Bonnes démonstrations en déduction modulo

Soutenance de thèse

Guillaume Burel

Université Henri Poincaré – Loria

23 mars 2009
Introduction

Motivation

How to be sure of a complex mathematical proof? (for instance: 4-color theorem)

How to certify complex software?

⇒ formalizing and automating proofs
Pure logic: well-studied (Frege, Hilbert, Gentzen, etc.)

But proofs are generally done within a theory
  ▶ first-order arithmetic
  ▶ pointer arithmetic
  ▶ etc.

How to present these theories to get better mechanized proof system?
Pure logic: well-studied (Frege, Hilbert, Gentzen, etc.)

But proofs are generally done within a theory
  ▶ first-order arithmetic
  ▶ pointer arithmetic
  ▶ etc.

How to present these theories to get better mechanized proof system?

Standard way of dealing with theories: axiomatization
  ▶ For instance, Peano’s axioms for first-order arithmetic
  ▶ Not adapted for proof search!
Introduction

1+1=2

In $\Gamma$:

\[ \forall x, x + O = x \]
\[ \forall x \ y, x + s(y) = s(x + y) \]
\[ \forall x \ y, x = y \Rightarrow X(x) \Rightarrow X(y) \]

\[ \therefore \]
\[ \Gamma, 1 + 1 = s(1 + O) \vdash 1 + 1 = s(1 + O), 1 + 1 = 2 \]
\[ \Rightarrow \]
\[ \Gamma \vdash 1 + 1 = s(1 + O), 1 + 1 = 2 \]
\[ \therefore \]
\[ \Gamma, 1 + 1 = 2 \vdash 1 + 1 = 2 \]

\[ \therefore \]
\[ \Gamma, 1 + O = 1 \vdash 1 + O = 1, 1 + 1 = 2 \]
\[ \Rightarrow \]
\[ \Gamma \vdash 1 + O = 1, 1 + 1 = 2 \]
\[ \therefore \]
\[ \Gamma, 1 + O = 1 \Rightarrow 1 + 1 = s(1 + O) \Rightarrow 1 + 1 = 2 \vdash 1 + 1 = 2 \]
\[ \therefore \]
\[ \Gamma \vdash 1 + 1 = 2 \]
Introduction

1+1=2

In $\Gamma$:

$$\forall x, \ x + O = x$$

$$\forall x \ y, \ x + s(y) = s(x + y)$$

$$\forall x \ y, \ x = y \Rightarrow X(x) \Rightarrow X(y)$$

$$\Gamma \vdash 1 + 1 = s(1 + O) \Rightarrow 1 + 1 = s(1 + O), 1 + 1 = 2$$

$$\Rightarrow \Gamma \vdash 1 + 1 = s(1 + O), 1 + 1 = 2$$

$$\Gamma, 1 + 1 = 2 \vdash 1 + 1 = 2$$

$$\Gamma \vdash 1 + 1 = 2$$
Other approaches

- **Satisfiability Modulo Theory**: efficient proof search methods, not generic
  
  DPLL($T$) [Ganzinger, Hagen, Nieuwenhuis, Oliveras and Tinelli, 2004]
Introduction

Other approaches

- Satisfiability Modulo Theory: efficient proof search methods, not generic
  DPLL(\(T\)) [Ganzinger, Hagen, Nieuwenhuis, Oliveras and Tinelli, 2004]

- Dependent and Inductive Types: universal, hard to automatize
  Coq, Isabelle, etc.
Other approaches

- Satisfiability Modulo Theory: efficient proof search methods, not generic
  DPLL($T$) [Ganzinger, Hagen, Nieuwenhuis, Oliveras and Tinelli, 2004]

- Dependent and Inductive Types: universal, hard to automatize
  Coq, Isabelle, etc.

- Deduction Modulo and Superdeduction
  [Dowek, Hardin and Kirchner, 2003, Wack, 2005]
Poincaré’s principle

In a proof, distinguish deduction from computation to better combine them.

