decision procedures for some fragments
  (e.g., realized in BLIKSEM)

References: [Bachmair+Ganzinger, Handbook of Aut. Reas. I]
[Nieuwenhuis+Rubio, Handbook of Aut. Reas. I]
[Bachmair+Ganzinger, Diverse CADE,LPAR]

Implementation

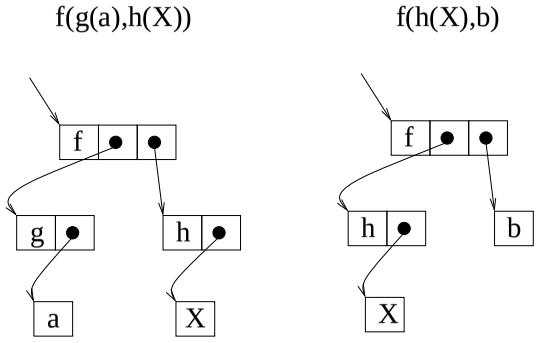
Organization



References: [McCune, OTTER 3.0 Manual, 1994]

Implementation

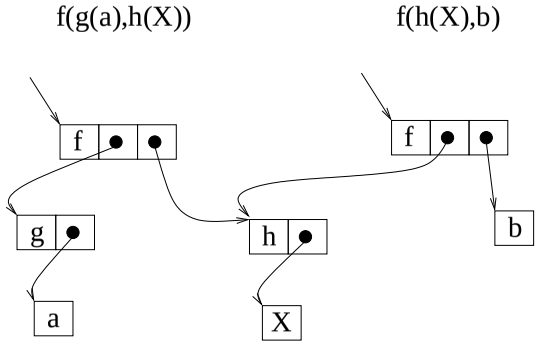
Efficient Data Structures



tree representation

Implementation

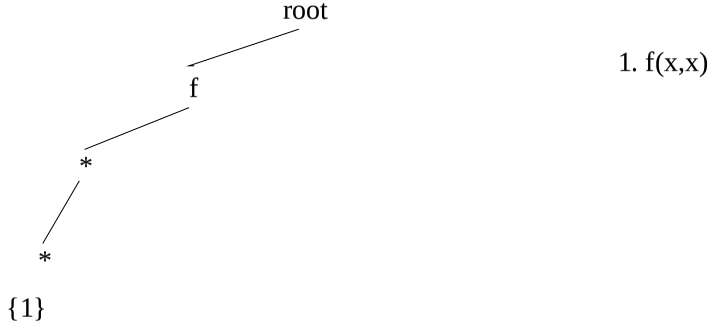
Efficient Data Structures



shared tree-like representation

Implementation

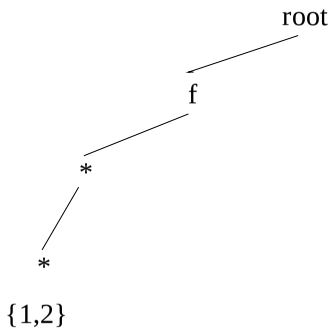
Efficient Data Structures



discrimination tree index

Implementation

Efficient Data Structures

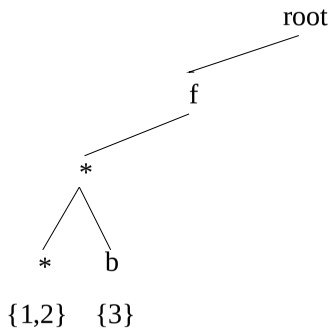


1. f(x,x)
2. f(x,y)

discrimination tree index

Implementation

Efficient Data Structures

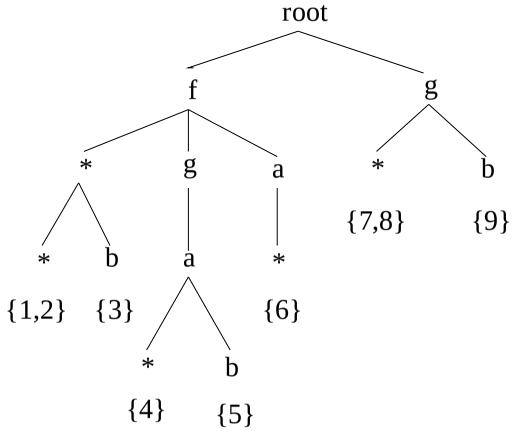


1. f(x,x)
2. f(x,y)
3. f(x,b)

discrimination tree index

Implementation

Efficient Data Structures

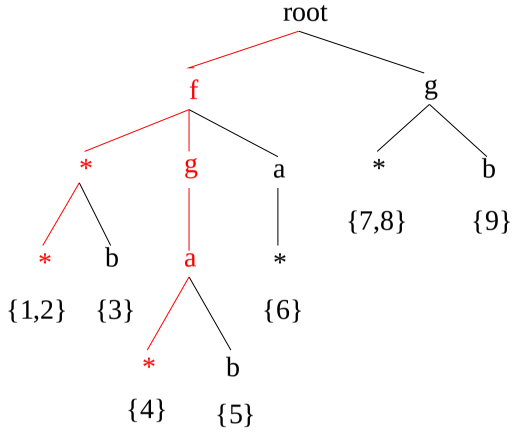


1. f(x,x)
2. f(x,y)
3. f(x,b)
4. f(g(a),x)
5. f(g(a),b)
6. f(a,y)
7. g(x)
8. g(z)
9. g(b)

discrimination tree index

Implementation

Efficient Data Structures

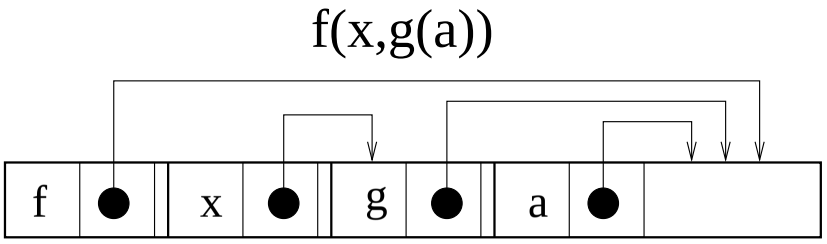


1. f(x,x)
2. f(x,y)
3. f(x,b)
4. f(g(a),x)
5. f(g(a),b)
6. f(a,y)
7. g(x)
8. g(z)
9. g(b)

retrieving variants of: $f(g(x), a)$

Implementation

Efficient Data Structures



array-based flat-term representation

Implementation

Efficient Data Structures

References: [Graf, LNCS 1053, 1996]

[McCune, JAR, 1992]

[Ramakrishnan+Sekar+Voronkov,

Handbook of Aut. Reas. II]

Control

parameterized algorithms

(e.g., 32 parameters to determine OTTER's main algorithm)

$\Rightarrow$ selection of the "right" parameterization is crucial

either by the user

or automatically by the system

For instance:

- OTTER: auto mode
- VAMPIRE: preprocessor
- WALDMEISTER: self-adaption component

Control

parameterized algorithms

(e.g., 32 parameters to determine OTTER's main algorithm)

selection of the "right" parameterization is crucial

OR: try different instances competitively

- RCTHEO randomized decision points in SETHEO [Ertel, LPAR, 1992]
- SICOTHEO pre-defined strategies in SETHEO [Schumann, 1995]

5 Wolfgang Windsteiger
RISC Linz, Austria

Course: The THEOREMA System

The *Theorema* System

Wolfgang Windsteiger

RISC Institute
A-4232 Hagenberg, Austria
Wolfgang.Windsteiger@RISC.Uni-Linz.ac.at

Calculemus Autumn School, Pisa, September 27, 2002.

1 History

1.1 Initial Motivation

Initial motivation for development of *Theorema*:

Observation [Buchberger, ≈1990 or earlier]:

$$\nexists_{\text{system}}\ \text{is-one-software[system]} \wedge \text{supports-all-of-algorithmic-mathematics[system]}$$

where

$$\text{supports-all-of-algorithmic-mathematics[s]} :\Leftrightarrow \bigwedge \begin{cases} \text{supports-defining-mathematical-concepts-in-a-natural-way[s]} \\ \text{supports-experimenting-by-large-nontrivial-computations[s]} \\ \text{supports-conjecturing-theorems[s]} \\ \text{supports-proving-theorems[s]} \\ \text{supports-gaining-insight-through-proving[s]} \\ \text{supports-turning-theorems-into-algorithms[s]} \\ \text{supports-publishing-mathematical-results[s]} \end{cases}$$

1.2 On the other hand …

Fact 1:

$$\exists_{\text{system}} \wedge \begin{cases} \text{supports-defining-mathematical-concepts-in-a-natural-way[system]} \\ \text{supports-proving-theorems[system]} \\ \text{supports-gaining-insight-through-proving[system]} \end{cases}$$

Proof:

Take system:="the logic system used by human mathematicians",
e.g. system:="some version of predicate logic".

