Deduction Modulo
Concepts, Mechanisation, Applications

Claude Kirchner
LORIA and INRIA
Nancy

Claude.Kirchner@loria.fr

joint works with *Gilles Dowek* and *Thérèse Hardin*.

Two main kinds of rules

Computation rules

$2+3$

fib(33)

sort(data-base)

Compute as fast as possible the unique result

Deduction rules

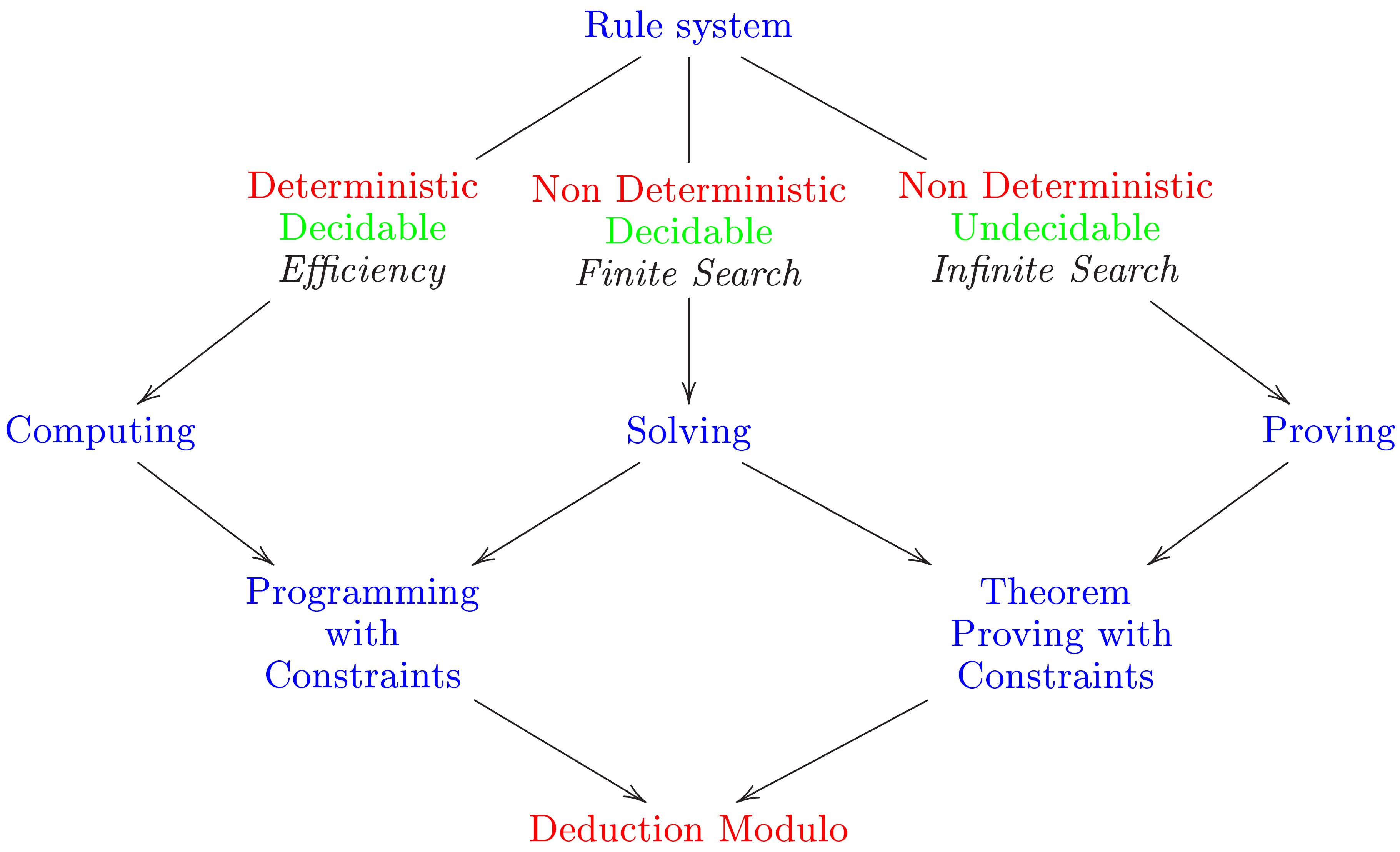
Sequent calculus

Resolution method

Solve($2x + 4y - 3z - u = 0$ in $\mathbf{N}$)