Deduction modulo: inference rules (deduction) are applied modulo a congruence (computation).

Universal model for computation: rewriting $\rightsquigarrow$ congruence based on a rewrite system over terms and formulæ.
Introduction

Example

\[
x + O \rightarrow x \\
x + s(y) \rightarrow s(x + y)
\]

\[
O = O \rightarrow \top \\
s(x) = s(y) \rightarrow x = y
\]

\[
1 + 1 = 2 \rightarrow s(1 + O) = 2 \rightarrow s(1) = 2 \rightarrow^+ O = O \rightarrow \top
\]

\[
\neg \top \rightarrow 1 + 1 = 2
\]
Compiling theories

\[ \text{Max}(x, a) \rightarrow x \in a \land \forall y, \ y \in a \Rightarrow y \leq x \]
Compiling theories

\[ \text{Max}(x, a) \rightarrow x \in a \land \forall y, \ y \in a \Rightarrow y \leq x \]

\[\Gamma, y \in b \vdash y \leq t \]

\[\Gamma, y \in b \vdash \forall y, \ y \in b \Rightarrow y \leq t \]

\[\Gamma, y \in b \vdash y \leq t \]

\[\Gamma \vdash t \in b \land \forall y, \ y \in b \Rightarrow y \leq t \]

\[\Gamma \vdash t \in b \land \forall y, \ y \in b \Rightarrow y \leq t \]

\[\Gamma \vdash \text{Max}(t, b) \]

\[\Gamma \vdash \text{Max}^\text{def} \quad \Gamma \vdash x \in a \quad \Gamma, y \in a \vdash y \leq x \]

\[\Gamma \vdash \text{Max}(x, a) \]
New rules (superrules) from a proposition rewrite system

- Natural deduction $\rightsquigarrow$ supernatural deduction
  [Wack, 2005]
  Introduction and elimination superrules

- Sequent calculus $\rightsquigarrow$ extensible sequent calculus
  [Brauner, Houtmann and Kirchner, 2007]
  Left and right superrules

Term rewrite rules are still applied modulo
Goal

How do deduction modulo and superdeduction help produce better proofs from the mechanised-theorem-proving viewpoint?

1. more direct

2. shorter

3. universal
How do deduction modulo and superdeduction help produce better proofs from the mechanised-theorem-proving viewpoint?

1. more direct: Cut admissibility

2. shorter

3. universal
How do deduction modulo and superdeduction help produce better proofs from the mechanised-theorem-proving viewpoint?

1. more direct: Cut admissibility

2. shorter: Proof length

3. universal
Goal

How do deduction modulo and superdeduction help produce better proofs from the mechanised-theorem-proving viewpoint?

1 more direct: Cut admissibility

2 shorter: Proof length

3 universal: Logical framework
Outline

1 Cut admissibility
   - Example
   - Undecidability
   - A completion procedure
   - Implementation

2 Proof length

3 Logical framework
Cut admissibility

The cut rule

\[ \Gamma, P \vdash \Delta \quad \Gamma \vdash P, \Delta \]

\[ \vdash \frac{\Gamma, P \vdash \Delta \quad \Gamma \vdash P, \Delta}{\Gamma \vdash \Delta} \]

Proof search procedures complete iff cut admissible.

Without modulo, cut admissible (Gentzen’s *Hauptsatz*)
Cut admissibility

Inadmissibility in deduction modulo

\[ A \rightarrow A \Rightarrow B \]

Let us search a “minimal” counter-example:
Inadmissibility in deduction modulo

Let us search a “minimal” counter-example:

\[ A \rightarrow A \Rightarrow B \]

\[
\begin{array}{c}
\vdash \neg A, A \Rightarrow B \\
\ll \neg A \\
\end{array}
\]

\[
\begin{array}{c}
A \Rightarrow B, A \vdash \\
\neg A \\
\end{array}
\]

\[
\begin{array}{c}
\neg A, A \Rightarrow B \vdash \\
\neg A \\
\end{array}
\]
Cut admissibility