Fact 2:

$$\exists_{\text{system}} \wedge \begin{cases} \text{is-one-software[system]} \\ \text{supports-experimenting-by-large-nontrivial-computations[system]} \\ \text{supports-turning-theorems-into-algorithms[system]} \end{cases}$$

Proof:

Take system:="any available computer algebra system".

Fact 3:

$$\exists_{\text{system}} \wedge \begin{cases} \text{is-one-software[system]} \\ \text{supports-proving-theorems[system]} \end{cases}$$

Proof:

Take system:="any available theorem proving system".

Fact 4:

$$\exists_{\text{system}} \wedge \begin{cases} \text{is-one-software[system]} \\ \text{supports-publishing-mathematical-results[system]} \end{cases}$$

Proof:

Take system:="any mathematical typesetting system",
e.g. system:=LaTeX.

1.3 Theorema

Implementation Design Specification

Find the most appropriate *programming language* or *software-system* for implementing one piece of software *system*, such that

$$\text{is-one-software[system]} \wedge \text{supports-all-of-algorithmic-mathematics[system]}$$

Implementation Design Decision Process (Buchberger and Co-Workers, ≈ 1990-1995)

Choose

- the Mathematica programming language (pattern matching),
- the Mathematica rewrite engine as the basis for the computation engine,
- the Mathematica Front End as user interface (configurable, extensible, programmable).

(see the early papers by Buchberger from that time!)

2 System Design

2.1 Language Layer Model

2.1.1 *Theorema* Expression Language

Formulating mathematical contents.
Higher order predicate logic, sets, tuples.
Syntax:
Internal representation as Mathematica data-structures.
External representation in "natural form" using typesetting features of the Mathematica Front End.
"Natural form" input supported by the Mathematica Front End.

```
⟨3, a, 2, N, 1, a⟩
```
⟨3, a, 2, N, 1, a⟩

```
{3, a, 2, N, 1, a}
```
{3, a, 2, N, 1, a}

```
{x^2 | x ∈ ℕ} // FullForm
List[Alternatives[Power[x, 2], Element[x, \[DoubleStruckCapitalN]]]]
```

Fortunately configurable!

```
∀_x P[x] ⋀ Q[f[x], y]
```
Syntax::sntxi : Incomplete expression; more input is needed.
```
∀_x P[x] ⋀ Q[f[x], y]
```

Fortunately extensible!

```
B * A
```
A B

```
5 * 2
```
10

```
2^5
```
32

Fortunately programmable!

```
Needs["Theorema`"]
SetDelayed::write : Tag NotElement in NotElement[x$___] is Protected.
```

```
•[  ⟨3, a, 2, N, 1, a⟩  ]
⟨3, a, 2, N, 1, a⟩
```

```
•[  {3, a, 2, N, 1, a}  ]
{3, a, 2, N, 1, a}
```

```
•[  {x^2 | x ∈ ℕ}  ] // InputForm
TM SetOf[•range[•simpleRange[•var[x]]],
 TM Element[•var[x], TM ℕ], TM Power[•var[x], 2]]
```

```
•[  ⋀_x P[x] ⋁ Q[f[x], y]  ]
⋀_x P[x] ⋁ Q[f[x], y]
```

```
•[  B ∗ A  ]
B ∗ A
```

```
•[  5 ∗ 2  ]
5 ∗ 2
```

```
•[  2^5  ]
2^5
```

Semantics:

Algorithmic (executable) semantics is provided for the algorithmic part of the language.
Theorema semantics **can** be used in computations or in proofs.

2.1.2 *Theorema* Formal Text Language

Organization and structuring of mathematical items.

```
Definition["repeated operation on group", any[a, n],
  self[a, 0] := u                                  "base"
  (self[a, n] := a ∘ self[a, n − 1]) ⇐ is-natural[n]   "general"]
```

```
Definition["reversed repeated operation", any[a, n], with[is-natural[n]],
  self-rev[n, a] := self[a, n]]
```



```
Definition["repeated operation on group"]
```

```
•def[repeated operation on group,
  •range[•simpleRange[a], •simpleRange[n]], True, •flist[•If[base, self[a, 0] := u],
    •If[general, (self[a, n] := a ∘ self[a, n − 1]) ⇐ IsNatural[n]]]]
```

```
Axiom["unit element", any[a],
  a ∘ u = a   "right unit"
  u ∘ a = a   "left unit"]
```

```
Theory["group",
  Axiom["unit element"]
  Definition["repeated operation on group"]
  Definition["reversed repeated operation"]]
```

```
Built-in["multiplicative group",
  ^ → self]
```

```
Built-in["additive group",
  ∗ → self-rev]
```

```
Theorem["Not over And", any[A, B],
  ¬ A ∨ ¬ B ⇒ ¬ (A ∧ B)]
```

2.1.3 *Theorema* User Language

Mathematical activities involving *Theorema* expressions and *Theorema* formal text entities
&
organizing the environment.

$$\text{Mathematical activity} := \begin{cases} \text{proving} & \rightarrow \text{Prove} \\ \text{computing} & \rightarrow \text{Compute} \\ \text{solving} & \rightarrow \text{Solve} \end{cases}$$

Uniform structure:

```
Prove[goal, using → KB, by → Method, built-in → additional knowledge]
```

```
Compute[expression, using → KB, by → Method, built-in → additional knowledge]
```

Computations using explicit knowledge:

```
Compute[self[a, 0]]
```

```
self[a, 0]
```

```
Compute[self[a, 0], using → Definition["repeated operation on group"]]
```
```
u
```
```
Compute[self[a, 5], using → Definition["repeated operation on group"]]
```
```
self[a, 5]
```
```
Compute[self[a, 5], using → Definition["repeated operation on group"],
  built-in → Built-in["Numbers"][is-natural]]
```
```
a∘self[a, 5 – 1]
```
```
Compute[self[a, 5], using → Definition["repeated operation on group"],
  built-in → Built-in["Numbers"][is-natural, –]]
```
```
a∘(a∘(a∘(a∘u)))
```
```
Compute[self[a, 5],
  using → ⟨Definition["repeated operation on group"], Axiom["unit element"]⟩,
  built-in → Built-in["Numbers"][is-natural, –]]
```
```
a∘(a∘(a∘a))
```
```
Compute[self[a, 5], using → Theory["group"],
  built-in → Built-in["Numbers"][is-natural, –]]
```
```
a∘(a∘(a∘a))
```
```
Use[⟨Theory["group"], Built-in["Numbers"][is-natural, –]⟩]
```
```
Compute[self[a, 5]]
```
```
a∘(a∘(a∘a))
```
```
UseAlso[⟨Built-in["multiplicative group"], Built-in["additive group"]⟩]
```
```
Compute[2^5]
```
```
2∘(2∘(2∘(2∘2)))
```
```
Compute[5∗2]
```
```
2∘(2∘(2∘(2∘2)))
```
```
DoNotUse[⟨Built-in["multiplicative group"], Built-in["additive group"]⟩]
```
```
Compute[2^5,
  built-in → Built-in["Numbers"][^]]
```
```
32
```
```
Compute[2^500,
  built-in → Built-in["Numbers"][^]]
```
```
3273390607896141870013189696827599152216642046043064789483291368096133⋯
  3796404674554883270092325904157150886684127560071009217256545885393⋯
  053328527589376
```
```
Use[]
```



```
Prove[Theorem["Not over And"], by → PredicateProver]
```
```
▪ProofObject▪
```

2.2 *Theorema* Proving Philosophy

- Multi-method system

"White-Box-Provers": Use inference rules similar to natural deduction rules. Proof generation and proof presentation is human-oriented.

  - Propositional logic prover
  - Predicate logic prover
  - PCS prover (specialized for alternating quantifiers)
  - Set theory prover
  - Induction over natural numbers
  - Induction over tuples
  - Simplification prover
  - …

"Black-Box-Provers": Transform proof problem and apply known techniques.