Search for a proof or a solution

From Computation versus Deduction ...



... To Reasoning versus Computing ...

Do we want to *prove* that $2{+}2 = 4$

or

Do we want to *check* it via a computation: $2 + 2 \to 4$?

... To Deduction modulo Computation

Computations could be considered as *much simpler* than *deduction*

$\rightarrow$ Poincaré principle

Indeed more abstractly we want to reason modulo a congruence that could be defined by computation rules, or by other means.

– For example 2+2 and 3+1 and 4 and 4+0 are all congruent modulo the definition of addition,

– $(x + y) + z, y + (x + z), (y + z) + x, \ldots$ are congruent modulo the associativity and commutativity of $+$.

What kind of congruences have been used up to now for reasoning modulo?

Congruence on terms

Andrews 71: equivalence modulo $\beta\eta$

operationally handled by $\beta\eta$-reduction

mechanization of deduction requires HO-unification

Plotkin 72: $\forall x\ \forall y\ \forall z\ ((x + y) + z = x + (y + z))$

operationally handled by rewriting

mechanization of deduction requires Associative-unification

Peterson& Stickel 81: E-completion

requires E-unification

Stickel 85: Theory resolution

requires solving specific constraints

Meseguer 89: logic = deduction rule + congruence on terms

We need more!

$$x * y = 0 \;\Leftrightarrow\; x = 0 \vee y = 0$$

$$x * y = x * z \;\Leftrightarrow\; y = z \vee x = 0$$

$$x \in \{y, z\} \Leftrightarrow (x = y) \vee (x = z)$$

$$x \in \mathcal{P}(y) \Leftrightarrow \forall z \; (z \in x \Rightarrow z \in y)$$

A congruence on propositions

It is useful to identify not only

Terms $x + y \equiv y + x$

but also

Propositions $x \in \mathcal{P}(y) \equiv \forall z\ (z \in x \Rightarrow z \in y)$

Reasoning Modulo

$$\frac{A \Rightarrow B \quad A}{B}$$

$$\frac{A' \Rightarrow B \quad A}{B} \quad \text{if } A \equiv A'$$

$$\frac{C \quad A}{B} \quad \text{if } C \equiv A \Rightarrow B$$

Main interests of this approach

1. Clear distinction between the decidable (computation) and undecidable (deduction),
2. Proofs becomes much simpler since the computation arguments are left out,
3. This provides a formal basis for the combination of theorem proving methods like model checking (which indeed is just computation) and resolution based ones (the deductive part),
4. This allows to be more efficient since the computational part could be implemented by specific and well tailored programs (e.g. AC-computation, arithmetic, . . .).
5. The approach is hierarchical:

Deduction Computation | Deduction Deduction Computation

Contents of this presentation

Deduction Modulo

- A way to present first-order logic: the sequent calculus modulo
- A proof search method and some applications: ENAR
- Completeness: the proof theoretic main arguments
- An application: HOL-$\lambda\sigma$