Inadmissibility in deduction modulo

\[ A \rightarrow A \Rightarrow B \]

Let us search a “minimal” counter-example:

\[ \Rightarrow \quad \begin{array}{c}
A, B \quad \leftarrow \\
\Rightarrow B, A \quad \leftarrow \\
\Rightarrow A \quad \leftarrow 
\end{array} \]

\[ \Rightarrow \quad \begin{array}{c}
A \quad \leftarrow \\
A, A \Rightarrow B \quad \leftarrow \\
A \quad \leftarrow 
\end{array} \]
Inadmissibility in deduction modulo

\[ A \rightarrow A \Rightarrow B \]

Let us search a “minimal” counter-example:
**Theorem 1** ([LFCS07]).

The problem:

*Given a rewrite system $\mathcal{R}$, does the sequent calculus modulo $\mathcal{R}$ admits cut?*

is undecidable

*Sketch of proof:* $P$ valid iff the sequent calculus modulo $A \rightarrow A \Rightarrow P$ admits cut
Cut admissibility

Completion

Recover confluence using standard completion
[Knuth and Bendix, 1970]

Complete $A \rightarrow A \Rightarrow B$ with $B \rightarrow \top$: cut admissibility recovered

If only terms are rewritten: cut admissibility $=$ confluence
[Dowek, 2003]

If propositions are rewritten: need for a generalization of standard completion
Cut admissibility

Basic mechanism of completion (w/o simplification)

Add $s \rightarrow t$

Confluent?

- Yes → Return $\mathcal{R}$
- No → Critical pair

$s \leftarrow \mathcal{R} \rightarrow t$
Basic mechanism of completion (w/o simplification)

- Cut admissibility

A completion procedure

1. Good property?
   - yes: Return $\mathcal{R}$
   - no: critical proof
     - add rules for building a proof smaller than $p$
Abstract canonical systems
[Dershowitz and Kirchner, 2006, Bonacina and Dershowitz, 2007]

Order on proofs
⇝ critical proofs (minimal counter-examples)
⇝ completion procedure

Instances: ground completion, standard completion, Moore families, Horn theories, ...
Cut admissibility

Deduction modulo as an ACS

Polarized unfolding sequent calculus:

\[
\begin{align*}
\Gamma, A, P &\vdash \Delta & A \to - P \\
\Gamma, A &\vdash \Delta & A \to + P
\end{align*}
\]

Equivalent to the sequent calculus modulo, especially w.r.t. cuts

Order on proofs: RPO with precedence \(\preceq > \precsim > r\) and

\[
\preceq(A \Rightarrow B) > \preceq(A)
\]

Well adapted to the cut elimination procedure

If the completion terminates, the limit admits cut
Cut admissibility

Critical proofs

\[
\begin{align*}
\Gamma, A, P \vdash \Delta & \quad \pi \quad \frac{\pi}{\Gamma, A \vdash \Delta} \quad A \rightarrow P \\
\Gamma, A \vdash \Delta & \quad \pi' \quad \frac{\pi'}{\Gamma \vdash Q, A, \Delta} \quad A \rightarrow Q
\end{align*}
\]

where

- \( \pi \) and \( \pi' \) without cut
- \( \pi \) and \( \pi' \) without useless application of rules
- \( \pi \) and \( \pi' \) apply \( \not\vdash \) an atomic formulæ only
- \( \Gamma \) contains only atomic or universally quantified formulæ
  \( \not= A \)
- Dual for \( \Delta \)
- All formulæ in \( \Gamma, \Delta \) are used somewhere
Cut admissibility

Completing formulæ

Find a proof smaller than a critical proof of $\Gamma \vdash \Delta$
$\iff$ Find a rewrite system $\mathcal{R}$ s.t. $\Gamma \vdash_{\mathcal{R}} \Delta$ w/o cut