  - Gröbner bases prover (geometry)
  - Paule Schorn prover (identities involving ∑)
  - Resolution prover
  - Links to external systems (Otter, Vampire, Waldmeister, …)

- *Present* proofs in a human-like style.
- *Produce* proofs in a human-like style.

3 Some Examples

3.1 Set Theory

3.1.1 Pure Set Theory

Proposition["union powerset", any[A],
$\bigcup \mathcal{P}[A] = A$]

Prove[Proposition["union powerset"], by → SetTheoryPCSProver,
ProverOptions → {DisableProver → {STC}}]

▪ ProofObject ▪

Transform[%, TransformerOptions → {branches → Proved, steps → Useful}]

▪ ProofObject ▪

Proposition["intersection powerset",
$\bigcap \mathcal{P}[A] = \emptyset$]

Prove[Proposition["intersection powerset"], by → SetTheoryPCSProver,
transformBy → ProofSimplifier,
TransformerOptions → {branches → Proved, steps → Useful},
ProverOptions → {DisableProver → {STC}}]

▪ ProofObject ▪

Lemma["intersection indexed union",

$$\bigcup_{i\in\mathcal{I}} A_i \cap \bigcup_{j\in\mathcal{J}} B_j = \bigcup_{i\in\mathcal{I}} \left(\bigcup_{j\in\mathcal{J}} (A_i \cap B_j) \right)$$

]

Prove[Lemma["intersection indexed union"],
by → SetTheoryPCSProver]

▪ ProofObject ▪

3.1.2 Mathematics Using Sets

Definition["reflexivity", any[~ , A],
$\text{is-reflexive}_A[\sim] :\Leftrightarrow \forall_{x\in A}\ x \sim x$]

Definition["class", any[x, A],
$\text{class}_A[x] := \{a \in A \mid a \sim x\}$]

Proposition["non–empty class", any[x ∈ A, A], with[is-reflexive$_A$[~]],
$\text{class}_A[x] \neq \emptyset$]

Proposition["in own class", any[x ∈ A, A], with[is-reflexive$_A$[~]],
$x \in \text{class}_A[x]$]

IsSet[▪[class_[___]]] := True;
Prove[Proposition["in own class"],
by → SetTheoryPCSProver,
using → ⟨Definition["reflexivity"], Definition["class"]⟩,
ProverOptions → {GRWTarget → {"goal", "kb"}, ApplyBuiltIns → {"goal"}}]

▪ ProofObject ▪

Prove[Proposition["non–empty class"],
by → SetTheoryPCSProver,
using → ⟨Definition["reflexivity"], Definition["class"]⟩,
ProverOptions → {GRWTarget → {"goal", "kb"},
RWInsideQuantifiers → True, ApplyBuiltIns → {"goal"}}, SearchDepth → 40]

▪ ProofObject ▪

3.2 PCS

Definition["sum of sequences/functions", any[f, g, x],
$(f+g)[x] = f[x] + g[x]$]

Definition["continuity", any[f, x],

$$\text{continuous}[f, x] :\Leftrightarrow \forall_{\epsilon>0}\ \exists_{\delta>0}\ \forall_{|y-x|<\delta}\ |f[y] - f[x]| < \epsilon$$

]

Lemma["distance of sum", any[x, y, z, t, δ, ϵ],
$|(x+z) - (y+t)| < (\delta+\epsilon) \Leftarrow (|x-y| < \delta \wedge |z-t| < \epsilon)$]

Lemma["mininum, greater", any[m, M1, M2],
$\min[M1, M2] > m \Leftrightarrow (M1 > m \wedge M2 > m)$]

Lemma["mininum, less", any[m, M1, M2],
$(m < M1 \wedge m < M2) \Leftrightarrow m < \min[M1, M2]$]

Proposition["continuity", any[f, g, x],
$\text{continuous}[f, x] \wedge \text{continuous}[g, x] \Rightarrow \text{continuous}[f+g, x]$]

Prove[Proposition["continuity"], using → ⟨Definition["continuity"],
Lemma["distance of sum"], Definition["sum of sequences/functions"],
Lemma["mininum, less"], Lemma["mininum, greater"]⟩,
by →
PCS]

▪ ProofObject ▪

3.3 Gröbner Bases

```
Formula["Geometry", any[x, y],
  ((x^3 y^4 + 27 x y^3 - 15 x^3) = 0) ∧ ((1/5 x^2 y - 23 x y + 17) = 0) ⇒
    (((17 + -23 x y + x^2 y/5) (15 x y + x^2 y^2) +
        (-15 + x y) (-15 x^3 + 27 x y^3 + x^3 y^4)) = 0)]
```

```
Prove[Formula["Geometry"], by → GroebnerBasesProver]
```

```
▪ProofObject▪
```

3.4 Algorithmic Mathematics

A *complete implementation* of the Gröbner bases algorithm (⟶ Davenport course).

```
Definition["Exponent tuples", any[n],
  PP[n] = Functor[ET, any[p1, p2],
    $ = ⟨⟩
    (p1 ≤_ET p2) : ⇔ ∃_{i=1,..,n} (p1_i < p2_i ∧ ∀_{j=1,..,i-1} (p1_j = p2_j))
    p1 *_ET p2 := ⟨p1_i + p2_i | i=1,..,n⟩
    p1 /_ET p2 := ⟨p1_i - p2_i | i=1,..,n⟩
    (p1 |_ET p2) : ⇔ ∀_{i=1,..,n} p1_i ≤ p2_i
    lcm_ET[p1, p2] := ⟨{p1_i ⇐ p1_i ≥ p2_i ; p2_i ⇐ otherwise | i=1,..,n}⟩
  ]]
```

```
General::spell1 :
  Possible spelling error: new symbol name "lcm" is similar to existing symbol "LCM".
```

⟨1,0,3⟩ stands for $x^1\, y^0\, z^3$.

(Lexicographic ordering, parametrized domain would allow the choice of term order.)


```
Definition["Monomials", any[K, n],
  Mon[K, n] = Functor[M, any[c1, c2, p1, p2, m1, m2],
    $ = ⟨⟩
    (⟨c1, p1⟩ ≤_M ⟨c2, p2⟩) : ⇔ p1 ≤_PP[n] p2
    ⟨c1, p1⟩ +_M ⟨c2, p1⟩ := ⟨c1 +_K c2, p1⟩
    ⟨c1, p1⟩ -_M ⟨c2, p1⟩ := ⟨c1 -_K c2, p1⟩
    -_M[⟨c1, p1⟩] := ⟨-_K[c1], p1⟩
    ⟨c1, p1⟩ *_M ⟨c2, p2⟩ := ⟨c1 *_K c2, p1 *_PP[n] p2⟩
    ⟨c1, p1⟩ /_M ⟨c2, p2⟩ := ⟨c1 /_K c2, p1 /_PP[n] p2⟩
    (⟨c1, p1⟩ |_M ⟨c2, p2⟩) : ⇔ p1 |_PP[n] p2
    lcm_M[⟨c1, p1⟩, ⟨c2, p2⟩] := ⟨c1 *_K c2, lcm_PP[n][p1, p2]⟩
    is-zero_M[⟨c1, p1⟩] : ⇔ (c1 = 0_K)
  ]]
```

⟨5, ⟨1, 0, 3⟩⟩ stands for $5\, x\, z^3$.