A sequent calculus modulo

The congruence $\equiv$ could be very conveniently defined by an equational rewrite system $=_{\mathcal{RE}}$

$$\frac{\Gamma, P \vdash_{\mathcal{RE}} \Delta \quad \Gamma \vdash_{\mathcal{RE}} Q, \Delta}{\Gamma \vdash_{\mathcal{RE}} \Delta} \text{ cut } \textsf{ if } P =_{\mathcal{RE}} Q$$

$$\frac{\Gamma, Q_1, Q_2 \vdash_{\mathcal{RE}} \Delta}{\Gamma, P \vdash_{\mathcal{RE}} \Delta} \text{ contr-l } \textsf{ if } P =_{\mathcal{RE}} Q_1 =_{\mathcal{RE}} Q_2$$

$$\frac{}{P \vdash_{\mathcal{RE}} Q}\text{axiom} \quad \text{if } P =_{\mathcal{RE}} Q$$

$$\frac{\Gamma, P \vdash_{\mathcal{RE}} \Delta \quad \Gamma \vdash_{\mathcal{RE}} Q, \Delta}{\Gamma \vdash_{\mathcal{RE}} \Delta}\text{cut} \quad \text{if } P =_{\mathcal{RE}} Q$$

$$\frac{\Gamma, Q_1, Q_2 \vdash_{\mathcal{RE}} \Delta}{\Gamma, P \vdash_{\mathcal{RE}} \Delta}\text{contr-l} \quad \text{if } P =_{\mathcal{RE}} Q_1 =_{\mathcal{RE}} Q_2$$

$$\frac{\Gamma \vdash_{\mathcal{RE}} Q_1, Q_2, \Delta}{\Gamma \vdash_{\mathcal{RE}} P, \Delta}\text{contr-r} \quad \text{if } P =_{\mathcal{RE}} Q_1 =_{\mathcal{RE}} Q_2$$

$$\frac{\Gamma \vdash_{\mathcal{RE}} \Delta}{\Gamma, P \vdash_{\mathcal{RE}} \Delta}\text{weak-l}$$

$$\frac{\Gamma \vdash_{\mathcal{RE}} \Delta}{\Gamma \vdash_{\mathcal{RE}} P, \Delta}\text{weak-r}$$

$$\frac{\Gamma, P, Q \vdash_{\mathcal{RE}} \Delta}{\Gamma, R \vdash_{\mathcal{RE}} \Delta}\wedge\text{-l} \quad \text{if } R =_{\mathcal{RE}} (P \wedge Q)$$

$$\frac{\Gamma \vdash_{\mathcal{RE}} P, \Delta \quad \Gamma \vdash_{\mathcal{RE}} Q, \Delta}{\Gamma \vdash_{\mathcal{RE}} R, \Delta}\wedge\text{-r} \quad \text{if } R =_{\mathcal{RE}} (P \wedge Q)$$

$$\frac{\Gamma, P \vdash_{\mathcal{RE}} \Delta \quad \Gamma, Q \vdash_{\mathcal{RE}} \Delta}{\Gamma, R \vdash_{\mathcal{RE}} \Delta}\vee\text{-l} \quad \text{if } R =_{\mathcal{RE}} (P \vee Q)$$

$$\frac{\Gamma \vdash_{\mathcal{RE}} P, Q, \Delta}{\Gamma \vdash_{\mathcal{RE}} R, \Delta}\vee\text{-r} \quad \text{if } R =_{\mathcal{RE}} (P \vee Q)$$

$$\frac{\Gamma \vdash_{\mathcal{RE}} P, \Delta \quad \Gamma, Q \vdash_{\mathcal{RE}} \Delta}{\Gamma, R \vdash_{\mathcal{RE}} \Delta}\Rightarrow\text{-l} \quad \text{if } R =_{\mathcal{RE}} (P \Rightarrow Q)$$

$$\frac{\Gamma, P \vdash_{\mathcal{RE}} Q, \Delta}{\Gamma \vdash_{\mathcal{RE}} R, \Delta}\Rightarrow\text{-r} \quad \text{if } R =_{\mathcal{RE}} (P \Rightarrow Q)$$

$$\frac{\Gamma \vdash_{\mathcal{RE}} P, \Delta}{\Gamma, R \vdash_{\mathcal{RE}} \Delta}\neg\text{-l} \quad \text{if } R =_{\mathcal{RE}} \neg P$$