An algorithm $Rew$ from sequents to rewrite systems

**Theorem 2.**

$\Theta \vdash P \iff \vdash_{Rew(\{\vdash H : H \in \Theta\})} P$

Transforms axiomatic presentations of a theory into rewrite systems
Cut admissibility

Search for critical proofs

\[
\begin{array}{c}
\Gamma, A, P \vdash \Delta \\
\Gamma, A \vdash \Delta \\
\end{array}
\quad \pi \\
\frac{\Gamma, A \vdash \Delta}{A \rightarrow P}
\]

\[
\begin{array}{c}
\Gamma, Q, A, \Delta \\
\Gamma \vdash A, \Delta \\
\end{array}
\quad \pi' \\
\frac{\Gamma \vdash Q, A, \Delta}{A \rightarrow Q}
\]
Cut admissibility

Search for critical proofs

\(\pi\)
\[\Gamma, A, P \vdash \Delta\]

\(\pi'\)
\[\Gamma \vdash Q, A, \Delta\]
Search for critical proofs

\[ \pi \]

\[ A, P \vdash \]

\[ \pi' \]

\[ \vdash Q, A \]

Search for a cut-free proof, complete branch respecting conditions of critical proofs to find \( \Gamma, \Delta \)
Search for critical proofs

\[ \pi \]
\[ A, P \vdash \]
\[ \pi' \]
\[ \vdash Q, A \]

Search for a cut-free proof, complete branch respecting conditions of critical proofs to find \( \Gamma, \Delta \)

Implementation of the tableaux method TaMed
[Bonichon and Hermant, 2006]
Contributions [LFCS07]

- Undecidability of cut admissibility in deduction modulo
- Completion procedure to recover it
- Algorithm to transform axiomatic presentations into rewrite systems used modulo
- Implementation in TOM/OCaml
Outline

1. Cut admissibility

2. Proof length
   - Motivation
   - First results
   - Application to higher-order arithmetic

3. Logical framework
Speed-ups in higher-order arithmetic

Second-order arithmetic proves more than first-order arithmetic, but also more quickly:

**Theorem 3 ([Gödel, 1936, Buss, 1994]).**

> There exists a family $(P_j)_{j \in \mathbb{N}}$ such that
> - for all $j$, $A_1 \vdash P_j$
> - there exists $k$ such that for all $j$, $A_2 \vdash_k P_j$
> - there exists no $k$ such that for all $j$, $A_1 \vdash_k P_j$

True for all orders $i$ over $i - 1$
Higher-order logic in deduction modulo

[Dowek, Hardin and Kirchner, 2001] HOL$\lambda\sigma$ encodes the higher-order logic based on simple type theory.

Same proof length in HOL-$\lambda$ and in the sequent calculus modulo HOL$\lambda\sigma$.

Possibility to encode higher-order arithmetic without increasing proof length?
No restrictions on the rewrite system?

\( \mathcal{R} \): rewrite system such that \( P \leftrightarrow^* \top \) for all first-order tautology \( P \)

All proofs can be abridged to \( \top \)

Are those really proofs?

Proof checking is not decidable
A formal framework

If interested with links with complexity theory, proof checking must be performed in polynomial time [Cook and Reckhow, 1979]

In deduction modulo: the congruence should be decidable in polynomial time
Simple example

\[ \text{Add} \overset{\text{def}}{=} \begin{cases} \text{Add}(\text{O}, y, y) \rightarrow \top \\ \text{Add}(s(x), y, s(z)) \rightarrow \text{Add}(x, y, z) \end{cases} \]

**Proposition 4.**

- \( \text{Add} \vdash \text{Add}(i, i, 2i) \)
- \( \Theta \vdash \text{Add}(i, i, 2i) \) for all finite compatible presentations \( \Theta \)