Definition["Polynomials", any[M],
Poly[M] = Functor[P, any[c, p, r, m, m1, m2, r1, r2, p1, p2],

$ = ⟨⟩

lpp[⟨⟨c, p⟩, r⟩] := p
lcoef[⟨⟨c, p⟩, r⟩] := c
lmon[⟨m, r⟩] := m
rp[⟨m, r⟩] := ⟨r⟩
⟨⟩ +_P p := p
p +_P ⟨⟩ := p

$$\langle m1, r1\rangle +_P \langle m2, r2\rangle := \begin{cases} m2 \frown \left(\langle m1, r1\rangle +_P \langle r2\rangle\right) & \Leftarrow\ m1 <_M m2 \\ m1 \frown \left(\langle r1\rangle +_P \langle m2, r2\rangle\right) & \Leftarrow\ m2 <_M m1 \\ \text{where}\left[s = m1 +_M m2,\ \begin{cases} \langle r1\rangle +_P \langle r2\rangle & \Leftarrow\ \text{is-zero}_M[s] \\ s \frown \left(\langle r1\rangle +_P \langle r2\rangle\right) & \Leftarrow\ \text{otherwise} \end{cases}\right] & \Leftarrow\ \text{otherwise} \end{cases}$$

$\overline{+}_P[⟨⟩] := ⟨⟩$

$\overline{+}_P[p] := \left\langle \overline{+}_M[p_i] \mid_{i=1,\ldots,|p|} \right\rangle$

$⟨⟩ \overline{+}_P p := \overline{+}_P[p]$
$p \overline{+}_P ⟨⟩ := p$

$$\langle m1, r1\rangle \overline{+}_P \langle m2, r2\rangle := \begin{cases} \overline{+}_M[m2] \frown \left(\langle m1, r1\rangle \overline{+}_P \langle r2\rangle\right) & \Leftarrow\ m1 <_M m2 \\ m1 \frown \left(\langle r1\rangle \overline{+}_P \langle m2, r2\rangle\right) & \Leftarrow\ m2 <_M m1 \\ \text{where}\left[s = m1 \overline{+}_M m2,\ \begin{cases} \langle r1\rangle \overline{+}_P \langle r2\rangle & \Leftarrow\ \text{is-zero}_M[s] \\ s \frown \left(\langle r1\rangle \overline{+}_P \langle r2\rangle\right) & \Leftarrow\ \text{otherwise} \end{cases}\right] & \Leftarrow\ \text{otherwise} \end{cases}$$

$m *_P ⟨⟩ := ⟨⟩$

$m *_P p := \left\langle m *_M p_i \mid_{i=1,\ldots,|p|} \right\rangle$

$$\text{normalized}_P[p] := \text{where}\left[lc = \text{lcoef}[p], \left\langle p_i \Big/_M \left\langle lc, \left\langle 0 \mid_{j=1,\ldots,|\text{lpp}[p]|} \right\rangle \right\rangle \right\rangle \Big|_{i=1,\ldots,|p|}\right]$$

]]

$\langle\langle 5, \langle 1, 0, 3\rangle\rangle, \langle -4, \langle 0, 0, 2\rangle\rangle\rangle$ stands for $5xz^3 - 4z^2$. Monomials must be ordered appropriately. Arithmetic operations preserve the ordering.

Definition["Reduction domain", any[P, M],
ReductionDomain[P[M]] = Functor[R, any[p, S, All, r, S],

$ = ⟨⟩

partial-normal-form_R[p, S] := partial-normal-form_R[p, S, S]
partial-normal-form_R[p, ⟨⟩, All] := p
partial-normal-form_R[⟨⟩, S, All] := ⟨⟩

$$\text{partial-normal-form}_R[p, \langle r, S\rangle, \text{All}] := \begin{cases} \text{partial-normal-form}_R\left[\text{rp}[p] \overline{*}_{P[M]} \left(\left(\text{lmon}[p] /_M \text{lmon}[r]\right) *_{P[M]} \text{rp}[r]\right), \text{All}\right] & \Leftarrow\ \text{lmon}[r] \mid_M \text{lmon}[p] \\ \text{partial-normal-form}_R[p, \langle S\rangle, \text{All}] & \Leftarrow\ \text{otherwise} \end{cases}$$

$$\text{complete-normal-form}_R[⟨⟩, S] := ⟨⟩$$

$$\text{complete-normal-form}_R[p, S] := \text{where}\left[pnf = \text{partial-normal-form}_R[p, S], \begin{cases} ⟨⟩ & \Leftarrow\ pnf = ⟨⟩ \\ \text{lmon}[pnf] \frown \text{complete-normal-form}_R[\text{rp}[pnf], S] & \Leftarrow\ \text{otherwise} \end{cases}\right]$$

]]

Definition["S-Polynomial", any[f, g, K, n],
S-Polynomial[f, g, K, n] :=
where[lmf = lmon[f], lmg = lmon[g], lcmmon = $\text{lcm}_{\text{Mon}[K,n]}$[lmf, lmg],

$$\left(\left(\text{lcmmon} /_{\text{Mon}[K,n]} \text{lmon}[f]\right) *_{\text{Poly}[\text{Mon}[K,n]]} f\right) \overline{-}_{\text{Poly}[\text{Mon}[K,n]]}$$

$$\left(\left(\text{lcmmon} /_{\text{Mon}[K,n]} \text{lmon}[g]\right) *_{\text{Poly}[\text{Mon}[K,n]]} g\right)]]$$

Definition["Polynomial pairs", any[P, p],

$$\text{all-pairs}[P] := \left\langle \left(\langle P_i, P_j\rangle \mid_{j=i+1,\ldots,|P|} \right) \Big|_{i=1,\ldots,|P|} \right\rangle$$

$$\text{new-pairs}[p, P] := \left\langle \langle p, P_i\rangle \mid_{i=1,\ldots,|P|} \right\rangle$$

```
Algorithm["Gröbner Basis", any[F, K, n, P, p1, p2, r],

  Gröbner-Basis[F, K, n] := Gröbner-Basis[F, all-pairs[F], K, n]
  Gröbner-Basis[F, ⟨⟩, K, n] := F
  Gröbner-Basis[F, ⟨⟨p1, p2⟩, r⟩, K, n] :=
    where[sp = S-Polynomial[p1, p2, K, n],
      nf = complete-normal-form[sp, F, ReductionDomain[Poly[Mon[K,n]]]]
      {Gröbner-Basis[F, ⟨r⟩, K, n]                    ⇐  nf = ⟨⟩
      {Gröbner-Basis[nf ∽ F, new-pairs[nf, F] ≍ ⟨r⟩, K, n]  ⇐  otherwise ]]
```

```
Algorithm["Reduced Gröbner Basis", any[F, G, K, n],

  Deleted-Multiples[G, K, n] :=
    ⟨normalized[G_i, Poly[Mon[K,n]]] | i=1,..,|G|, ∀ j=1,..,|G| j≠i ¬(lmon[G_j] | Mon[K,n] lmon[G_i])⟩
  Interreduced[G, K, n] := ⟨complete-normal-form[G_i, ⟨G_j | j=1,..,|G| j ≠ i⟩, ReductionDomain[Poly[Mon[K,n]]]] | i=1,..,|G|⟩
  Reduced-Gröbner-Basis[F, K, n] :=
    Interreduced[Deleted-Multiples[Gröbner-Basis[F, K, n], K, n], K, n]]
```

```
Use[{Algorithm["Reduced Gröbner Basis"], Algorithm["Gröbner Basis"],
    Definition["Polynomial pairs"], Definition["S-Polynomial"],
    Definition["Reduction domain"], Definition["Polynomials"],
    Definition["Monomials"], Definition["Exponent tuples"],
    Built-in["Quantifiers"], Built-in["Connectives"], Built-in["Tuples"],
    Built-in["Sets"], Built-in["Numbers"], Built-in["Number Domains"]}]
```

```
Compute[
  S-Polynomial[⟨⟨1, ⟨1, 0⟩⟩, ⟨1, ⟨0, 1⟩⟩, ⟨−1, ⟨0, 0⟩⟩⟩, ⟨⟨1, ⟨1, 1⟩⟩, ⟨−3, ⟨0, 0⟩⟩⟩, ℚ, 2]]
⟨⟨1, ⟨0, 2⟩⟩, ⟨−1, ⟨0, 1⟩⟩, ⟨3, ⟨0, 0⟩⟩⟩
```

```
Compute[Gröbner-Basis[
    ⟨⟨⟨1, ⟨1, 0⟩⟩, ⟨1, ⟨0, 1⟩⟩, ⟨−1, ⟨0, 0⟩⟩⟩, ⟨⟨1, ⟨1, 1⟩⟩, ⟨−3, ⟨0, 0⟩⟩⟩⟩, ℚ, 2]]
⟨⟨⟨1, ⟨0, 2⟩⟩, ⟨−1, ⟨0, 1⟩⟩, ⟨3, ⟨0, 0⟩⟩⟩,
  ⟨⟨1, ⟨1, 0⟩⟩, ⟨1, ⟨0, 1⟩⟩, ⟨−1, ⟨0, 0⟩⟩⟩, ⟨⟨1, ⟨1, 1⟩⟩, ⟨−3, ⟨0, 0⟩⟩⟩⟩
```

```
GroebnerBasis[{x + y − 1, x y − 3}, {x, y}]
```

```
{(−3) + y + −y^2, (−1) + x + y}
```

```
Compute[Reduced-Gröbner-Basis[
    ⟨⟨⟨1, ⟨1, 0⟩⟩, ⟨1, ⟨0, 1⟩⟩, ⟨−1, ⟨0, 0⟩⟩⟩, ⟨⟨1, ⟨1, 1⟩⟩, ⟨−3, ⟨0, 0⟩⟩⟩⟩, ℚ, 2]]
⟨⟨⟨1, ⟨0, 2⟩⟩, ⟨−1, ⟨0, 1⟩⟩, ⟨3, ⟨0, 0⟩⟩⟩, ⟨⟨1, ⟨1, 0⟩⟩, ⟨1, ⟨0, 1⟩⟩, ⟨−1, ⟨0, 0⟩⟩⟩⟩
```



```
Compute[Reduced-Gröbner-Basis[
    ⟨⟨⟨3, ⟨1, 2⟩⟩, ⟨2, ⟨1, 1⟩⟩, ⟨1, ⟨1, 0⟩⟩, ⟨9, ⟨0, 2⟩⟩, ⟨5, ⟨0, 1⟩⟩, ⟨−3, ⟨0, 0⟩⟩⟩,
      ⟨⟨2, ⟨1, 3⟩⟩, ⟨−1, ⟨1, 1⟩⟩, ⟨−1, ⟨1, 0⟩⟩, ⟨6, ⟨0, 3⟩⟩,
        ⟨−2, ⟨0, 2⟩⟩, ⟨−3, ⟨0, 1⟩⟩, ⟨3, ⟨0, 0⟩⟩⟩,
      ⟨⟨1, ⟨1, 3⟩⟩, ⟨1, ⟨1, 2⟩⟩, ⟨3, ⟨0, 3⟩⟩, ⟨2, ⟨0, 2⟩⟩⟩⟩, ℚ, 2]] // Timing
{14.48 Second, ⟨⟨⟨1, ⟨0, 3⟩⟩, ⟨−5/2, ⟨0, 2⟩⟩, ⟨−5/2, ⟨0, 1⟩⟩⟩,
    ⟨⟨1, ⟨1, 0⟩⟩, ⟨1, ⟨0, 2⟩⟩, ⟨−3/2, ⟨0, 1⟩⟩, ⟨−3, ⟨0, 0⟩⟩⟩⟩}
```

```
GroebnerBasis[
    {3 x y^2 + 2 x y + x + 9 y^2 + 5 y − 3, 2 x y^3 − x y − x + 6 y^3 − 2 y^2 − 3 y + 3,
      x y^3 + x y^2 + 3 y^3 + 2 y^2}, {x, y}] // Timing
{0.02 Second, {−5 y + −5 y^2 + 2 y^3, 6 + −2 x + 3 y + −2 y^2}}
```

```
Use[]
```