$$\frac{\Gamma, P \vdash_{\mathcal{RE}} \Delta}{\Gamma \vdash_{\mathcal{RE}} R, \Delta}\neg\text{-r} \quad \text{if } R =_{\mathcal{RE}} \neg P$$

$$\frac{}{\Gamma, P \vdash_{\mathcal{RE}} \Delta}\bot\text{-l} \quad \text{if } P =_{\mathcal{RE}} \bot$$

$$\frac{\Gamma, Q\{x \leftarrow t\} \vdash_{\mathcal{RE}} \Delta}{\Gamma, P \vdash_{\mathcal{RE}} \Delta}(Q, t)\ \forall\text{-l} \quad \text{if } P =_{\mathcal{RE}} \forall x\ Q$$

$$\frac{\Gamma \vdash_{\mathcal{RE}} Q\{x \leftarrow y\}, \Delta}{\Gamma \vdash_{\mathcal{RE}} P, \Delta}(Q, y)\ \forall\text{-r} \quad \text{if } P =_{\mathcal{RE}} \forall x\ Q$$

$$\frac{\Gamma, Q\{x \leftarrow y\} \vdash_{\mathcal{RE}} \Delta}{\Gamma, P \vdash_{\mathcal{RE}} \Delta}(Q, y)\ \exists\text{-l} \quad \text{if } P =_{\mathcal{RE}} \exists x\ Q$$

$$\frac{\Gamma \vdash_{\mathcal{RE}} Q\{x \leftarrow t\}, \Delta}{\Gamma \vdash_{\mathcal{RE}} P, \Delta}(Q, t)\ \exists\text{-r} \quad \text{if } P =_{\mathcal{RE}} \exists x\ Q$$

$\mathcal{RE}$

– $\mathcal{R}$ a rewrite system that rewrites Atomic Proposition to Proposition

$$x * y = 0 \rightarrow_{\mathcal{R}} x = 0 \vee y = 0$$

– $\mathcal{E}$ a set of term rewrite rules and equational axioms equating:

- Atomic Proposition to Atomic Proposition

$$(x = y) =_{\mathcal{E}} (y = x)$$

- Term to Term

$$(x * (y * z)) =_{\mathcal{E}} ((x * y) * z)$$
$$(x * y) =_{\mathcal{E}} (y * x)$$

– $\mathcal{RE}$ a class rewrite system $\longrightarrow^{\mathcal{RE}}$:

- $t \longrightarrow^{\mathcal{RE}} t'$ when $t =_{\mathcal{E}} t_0 \rightarrow_{\mathcal{R}} t_1 =_{\mathcal{E}} t'$

For E = AC(+) and for $\mathcal{R}$ consisting in the rule $a + b \rightarrow c$

$$a + (b + c) \longrightarrow^{\mathcal{RE}} c + c$$

Compatibility

A set of axioms $\mathcal{T}$ and a class rewrite system $\mathcal{RE}$ are said to be compatible if:

- For all propositions P and Q, $P =_{\mathcal{RE}} Q$ implies $\mathcal{T} \vdash P \Leftrightarrow Q$.
- For all proposition P in $\mathcal{T}$, we have $\vdash_{\mathcal{RE}} P$.