\( \text{Add} \) is decidable in polynomial time
Higher-order arithmetic

Higher order: $\mathcal{HO}_i$

Remaining axioms: $fA$

= comprehension schema + rules encoding formulæ by terms

[Kirchner, 2006]

Theorem 5.

$$A_i \mid^k P \leadsto fA \mid^{O(k)}_{\mathcal{HO}_i} P$$
Proof length

"0th order"  1st order  ...  $i-1$st order  $i$th order

$A_{i-1} \vdash$

speed-up

$fA, \Theta_i \vdash$

shorter proofs

linear

$\vdash_{HHA} A_i^{mod}$

speed-up (Buss)

linear

$A_i \vdash$

linear

$\vdash_{HO} fA_i$
Contributions [CSL07]

- Simple speed-ups in deduction modulo
- Even when counting rewrite steps (using deep inference [Bruscoli and Guglielmi, 2008])
- Length-preserving simulation of higher-order arithmetic in first order modulo
- Purely computational presentation of higher-order arithmetic
Outline

1. Cut admissibility
2. Proof length

3. Logical framework
   - Motivations
   - Application to the functional pure type systems
Encoding higher-order systems in first order modulo?

- well studied
- existing efficient proof search procedures
- near to implementation
- universal (tool cooperation)
Logical framework

Motivations

Encoding higher-order systems in first order modulo?

- well studied
- existing efficient proof search procedures
- near to implementation
- universal (tool cooperation)
[Pfenning, 1996]:

“a meta-language for the specification of deductive systems”

most famous: ELF (based on $\lambda\Pi$)

HOL$\lambda\sigma = \text{specification of HOL-}\lambda$

Deduction modulo as a logical framework?
[Pfenning, 1996]:

“a meta-language for the specification of deductive systems”

most famous: ELF (based on $\lambda\Pi$)

$\text{HOL}\lambda\sigma = \text{specification of HOL-}\lambda$

Superdeduction as a logical framework?
Logical framework

[Pfenning, 1996]:

“a meta-language for the specification of deductive systems”

most famous: ELF (based on \( \lambda \Pi \))

\( \text{HOL}\lambda\sigma = \text{specification of HOL-\( \lambda \))

Superdeduction as a logical framework?
A methodology to specify a deductive system

\( \rightsquigarrow \) Application to functional pure type systems
Pure type systems [Geuvers and Nederhof, 1991]

A pure type system is given by

- sorts $S$
- axioms $A \subseteq S \times S$
- rules $R \subseteq S \times S \times S$

Functional if $A$ and $R$ are graphs defining functions
**Logical framework**

**Typing system**

Empty

\[ \varepsilon \text{ well-formed} \]

Declaration

\[
\frac{\Gamma \text{ well-formed}}{\Gamma, x : A \text{ well-formed}} \quad s \in S \text{ and } x \text{ not in } \Gamma
\]

Sort

\[
\frac{\Gamma \text{ well-formed}}{\Gamma \vdash s_1 : s_2 \quad (s_1, s_2) \in A}
\]

Variable

\[
\frac{\Gamma \text{ well-formed}}{\Gamma \vdash x : A \quad x : A \in \Gamma}
\]
Typing system (cont.)

Product
\[
\frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A, B : s_3} \quad (s_1, s_2, s_3) \in R
\]

Application
\[
\frac{\Gamma \vdash T : \Pi x : A, B \quad \Gamma \vdash U : A}{\Gamma \vdash (T \ U) : \{U/x\}B}
\]

Abstraction
\[
\frac{\Gamma \vdash \Pi x : A, B : s \quad \Gamma, x : A \vdash T : B}{\Gamma \vdash \lambda x : A, T : \Pi x : A, B}
\]

Conversion
\[
\frac{\Gamma \vdash T : A \quad \Gamma \vdash B : s}{\Gamma \vdash T : B} \quad s \in S \text{ and } A \not\rightarrow^*_\beta B
\]
Logical framework