Prove:

(Proposition (non-emptyclass)) $\forall_{A,x} \left(x \in A \bigwedge_{\text{is-reflexive}_A} \Rightarrow \text{class}_A[x] \neq \{\} \right)$,

under the assumptions:

(Definition (reflexivity)) $\forall_{A,\sim} \left(\text{is-reflexive}_A[\sim] :\Leftrightarrow \forall_x (x \in A \Rightarrow x \sim x) \right)$,

(Definition (class)) $\forall_{A,x} \left(\text{class}_A[x] := \left\{ a \mid_a a \in A \wedge a \sim x \right\} \right)$.

We assume

(1) $x_0 \in A_0 \bigwedge_{\text{is-reflexive}_{A_0}}$,

and show

(2) $\text{class}_{A_0}[x_0] \neq \{\}$.

Formula (2) means that we have to show that

(3) $\exists_{x1} (x1 \in \text{class}_{A_0}[x_0])$.

Formula (3), using (Definition (class)), is implied by:

(5) $\exists_{x1} \left(x1 \in \left\{ a \mid_a a \in A_0 \wedge a \sim x_0 \right\} \right)$.

In order to prove (5) we have to show:

(6) $\exists_{x1} (x1 \in A_0 \wedge x1 \sim x_0)$.

Now, let $x1 := x_0$. Thus, for proving (6) it is sufficient to prove:

(7) $x_0 \in A_0 \wedge x_0 \sim x_0$.

We prove the individual conjunctive parts of (7):

Proof of (7.1) $x_0 \in A_0$:

Formula (7.1) is true because it is identical to (1.1).

Proof of (7.2) $x_0 \sim x_0$:

Formula (1.2), by (Definition (reflexivity)), implies:

(8) $\forall_x (x \in A_0 \Rightarrow x \sim x)$.

Formula (7.2), using (8), is implied by:

(9) $x_0 \in A_0$.

Formula (9) is true because it is identical to (1.1).

□

■ Additional Proof Generation Information

Prove:

(Proposition (continuity))

$$\forall_{f,g,x} \left(\text{continuous}[f, x] \wedge \text{continuous}[g, x] \Rightarrow \text{continuous}[f + g, x]\right),$$

under the assumptions:

(Definition (continuity)) $$\forall_{f,x} \left(\text{continuous}[f, x] :\Leftrightarrow \forall_{\epsilon>0} \exists_{\delta>0} \forall_{y, |y-x|<\delta} \left(|f[y] - f[x]| < \epsilon\right)\right),$$

(Lemma (distance of sum)) $$\forall_{x,y,z,t,\delta,\epsilon} \left(|(x + z) - (y + t)| < \delta + \epsilon \Leftarrow (|x - y| < \delta \wedge |z - t| < \epsilon)\right),$$

(Definition (sum of sequences/functions)) $$\forall_{f,g,x} \left((f + g)[x] = f[x] + g[x]\right),$$

(Lemma (mininum, less)) $$\forall_{m,M1,M2} \left(m < M1 \wedge m < M2 \Leftrightarrow m < \min[M1, M2]\right),$$

(Lemma (mininum, greater)) $$\forall_{m,M1,M2} \left(\min[M1, M2] > m \Leftrightarrow M1 > m \wedge M2 > m\right).$$

We assume

(1) $$\text{continuous}[f_0, x_0] \wedge \text{continuous}[g_0, x_0],$$

and show

(2) $$\text{continuous}[f_0 + g_0, x_0].$$

Formula (1.1), by (Definition (continuity)), implies:

(3) $$\forall_{\epsilon>0} \exists_{\delta>0} \forall_{y, |y-x_0|<\delta} \left(|f_0[y] - f_0[x_0]| < \epsilon\right).$$

By (3), we can take an appropriate Skolem function such that

(4) $$\forall_{\epsilon>0} \left(\delta_0[\epsilon] > 0 \bigwedge \forall_{y, |y-x_0|<\delta_0[\epsilon]} \left(|f_0[y] - f_0[x_0]| < \epsilon\right)\right),$$

Formula (4) is simplified to:

(5) $$\forall_{\epsilon} \left(\epsilon > 0 \Rightarrow \delta_0[\epsilon] > 0\right) \bigwedge \forall_{y,\epsilon} \left(\epsilon > 0 \wedge |y - x_0| < \delta_0[\epsilon] \Rightarrow |f_0[y] - f_0[x_0]| < \epsilon\right).$$

Formula (1.2), by (Definition (continuity)), implies:

(6) $$\forall_{\epsilon>0} \exists_{\delta>0} \forall_{y, |y-x_0|<\delta} \left(|g_0[y] - g_0[x_0]| < \epsilon\right).$$

By (6), we can take an appropriate Skolem function such that

(7) $$\forall_{\epsilon>0} \left(\delta_1[\epsilon] > 0 \bigwedge \forall_{y, |y-x_0|<\delta_1[\epsilon]} \left(|g_0[y] - g_0[x_0]| < \epsilon\right)\right),$$

Formula (7) is simplified to:

(8) $$\forall_{\epsilon} \left(\epsilon > 0 \Rightarrow \delta_1[\epsilon] > 0\right) \bigwedge \forall_{y,\epsilon} \left(\epsilon > 0 \wedge |y - x_0| < \delta_1[\epsilon] \Rightarrow |g_0[y] - g_0[x_0]| < \epsilon\right).$$

Formula (2), using (Definition (continuity)), is implied by:

(9) $$\forall_{\epsilon>0} \exists_{\delta>0} \forall_{y, |y-x_0|<\delta} \left(|(f_0 + g_0)[y] - (f_0 + g_0)[x_0]| < \epsilon\right).$$