Compatibility is natural

For a given $\mathcal{RE}$ we can always find a theory $\mathcal{T}_{\mathcal{RE}}$ such that $\mathcal{T}_{\mathcal{RE}}$ and $\mathcal{RE}$ are compatible:

- For $L \rightarrow R$ or $L = R$ in $\mathcal{RE}$ and L, R propositions:
  take $\mathcal{T}_{\mathcal{RE}} = \{\forall L \Leftrightarrow R\}$.
- For $l = r$ in $\mathcal{E}$
  either take $\mathcal{T}_{\mathcal{RE}} = \{\forall l = r\}$,
  but this requires to have an equality predicate and the corresponding axioms of equality,
  or take $\mathcal{T}_{\mathcal{RE}} = \{\forall P \Leftrightarrow Q \mid P =_{\mathcal{RE}} Q\}$

Relations between $\vdash$ and $\vdash_{\mathcal{RE}}$

When $\mathcal{RE}$ and $\mathcal{T}_{\mathcal{RE}}$ are compatible

$$\mathcal{T}_{\mathcal{RE}}, \Gamma \vdash \Delta$$

$$\Leftrightarrow$$

$$\mathcal{T}_{\mathcal{RE}}, \Gamma \vdash_{\mathcal{RE}} \Delta$$

$$\Leftrightarrow$$

$$\Gamma \vdash_{\mathcal{RE}} \Delta$$

Example: Integral rings

$$\forall x, y \quad x * y = 0 \Leftrightarrow x = 0 \vee y = 0$$

$$\mathcal{R} : x * y = 0 \to x = 0 \vee y = 0$$

How to prove: (A) $\quad \exists z (a * a = z \Rightarrow a = z)$?

$$\dfrac{\dfrac{\dfrac{a = 0 \vdash_{\mathcal{RE}} a = 0 \quad a = 0 \vdash_{\mathcal{RE}} a = 0}{\boxed{\begin{array}{c} a = 0 \vee a = 0 \vdash_{\mathcal{RE}} a = 0 \\ =_{\mathcal{RE}} \\ a * a = 0 \vdash_{\mathcal{RE}} a = 0 \end{array}}} \vee\text{-l}}{\vdash_{\mathcal{RE}} a * a = 0 \Rightarrow a = 0} \Rightarrow\text{-r}}{\vdash_{\mathcal{RE}} \exists z (a * a = z \Rightarrow a = z)} \exists\text{-r}$$

Example: Integral rings, continuing

How to prove by resolution: (A) $\exists z(a * a = z \Rightarrow a = z)$?

The clausal form of $\neg A$ is:

$a * a = z \qquad \neg(a = z)$

And there is NO resolution step possible, why? ...

Example: Integral rings, continuing

How to prove by resolution: (A) $\exists z(a * a = z \Rightarrow a = z)$?

The clausal form of $\neg A$ is:

$a * a = z \qquad \neg(a = z)$

And there is NO resolution step possible, why? ...

Because with the standard resolution rule, there is no way to use the now "built-in" information that $\forall x, y \quad x * y = 0 \Leftrightarrow x = 0 \vee y = 0$

Of course we can put everything in a big bag and apply resolution blindly, but the main idea of deduction modulo is different and we would like to use rewrite rules as such.

Theorem Proving Modulo

Mechanize the discovery of a proof of $A_1, \ldots, A_n \vdash B_1, \ldots, B_m$

ConvenientRepresentationOf$(A_1, \ldots, A_n \vdash B_1, \ldots, B_m)$

$\mapsto\!\!\!\!\twoheadrightarrow$

$\ldots$

$\mapsto\!\!\!\!\twoheadrightarrow$

Evidence of the proof

Extension of the resolution method

[Robinson 1965], [Stickel 1985], [Kirchner^{2}Rusinowitch 1990], ...

Clauses with constraints

Given an equational theory $\mathcal{E}$

$t =^?_{\mathcal{E}} t'$: equation modulo $\mathcal{E}$ i.e. pair of terms or atomic propositions
σ is a $\mathcal{E}$-solution of $t =^?_{\mathcal{E}} t'$ when $\sigma(t) =_{\mathcal{E}} \sigma(t')$

A constrained clause is a pair $U\,[\mathcal{C}]$ such that:

- U is a clause
- $\mathcal{C}$ is a set of equations.