Encoding the $\lambda$-terms

Binary predicate $\epsilon(t, u)$ to encode $T : U$ (shallow encoding)

$\lambda$-calculus with explicit substitutions [Kesner, 2000]

+ constants $\dot{s}$ for all sorts $s \in S$

+ binary function $\dot{\pi}_{\langle s_1, s_2, s_3 \rangle}$ for all rules $(s_1, s_2, s_3) \in R$

Additional term rewrite rules:

$\dot{s}[t] \rightarrow \dot{s}$

$\dot{\pi}_{\langle s_1, s_2, s_3 \rangle}(a, b)[s] \rightarrow \dot{\pi}_{\langle s_1, s_2, s_3 \rangle}(a[s], b[lift(s)])$
Encoding the inference rules through superrules

Find a rewrite rule of which one superrule correspond to the inference rule

\[
\begin{array}{c}
\text{Product} \quad \frac{\Gamma \vdash A : s_1 \quad \Gamma, x : A \vdash B : s_2}{\Gamma \vdash \Pi x : A, B : s_3} (s_1, s_2, s_3) \in R
\end{array}
\]

\[
\varepsilon (\pi_{s_1, s_2, s_3} (a, b), \dot{s}_3) \to \varepsilon (a, \dot{s}_1) \land \forall z. \varepsilon (z, a) \Rightarrow \varepsilon (b [\text{cons}(z)], \dot{s}_2)
\]

\[ (2) \]

\[
\begin{array}{c}
(2) \quad \frac{\Gamma \vdash \varepsilon (a, \dot{s}_1) \quad \Gamma, \varepsilon (z, a) \vdash \varepsilon (b [\text{cons}(z)], \dot{s}_2)}{\Gamma \vdash \varepsilon (\pi_{s_1, s_2, s_3} (a, b), \dot{s}_3) \quad z \notin \text{FV}(\Gamma, a, b)}
\end{array}
\]
**Logical framework**

**Correctness**

\[ \mathcal{PTS}_{(S,A,R)}: \text{explicit substitutions} + \]

\[
\begin{align*}
\epsilon (\hat{s}_1, \hat{s}_2) & \rightarrow \top & (s_1, s_2) \in A & (1) \\
\epsilon (\hat{\pi}_{(s_1,s_2,s_3)} (a, b), \hat{s}_3) & \rightarrow \epsilon (a, \hat{s}_1) \land \forall z. \epsilon (z, a) \Rightarrow \epsilon (b \left[ \text{cons}(z) \right], \hat{s}_2) & (2) \\
\epsilon (t, \hat{\pi}_{(s_1,s_2,s_3)} (a, b)) & \rightarrow \epsilon (\hat{\pi}_{(s_1,s_2,s_3)} (a, b), \hat{s}_3) \land \\
& \forall z. \epsilon (z, a) \Rightarrow \epsilon (t \, z, b \left[ \text{cons}(z) \right]) & (3)
\end{align*}
\]

**Theorem 6.**

If \( \Gamma \vdash_{(S,A,R)} T : A \) then \( \mid \Gamma \mid \vdash_{+\mathcal{PTS}_{(S,A,R)}} \epsilon \left( \mid T \mid_\Gamma, \mid A \mid_\Gamma \right) \)
Extra rules
Logical framework

Extra rules

\[(S, A, R)\]

Variable

Sort

Product

Abstraction

Application

Conversion

\[\text{SND}(\mathcal{PTS}(S, A, R))\]

ND

modulo \(\lambda_W\)

\[\Rightarrow\]

\[\land 1\]

\[\forall\]

\[\ldots\]

\[(1)\]

\[(2)\]

\[(3)\]

\[(2)_1\]

\[(2)_2\]

\[(3)_1\]

\[(3)_2\]
Logical framework

Extra rules

\[
(S, A, R) \quad \text{SND} \left( \mathcal{PTS}_{(S, A, R)} \right)
\]