We assume

(10) $$\epsilon_0 > 0,$$

and show

(11) $$\exists_{\delta>0} \forall_{y, |y-x_0|<\delta} \left(|(f_0 + g_0)[y] - (f_0 + g_0)[x_0]| < \epsilon_0\right).$$

We have to find δ^{***} such that

(12) $$\delta^{***} > 0 \bigwedge \forall_{y} \left(|y - x_0| < \delta^{***} \Rightarrow |(f_0 + g_0)[y] - (f_0 + g_0)[x_0]| < \epsilon_0\right).$$

Formula (12), using (Definition (sum of sequences/functions)), is implied by:

(13) $$\delta^{***} > 0 \bigwedge \forall_{y} \left(|y - x_0| < \delta^{***} \Rightarrow |(f_0[y] + g_0[y]) - (f_0[x_0] + g_0[x_0])| < \epsilon_0\right).$$

Formula (13), using (Lemma (distance of sum)), is implied by:

(14) $$\exists_{\delta,\epsilon, \delta+\epsilon=\epsilon_0} \left(\delta^{***} > 0 \bigwedge \forall_{y} \left(|y - x_0| < \delta^{***} \Rightarrow |f_0[y] - f_0[x_0]| < \delta \wedge |g_0[y] - g_0[x_0]| < \epsilon\right)\right).$$

We have to find δ^{***}, δ^{****}, and ϵ^{**} such that

(15) $$(\delta^{****} + \epsilon^{**} = \epsilon_0) \bigwedge \delta^{***} > 0 \bigwedge \forall_{y} \left(|y - x_0| < \delta^{***} \Rightarrow |f_0[y] - f_0[x_0]| < \delta^{****} \wedge |g_0[y] - g_0[x_0]| < \epsilon^{**}\right).$$

Formula (15), using (8.2), is implied by:

$$(\delta^{****} + \epsilon^{**} = \epsilon_0) \bigwedge \delta^{***} > 0 \bigwedge \forall_{y} \left(|y - x_0| < \delta^{***} \Rightarrow \epsilon^{**} > 0 \wedge |y - x_0| < \delta_1[\epsilon^{**}] \wedge |f_0[y] - f_0[x_0]| < \delta^{****}\right),$$

which, using (5.2), is implied by:

$$(\delta^{****} + \epsilon^{**} = \epsilon_0) \bigwedge \delta^{***} > 0 \bigwedge \forall_{y} \left(|y - x_0| < \delta^{***} \Rightarrow \delta^{****} > 0 \wedge \epsilon^{**} > 0 \wedge |y - x_0| < \delta_0[\delta^{****}] \wedge |y - x_0| < \delta_1[\epsilon^{**}]\right),$$

which, using (Lemma (mininum, less)), is implied by:

(16) $$(\delta^{****} + \epsilon^{**} = \epsilon_0) \bigwedge \delta^{***} > 0 \bigwedge \forall_{y} \left(|y - x_0| < \delta^{***} \Rightarrow \delta^{****} > 0 \wedge \epsilon^{**} > 0 \wedge |y - x_0| < \min[\delta_0[\delta^{****}], \delta_1[\epsilon^{**}]]\right).$$

Formula (16) is implied by

(17) $(\delta^{****} + \epsilon^{**} = \epsilon_0) \bigwedge \delta^{***} > 0 \bigwedge \delta^{****} > 0 \bigwedge \epsilon^{**} > 0 \bigwedge \forall_y (|y - x_0| < \delta^{***} \Rightarrow |y - x_0| < \min[\delta_0[\delta^{****}], \delta_1[\epsilon^{**}]])$.

Partially solving it, formula (17) is implied by

(18) $(\delta^{****} + \epsilon^{**} = \epsilon_0) \wedge \min[\delta_0[\delta^{****}], \delta_1[\epsilon^{**}]] > 0 \wedge \delta^{****} > 0 \wedge \epsilon^{**} > 0 \wedge (\delta^{***} = \min[\delta_0[\delta^{****}], \delta_1[\epsilon^{**}]])$.

Formula (18), using (Lemma (mininum, greater)), is implied by:

$(\delta^{***} = \min[\delta_0[\delta^{****}], \delta_1[\epsilon^{**}]]) \wedge (\delta^{****} + \epsilon^{**} = \epsilon_0) \wedge \delta^{****} > 0 \wedge \epsilon^{**} > 0 \wedge \delta_0[\delta^{****}] > 0 \wedge \delta_1[\epsilon^{**}] > 0$,

which, using (8.1), is implied by:

$(\delta^{***} = \min[\delta_0[\delta^{****}], \delta_1[\epsilon^{**}]]) \wedge (\delta^{****} + \epsilon^{**} = \epsilon_0) \wedge \delta^{****} > 0 \wedge \epsilon^{**} > 0 \wedge \delta_0[\delta^{****}] > 0$,

which, using (5.1), is implied by:

(19) $(\delta^{***} = \min[\delta_0[\delta^{****}], \delta_1[\epsilon^{**}]]) \wedge (\delta^{****} + \epsilon^{**} = \epsilon_0) \wedge \delta^{****} > 0 \wedge \epsilon^{**} > 0$.

Now,

$(\delta^{****} + \epsilon^{**} = \epsilon_0) \wedge \delta^{****} > 0 \wedge \epsilon^{**} > 0$

can be solved for δ^{****} and ϵ^{**} by a call to Collins cad-method yielding a sample solution

$\delta^{****} \leftarrow \frac{\epsilon_0}{2}$,

$\epsilon^{**} \leftarrow \frac{\epsilon_0}{2}$.

Furthermore, we can immediately solve

$\delta^{***} = \min[\delta_0[\delta^{****}], \delta_1[\epsilon^{**}]]$

for δ^{***} by taking

$\delta^{***} \leftarrow \min[\delta_0[\frac{\epsilon_0}{2}], \delta_1[\frac{\epsilon_0}{2}]]$.

Hence formula (19) is solved, and we are done.

□

■ Additional Proof Generation Information

Prove:

(Formula (Geometry))

$\forall_{x,y} \left(\left(\left(x^3 * y^4 + 27 * xy^3\right) - 15 * x^3\right) = 0\right) \wedge \left(\left(\frac{1}{5} * x^2 * y - 23 * x * y\right) + 17 = 0\right) \Rightarrow$,
$\left(\left(17 + (-23) * x * y + \frac{x^2 * y}{5}\right) * (15 * x * y + x^2 * y^2) + ((-15) + x * y) * ((-15) * x^3 + 27 * xy^3 + x^3 * y^4) = 0\right)\right)$

with no assumptions.

Proved.

The Theorem is proved by the Groebner Bases method.

The formula in the scope of the universal quantifier is transformed into an equivalent formula that is a conjunction of disjunctions of equalities and negated equalities. The universal quantifier can then be distributed over the individual parts of the conjunction. By this, we obtain:

Independent proof problems:

(Formula (Geometry).1)

$\forall_{x,y} ((((-405) * xy^3 + 225 * x^3 + 255 * x * y + 27 * xy^3 * x * y + (-15) * x^4 * y + (-328) * x^2 * y^2 + 3 * x^3 * y^2 + (-23) * x^3 * y^3 + \frac{1}{5} * x^4 * y^3 + (-15) * x^3 * y^4 + x^4 * y^5 = 0) \vee 27 * xy^3 + (-15) * x^3 + x^3 * y^4 \neq 0 \vee 17 + (-23) * x * y + \frac{1}{5} * x^2 * y \neq 0)$

We now prove the above individual problems separately:

Proof of (Formula (Geometry).1):

This proof problem has the following structure:

(Formula (Geometry).1.structure) $\forall_{x,y} (\text{Poly}[1] \neq 0 \vee \text{Poly}[2] \neq 0 \vee (\text{Poly}[3] = 0))$,

where

$\text{Poly}[1] = 17 + (-23) * x * y + \frac{1}{5} * x^2 * y$
$\text{Poly}[2] = 27 * xy^3 + (-15) * x^3 + x^3 * y^4$
$\text{Poly}[3] = (-405) * xy^3 + 225 * x^3 + 255 * x * y + 27 * xy^3 * x * y + (-15) * x^4 * y + (-328) * x^2 * y^2$

(Formula (Geometry).1.structure) is equivalent to

(Formula (Geometry).1.implication) $\forall_{x,y} ((\text{Poly}[1] = 0) \wedge (\text{Poly}[2] = 0) \Rightarrow (\text{Poly}[3] = 0))$.