It schematizes $\{\sigma(U) | \sigma \in \text{Solution}(\mathcal{C})\}$

▶ Allows for a clear separation between the "built-in" theory $\mathcal{E}$ and the rest of the deduction system,

▶ Permits to postpone the problem of constraint *solving*,

- undecidable unification problems in theories of main interest (HOL)
- Solving could be much more complex than just testing satisfiability

▶ Additional benefits: sharing, modularity, clever strategies, ...

For example

1. To make visible every symbolic computation step: unification, orientation, typing.
2. To schematize (infinitely) many objects.
   $(x * y)/y \to x \quad [y \neq 0]$
   $f(g(x)) \to g(x) \quad [x \in \{g^n(f(a)), n \geq 0\}]$
   $f(x) \quad [x + a =^?_A a + x]$
3. To take into account structure sharing.
   $f(x, x, x, x) \quad [x = \mathsf{bigterm}]$
4. To take advantage of constraint accumulation: $f(x, x, x, x) =^? f(u, v, w, z)$ has 34 359 607 481 minimal AC-solutions.

5. $$\begin{aligned}
f(x,x,x,x) &=^? f(u,v,w,z)\\
f(u,u,u,u) &=^? f(x,v,w,z)\\
f(v,v,v,v) &=^? f(u,x,w,z)\\
f(w,w,w,w) &=^? f(u,v,x,z)\\
f(z,z,z,z) &=^? f(u,v,w,x)\\
g(y,y,y,y) &=^? g(z,p,r,z)\\
g(z,z,z,z) &=^? g(y,p,r,s)\\
g(p,p,p,p) &=^? g(z,y,r,s)\\
g(r,r,r,r) &=^? g(z,p,y,s)\\
g(s,s,s,s) &=^? g(z,p,r,y)
\end{aligned}$$

has one minimal AC-solution.

Extended Narrowing and Resolution

Extended Resolution

$$\frac{\{P_1, \ldots, P_n, Q_1, \ldots, Q_m\}\,[\mathcal{C}_1] \quad \{\neg R_1, \ldots, \neg R_p, S_1, \ldots, S_q\}\,[\mathcal{C}_2]}{\{Q_1, \ldots, Q_m, S_1, \ldots, S_q\}\,[\mathcal{C}_1 \cup \mathcal{C}_2 \cup \{P_1 =^?_{\mathcal{E}} \ldots =^?_{\mathcal{E}} P_n =^?_{\mathcal{E}} R_1 \ldots =^?_{\mathcal{E}} R_p\}]}$$

Extended Narrowing

$$\frac{U\,[\mathcal{C}]}{U'\,[\mathcal{C} \cup \{U_{|\omega} =^?_{\mathcal{E}} l\}]} \text{ if } l \to r \in \mathcal{R},\ U_{|\omega} \text{ is not a variable and } U' \in c\ell(\{U[r]_\omega\})$$

Exemple: Back to integral rings

$\mathcal{R} = \{x * y = 0 \rightarrow x = 0 \vee y = 0\}$

How to prove (A): $\exists z(a * a = z \Rightarrow a = z)$?

The clausal form of $\neg A$ is: $a * a = z \qquad \neg(a = z)$

We can narrow $a * a = z$ using $\mathcal{R}$ and we get:

$$c\ell(x = 0 \vee y = 0\,[(a * a = z) =^? (x * y = 0)]) \qquad \neg(a = z)$$

$$x = 0,\ y = 0\,[(a * a = z) =^? (x * y = 0)] \qquad \neg(a = z)$$

Applying resolution we have furthermore:

$$y = 0\,[(x = 0) =^? (a = z) \wedge (a * a = z) =^? (x * y = 0)]$$

And once again:

$$\Box\,[(y = 0) =^? (a = z) \wedge (x = 0) =^? (a = z) \wedge (a * a = z) =^? (x * y = 0)]$$