Variable
Sort
Product
Abstraction
Application
Conversion

modulo $\lambda_W$
Logical framework

Application to the functional pure type systems

Conservativeness

Extra rules:

\[
\Gamma \vdash \epsilon \left( \tilde{\pi}_{\langle s_1, s_2, s_3 \rangle} (a, b), \dot{s}_3 \right)
\]

\[
\vdash \epsilon (a, \dot{s}_1)
\]

By correctness of the translation, if

\[|\Pi x : A, B| = \tilde{\pi}_{\langle s_1, s_2, s_3 \rangle} (a, b)\]

then

\[A : s_1\]

Theorem 7.

If \(\Gamma\) well formed and

\[|\Gamma| \vdash +_{\mathcal{PTS}(S, A, R)} \epsilon (a, b)\]

there exists \(A\) and \(B\) such that

\[a \xrightarrow{*} |A| \quad b \xrightarrow{*} |B| \quad \Gamma \vdash^{(S, A, R)} A : B\]
Contributions [LICS08]

▶ Methodology to encode deductive systems in superdeduction
▶ Correct and conservative encoding of functional pure type systems
▶ Proof search in PTS via the extensible sequent calculus
▶ New insight on normalization in PTS
Outline

0 Introduction

1 Cut admissibility

2 Proof length

3 Logical framework

4 Conclusion
  ■ Further work
Conclusion

Other simplicity criteria

- **Normalization**
  - new instance of abstract canonical system?
  - simplification rules?
    \[ A \rightarrow A \Rightarrow B ; \ B \rightarrow \top \ \Rightarrow A \rightarrow \top ; \ B \rightarrow \top \]

- **Decidability of the congruence**
  - decidable proof checking
  - decidability in polynomial time

- **Proof length**
  - A formal framework for proof complexity
  - Link deduction modulo – Tseytin’s extensions
Automating the logical framework

- From axiomatic presentations to rewrite systems:
  - automate
  - ensure the good properties
  - not always possible in intuitionistic logic

- Automated theorem proving
  - term rewrite rule strategies for the modulo
  - superrules application strategies
  - automated or user specified?

- A universal proof environment
  - share proof developments from different tools
  - modular deduction modulo
  - inductive types, subtyping, ...
Conclusion

Further work

Lemuridæ

Intuitionistic Super Sequent Calculus

Fellowship FO sequent calculus

Coq CC PVS λHOL

Europa λΠ-calculus modulo

Supernatural deduction

Lemuridæ Intuitionistic Super Sequent Calculus
Conclusion

Further work

Lemuridæ

[9x261] Intuitionistic Super Sequent Calculus

Fellowship FO sequent calculus

[9x261] [Sacerdoti Coen and Kirchner, 2006]

Coq CC PVS $\lambda$HOL

[9x261] [Cousineau and Dowek, 2007]

Europa $\lambda\Pi$-calculus modulo

[9x261] [Wack, 2006]

+ [Gentzen, 1934]

Supernatural deduction

Lemuridæ Intuitionistic Super Sequent Calculus

[9x261] [Cousineau and Dowek, 2007]

[9x261] [Wack, 2006]


Conclusion


Conclusion


Gentzen, G. (1934).
Conclusion

Untersuchungen über das logische Schliessen.
Translated in Szabo, editor., *The Collected Papers of Gerhard Gentzen* as “Investigations into Logical Deduction”.

Gödel, K. (1936).
Über die Länge von Beweisen.
*Ergebnisse eines Mathematischen Kolloquiums*, 7:23–24.
English translation in [Gödel, 1986].

On the length of proofs.

Conclusion

Confluence of extensional and non-extensional $\lambda$-calculi with explicit substitutions.


Kirchner, F. (2006).
A finite first-order theory of classes.

Simple word problems in universal algebras.

The practice of logical frameworks.
http://www.lix.polytechnique.fr/Labo/Florent.Kirchner/fellowship/.