(Formula (Geometry).1.implication) is equivalent to

(Formula (Geometry).1.not-exists) $\nexists_{x,y} (((\text{Poly}[1] = 0) \wedge (\text{Poly}[2] = 0)) \wedge \text{Poly}[3] \neq 0)$.

By introducing the slack variable(s)

{ξ}

(Formula (Geometry).1.not–exists) is transformed into the equivalent formula

(Formula (Geometry).1.not–exists–slack)

$$\nexists_{x,y,\xi} \left(((\text{Poly}[1] = 0) \wedge (\text{Poly}[2] = 0)) \wedge ((-1) + \xi\, \text{Poly}[3] = 0)\right).$$

Hence, we see that the proof problem is transformed into the question on whether or not a system of polynomial equations has a solution or not. This question can be answered by checking whether or not the (reduced) Groebner basis of

```
{Poly[1], Poly[2], (-1) + ξ Poly[3]}
```

is exactly {1}.

Hence, we compute the Groebner basis for the following polynomial list:

$$\{(-1) + 225\, x^3\, \xi + -405\, x y^3\, \xi + 255\, x\, y\, \xi + -15\, x^4\, y\, \xi + 27\, x\, x y^3\, y\, \xi + -328\, x^2\, y^2\, \xi + 3\, x^3\, y^2\, \xi + -23\, x^3\, y^3\, \xi + \tfrac{1}{5}\, x^4\, y^3\, \xi + -15\, x^3\, y^4\, \xi + x^4\, y^5\, \xi,\ -15\, x^3 + 27\, x y^3 + x^3\, y^4,\ 17 + -23\, x\, y + \tfrac{x^2\, y}{5}\}$$

The Groebner basis:

```
{1}
```

Hence, (Formula (Geometry).1) is proved.

Since all of the individual subtheorems are proved, the original formula is proved.

□

■ Additional Proof Generation Information

Prove:

(Proposition (in own class)) $\forall_{A,x} \left(x \in A \bigwedge_{\text{is-reflexive}_A} \Rightarrow x \in \text{class}_A[x]\right)$,

under the assumptions:

(Definition (reflexivity)) $\forall_{A,\sim} \left(\text{is-reflexive}_A[\sim] :\Leftrightarrow \forall_x (x \in A \Rightarrow x \sim x)\right)$,

(Definition (class)) $\forall_{A,x} \left(\text{class}_A[x] := \left\{a \mid_a a \in A \wedge a \sim x\right\}\right)$.

We assume

(1) $x_0 \in A_0 \bigwedge_{\text{is-reflexive}_{A_0}}$,

and show

(2) $x_0 \in \text{class}_{A_0}[x_0]$.

Formula (2), using (Definition (class)), is implied by:

(4) $x_0 \in \left\{a \mid_a a \in A_0 \wedge a \sim x_0\right\}$.

In order to prove (4) we have to show:

(5) $x_0 \in A_0 \wedge x_0 \sim x_0$.

We prove the individual conjunctive parts of (5):

Proof of (5.1) $x_0 \in A_0$:

Formula (5.1) is true because it is identical to (1.1).

Proof of (5.2) $x_0 \sim x_0$:

Formula (1.2), by (Definition (reflexivity)), implies:

(6) $\forall_x (x \in A_0 \Rightarrow x \sim x)$.

Formula (5.2), using (6), is implied by:

(7) $x_0 \in A_0$.

Formula (7) is true because it is identical to (1.1).

□

■ Additional Proof Generation Information

Prove:

(Lemma (intersection indexed union)) $\left(\bigcup_{\substack{i\\ i\in\mathcal{I}}} A_i\right) \cap \left(\bigcup_{\substack{j\\ j\in\mathcal{J}}} B_j\right) = \bigcup_{\substack{i\\ i\in\mathcal{I}}} \bigcup_{\substack{j\\ j\in\mathcal{J}}} A_i \cap B_j$,

with no assumptions.

We show (Lemma (intersection indexed union)) by mutual inclusion:

$\subseteq$: We assume

(1) $il_0 \in \left(\bigcup_{\substack{i\\ i\in\mathcal{I}}} A_i\right) \cap \left(\bigcup_{\substack{j\\ j\in\mathcal{J}}} B_j\right)$

and show:

(2) $il_0 \in \bigcup_{\substack{i\\ i\in\mathcal{I}}} \bigcup_{\substack{j\\ j\in\mathcal{J}}} A_i \cap B_j$.

From what we already know follows:

From (1) we can infer

(4) $il_0 \in \bigcup_{\substack{i\\ i\in\mathcal{I}}} A_i$,

(5) $il_0 \in \bigcup_{\substack{j\\ j\in\mathcal{J}}} B_j$,

(3) $\left(\bigcup_{\substack{i\\ i\in\mathcal{I}}} A_i\right) \cap \left(\bigcup_{\substack{j\\ j\in\mathcal{J}}} B_j\right) \neq \{\}$.

From what we already know follows:

From (4) we know by definition of the $\bigcup$-quantifier that we can choose an appropriate value such that

(7) $il_0 \in A_{i_0}$,

(6) $i_0 \in \mathcal{I}$.

From (5) we know by definition of the $\bigcup$-quantifier that we can choose an appropriate value such that

(9) $il_0 \in B_{j_0}$,

(8) $j_0 \in \mathcal{J}$.

In order to show (2) we have to show

(14) $\exists_i \left(il_0 \in \bigcup_{\substack{j\\ j\in\mathcal{J}}} A_i \cap B_j \bigwedge i \in \mathcal{I}\right)$.

Now, let $i := i_0$. Thus, for proving (14) it is sufficient to prove:

(15) $il_0 \in \bigcup_{\substack{j\\ j\in\mathcal{J}}} A_{i_0} \cap B_j \bigwedge i_0 \in \mathcal{I}$.

We prove the individual conjunctive parts of (15):

Proof of (15.1) $il_0 \in \bigcup_{\substack{j\\ j\in\mathcal{J}}} A_{i_0} \cap B_j$:

In order to show (15.1) we have to show

(16) $\exists_j (il_0 \in A_{i_0} \cap B_j \wedge j \in \mathcal{J})$.

Now, let $j := j_0$. Thus, for proving (16) it is sufficient to prove:

(17) $il_0 \in A_{i_0} \cap B_{j_0} \wedge j_0 \in \mathcal{J}$.

We prove the individual conjunctive parts of (17):

Proof of (17.1) $il_0 \in A_{i_0} \cap B_{j_0}$:

We prove (17.1) by splitting up the intersection into its individual components:

We have to prove:

(18) $il_0 \in A_{i_0}$.

Formula (18) is true because it is identical to (7).

We have to prove:

(19) $il_0 \in B_{j_0}$.

Formula (19) is true because it is identical to (9).

Proof of (17.2) $j_0 \in \mathcal{J}$:

Formula (17.2) is true because it is identical to (8).

Proof of (15.2) $i_0 \in \mathcal{I}$:

Formula (15.2) is true because it is identical to (6).

$\supseteq$: Now we assume

(2) $il_0 \in \bigcup_{\substack{i\\ i\in\mathcal{I}}} \bigcup_{\substack{j\\ j\in\mathcal{J}}} A_i \cap B_j$

and show:

(1) $il_0 \in \left(\bigcup_{\substack{i\\ i\in\mathcal{I}}} A_i\right) \cap \left(\bigcup_{\substack{j\\ j\in\mathcal{J}}} B_j\right)$.

From what we already know follows:

From (2) we know by definition of the $\bigcup$-quantifier that we can choose an appropriate value such that

(21) $il_0 \in \bigcup_{\substack{j \\ j \in \mathcal{J}}} A_{i_1} \bigcap B_j$,

(20) $i_1 \in \mathcal{I}$.

From what we already know follows:

From (21) we know by definition of the $\bigcup$-quantifier that we can choose an appropriate value such that

(23) $il_0 \in A_{i_1} \bigcap B_{j_1}$,

(22) $j_1 \in \mathcal{J}$.

From (20) we can infer

(24) $\mathcal{I} \neq \{\}$.

From what we already know follows:

From (23) we can infer

(26) $il_0 \in A_{i_1}$,

(27) $il_0 \in B_{j_1}$,

(25) $A_{i_1} \bigcap B_{j_1} \neq \{\}$.

From (22) we can infer

(28) $\mathcal{J} \neq \{\}$.