And check satisfiability of the constraint:

$$y =^? a \ \wedge \ 0 =^? z \ \wedge \ x =^? a \ \wedge \ 0 =^? z \ \wedge \ a =^? x \ \wedge \ a =^? y \ \wedge \ z =^? 0$$

QED

Theorem Proving Modulo: Main Results

Let $\mathcal{RE}$ be a class rewrite system such that $\longrightarrow^{\mathcal{RE}}$ is confluent

Then, for all sentences $P_1, \ldots, P_n, Q_1, \ldots, Q_m$:

- If

$$c\ell(P_1, \ldots, P_n, \neg Q_1, \ldots, \neg Q_m)\,[\emptyset] \underset{\mathcal{RE}}{\mapsto\!\!\!\twoheadrightarrow} \square\,[\mathcal{C}]$$

  where $\mathcal{C}$ is a $\mathcal{E}$-unifiable set of constraints, then the sequent

$$P_1, \ldots, P_n \vdash_{\mathcal{RE}} Q_1, \ldots, Q_m$$

  is derivable.

- If the sequent $P_1, \ldots, P_n \vdash_{\mathcal{RE}} Q_1, \ldots, Q_m$ has a cut free proof then

$$c\ell(P_1, \ldots, P_n, \neg Q_1, \ldots, \neg Q_m)\,[\emptyset] \underset{\mathcal{RE}}{\mapsto\!\!\!\twoheadrightarrow} \square\,[\mathcal{C}]$$

  where $\mathcal{C}$ is a $\mathcal{E}$-unifiable set of constraints.

Corollary

When the cut rule is redundant in the sequent calculus modulo $\mathcal{RE}$ we have the following equivalence:

$$P_1, \ldots, P_n \vdash_{\mathcal{RE}} Q_1, \ldots, Q_m$$

$$\Leftrightarrow$$

$$c\ell(P_1, \ldots, P_n, \neg Q_1, \ldots, \neg Q_m)\,[\emptyset] \;\mapsto\!\!\!\twoheadrightarrow_{\mathcal{RE}}\; \Box\,[\mathcal{C}]$$

where $\mathcal{C}$ is a $\mathcal{E}$-unifiable set of constraints.

If $\mathcal{T}$ is a set of axioms compatible with $\mathcal{RE}$, this can be restated

$$\mathcal{T}, P_1, \ldots, P_n \vdash Q_1, \ldots, Q_m$$

$$\Leftrightarrow$$

$$c\ell(P_1, \ldots, P_n, \neg Q_1, \ldots, \neg Q_m)\,[\emptyset] \;\mapsto\!\!\!\twoheadrightarrow_{\mathcal{RE}}\; \Box\,[\mathcal{C}]$$

When is the cut rule redundant in the sequent calculus modulo $\mathcal{RE}$?

This has been studied in [Dowek & Werner, 98]

Sufficient conditions is to have $\mathcal{RE}$ confluent, terminating and either:

– without quantifier introduction

– without negative occurrence of literals in right-hand sides of rules

But also theories like first order presentation of HOL either based on combinatory logic or on theories of explicit substitutions.

Example: A simple theorem on sets

Prove: $A : \forall x (x \cap x = x)$

The clausal form of the negation of A is: $\neg a \cap a = a$

The following definitions of predicates:

$$\begin{array}{lcl} x = y & \Leftrightarrow & x \subseteq y \wedge y \subseteq x \\ x \subseteq y & \Leftrightarrow & \forall z (z \in x \Rightarrow z \in y) \\ z \in x \cap y & \Leftrightarrow & z \in x \wedge z \in y \end{array}$$

are used in an oriented way:

$$\begin{array}{lcl} x = y & \rightarrow & x \subseteq y \wedge y \subseteq x \\ x \subseteq y & \rightarrow & \forall z (z \in x \Rightarrow z \in y) \\ z \in x \cap y & \rightarrow & z \in x \wedge z \in y \end{array}$$

We are in a trivial case where the **Narrowing** rule is restricted to rewriting. Its repeated application yields the set of ground clauses:

$b \in a, c \in a$ $\qquad$ $\neg(b \in a), c \in a$ $\qquad$ $b \in a, \neg(c \in a)$ $\qquad$ $\neg(b \in a), \neg(c \in a)$

which could be trivially resolved into the empty clause.