We prove (1) by splitting up the intersection into its individual components:

We have to prove:

(31) $il_0 \in \bigcup_{\substack{i \\ i \in \mathcal{I}}} A_i$.

In order to show (31) we have to show

(33) $\exists_i \, (il_0 \in A_i \wedge i \in \mathcal{I})$.

Now, let $i := i_1$. Thus, for proving (33) it is sufficient to prove:

(34) $il_0 \in A_{i_1} \wedge i_1 \in \mathcal{I}$.

We prove the individual conjunctive parts of (34):

Proof of (34.1) $il_0 \in A_{i_1}$:

Formula (34.1) is true because it is identical to (26).

Proof of (34.2) $i_1 \in \mathcal{I}$:

Formula (34.2) is true because it is identical to (20).

We have to prove:

(32) $il_0 \in \bigcup_{\substack{j \\ j \in \mathcal{J}}} B_j$.

In order to show (32) we have to show

(35) $\exists_j \, (il_0 \in B_j \wedge j \in \mathcal{J})$.

Now, let $j := j_1$. Thus, for proving (35) it is sufficient to prove:

(36) $il_0 \in B_{j_1} \wedge j_1 \in \mathcal{J}$.

We prove the individual conjunctive parts of (36):

Proof of (36.1) $il_0 \in B_{j_1}$:

Formula (36.1) is true because it is identical to (27).

Proof of (36.2) $j_1 \in \mathcal{J}$:

Formula (36.2) is true because it is identical to (22).

□

■ Additional Proof Generation Information

Prove:

(Proposition (intersection powerset)) $\bigcap \mathcal{P}[A] = \{\}$,

with no assumptions.

We have to prove (Proposition (intersection powerset)), hence, we have to show:

(1) $A1_0 \notin \bigcap \mathcal{P}[A]$.

We prove (1) by contradiction.

We assume

(2) $A1_0 \in \bigcap \mathcal{P}[A]$,

and show a contradiction.

From what we already know follows:

From (2) we can infer

(3) $\forall_{A2} (A2 \in \mathcal{P}[A] \Rightarrow A1_0 \in A2)$.

From what we already know follows:

From (3) we can infer

(4) $A1_0 \in \{\}$,

(5) $A1_0 \in A$.

Using available computation rules we can simplify the knowledge base:

Formula (4) simplifies to

(6) False,

From what we already know follows:

From (5) we can infer

(7) $A \neq \{\}$.

Formula (a contradiction) is true because the assumption (6) is false.

□

■ Additional Proof Generation Information

Prove:

(Theorem (Not over And)) $\forall_{A,B} (\neg A \vee \neg B \Rightarrow \neg (A \wedge B))$,

with no assumptions.

For proving (Theorem (Not over And)) we take all variables arbitrary but fixed and prove:

(1) $\neg A_0 \vee \neg B_0 \Rightarrow \neg (A_0 \wedge B_0)$.

We prove (1) by the deduction rule.

We assume

(2) $\neg A_0 \vee \neg B_0$

and show

(3) $\neg (A_0 \wedge B_0)$.

We prove (3) by contradiction.

We assume

(4) $A_0 \wedge B_0$,

and show a contradiction.

From (4.1) and (2) we obtain by resolution

(5) $\neg B_0$.

Formula (a contradiction) is proved because (4.2) and (5) are contradictory.

□

■ Additional Proof Generation Information

Prove:

(Proposition (union powerset)) $\forall_{A} (\bigcup \mathcal{P}[A] = A)$,

with no assumptions.

For proving (Proposition (union powerset)) we take all variables arbitrary but fixed and prove:

(1) $\bigcup \mathcal{P}[A_0] = A_0$.

We show (1) by mutual inclusion:

⊆: We assume

(2) $A1_0 \in \bigcup \mathcal{P}[A_0]$

and show:

(3) $A1_0 \in A_0$.

From what we already know follows:

From (2) we know by definition of the big ⋃-operator that we can choose an appropriate value such that

(4) $A2_0 \in \mathcal{P}[A_0]$,

(5) $A1_0 \in A2_0$.

From what we already know follows:

From (4) we can infer

(6) $A2_0 \subseteq A_0$.

From (5) we can infer

(7) $A2_0 \neq \{\}$.

From what we already know follows:

From (6) we can infer

(8) $\forall_{A21} (A21 \in A2_0 \Rightarrow A21 \in A_0)$.

Formula (5), by (8), implies:

(9) $A1_0 \in A_0$.

Formula (3) is true because it is identical to (9).

⊇: Now we assume

(3) $A1_0 \in A_0$

and show:

(2) $A1_0 \in \bigcup \mathcal{P}[A_0]$.

In order to show (2) we have to show

(11) $\exists_{A3} (A1_0 \in A3 \wedge A3 \in \mathcal{P}[A_0])$.

Now, let $A3 := A_0$. Thus, for proving (11) it is sufficient to prove:

(12) $A1_0 \in A_0 \wedge A_0 \in \mathcal{P}[A_0]$.

We prove the individual conjunctive parts of (12):

Proof of (12.1) $A1_0 \in A_0$:

Formula (12.1) is true because it is identical to (3).

Proof of (12.2) $A_0 \in \mathcal{P}[A_0]$:

For proving (12.2) we choose

(13) $A4_0 \in A_0$,

and show:

(14) $A4_0 \in A_0$.

Formula (14) is true because it is identical to (13).

□

■ Additional Proof Generation Information

Prove:

(Proposition (union powerset)) $\forall_A (\bigcup \mathcal{P}[A] = A)$,

with no assumptions.

As there are several methods which can be applied, we have different choices to proceed with the proof.

Alternative proof 1: proved

For proving (Proposition (union powerset)) we take all variables arbitrary but fixed and prove:

(1) $\bigcup \mathcal{P}[A_0] = A_0$.

We show (1) by mutual inclusion:

$\subseteq$: We assume

(2) $A1_0 \in \bigcup \mathcal{P}[A_0]$

and show:

(3) $A1_0 \in A_0$.

From what we already know follows:

From (2) we know by definition of the big $\bigcup$-operator that we can choose an appropriate value such that

(4) $A2_0 \in \mathcal{P}[A_0]$,

(5) $A1_0 \in A2_0$.

From what we already know follows:

From (4) we can infer

(6) $A2_0 \subseteq A_0$.

From (5) we can infer

(7) $A2_0 \neq \{\}$.

From what we already know follows:

From (6) we can infer

(8) $\forall_{A21} (A21 \in A2_0 \Rightarrow A21 \in A_0)$.

As there are several methods which can be applied, we have different choices to proceed with the proof.

Alternative proof 1: proved

Formula (5), by (8), implies:

(9) $A1_0 \in A_0$.

Formula (3) is true because it is identical to (9).

Alternative proof 2: pending

Pending proof of (3).

$\supseteq$: Now we assume

(3) $A1_0 \in A_0$

and show:

(2) $A1_0 \in \bigcup \mathcal{P}[A_0]$.

From what we already know follows:

From (3) we can infer

(10) $A_0 \neq \{\}$.

In order to show (2) we have to show

(11) $\exists_{A3} (A1_0 \in A3 \wedge A3 \in \mathcal{P}[A_0])$.

We have no means to solve (11).

As there are several methods which can be applied, we have different choices to proceed with the proof.

Alternative proof 1: proved

Because parts of the knowledge base match a part of (11), we try to find an instance of (11).

Alternative proof 1: proved

Now, let $A3 := A_0$. Thus, for proving (11) it is sufficient to prove:

(12) $A1_0 \in A_0 \wedge A_0 \in \mathcal{P}[A_0]$.

As there are several methods which can be applied, we have different choices to proceed with the proof.

Alternative proof 1: proved

We prove the individual conjunctive parts of (12):

Proof of (12.1) $A1_0 \in A_0$:

Formula (12.1) is true because it is identical to (3).

Proof of (12.2) $A_0 \in \mathcal{P}[A_0]$:

For proving (12.2) we choose

(13) $A4_0 \in A_0$,

and show:

(14) $A4_0 \in A_0$.

Formula (14) is true because it is identical to (13).

Alternative proof 2: pending

Pending proof of (12).

Alternative proof 3: pending

Pending proof of (12).

Alternative proof 2: pending

Pending proof of (11).

Alternative proof 2: pending

Pending proof of (11).

Alternative proof 3: pending

Pending proof of (11).

Alternative proof 2: pending

Pending proof of (Proposition (union powerset)).

□

■ Additional Proof Generation Information