Example: Sets again

Just adding the following definitions of union ($\cup$) and powerset ($\mathcal{P}$):

$$x \in \mathcal{P}(y) \;\rightarrow\; x \subseteq y$$

$$z \in x \cup y \;\rightarrow\; z \in x \vee z \in y$$

For the proof of: $\mathcal{P}(x \cap y) = \mathcal{P}(x) \cap \mathcal{P}(y)$

following [Plaisted and Zhu, 99], no proof were found by Otter after 1000s and 27656 generated clauses and it tooks 11s to their prover (OSHL) to prove it.

- Confirms that "removing" computational proofs makes a big difference.

Applications

To automated first-order theorem proving

To higher-order theorem proving: HOL-$\lambda\sigma$ [DHK-RTA99]

The "Deduction Modulo" paradigm is provided in the ELAN language

- Implementation of TPM
- Extensions (equational, ...)
- CoC modulo

Part II

Tools and proof schemas

Roadmap of the main theorem's proof

$$\mathcal{T}_{\mathcal{R}\mathcal{E}}, \Gamma \vdash \Delta \;\Leftrightarrow\; \Gamma \vdash_{\mathcal{R}\mathcal{E}} \Delta \;\Leftrightarrow\; c\ell(\Gamma, \neg\Delta) \hookrightarrow \Box \;\Leftrightarrow\; c\ell(\Gamma, \neg\Delta)[\![\emptyset]\!] \mapsto\!\!\!\twoheadrightarrow \Box[\![E]\!]$$

Extended Identical Resolution (EIR)

$$\frac{U}{U\{x \mapsto t\}}\textbf{Instantiation}$$

$$\frac{U}{U'}\textbf{Conversion} \quad \text{if } U =_{\mathcal{E}} U'$$

$$\frac{U}{U'}\textbf{Reduction} \quad \text{if } U \rightarrow_{\mathcal{R}} \psi \text{ and } U' \in c\ell(\psi)$$

$$\frac{U \cup \{P\} \quad U' \cup \{\neg P\}}{U \cup U'}\textbf{Identical Resolution}$$

EIR is sound

Let $P_1, \ldots, P_n, Q_1, \ldots, Q_m$ be sentences. If:

$$c\ell(P_1, \ldots, P_n, \neg Q_1, \ldots, \neg Q_m) \hookrightarrow \Box$$

then:

$$P_1, \ldots, P_n \vdash_{\mathcal{RE}} Q_1, \ldots, Q_m$$

Let $\{U_1, \ldots, U_p\} = c\ell(P_1, \ldots, P_n, \neg Q_1, \ldots, \neg Q_m)$

The proof works by induction on the structure of the derivation of $U_1, \ldots, U_p \hookrightarrow \Box$.

EIR Completeness

Let $P_1, \ldots, P_n, Q_1, \ldots, Q_m$ be sentences. If the sequent

$$P_1, \ldots, P_n \vdash_{\mathcal{RE}} Q_1, \ldots, Q_m$$

has a *cut free proof* then:

$$c\ell(P_1, \ldots, P_n, \neg Q_1, \ldots, \neg Q_m) \hookrightarrow \Box$$

By induction on the structure of a closed cut free proof of

$$P_1, \ldots, P_n \vdash_{\mathcal{RE}} Q_1, \ldots, Q_m$$

ENAR soundness

Let $P_1, \ldots, P_n, Q_1, \ldots, Q_m$ be sentences. If

$$c\ell(P_1, \ldots, P_n, \neg Q_1, \ldots, \neg Q_m)[\emptyset] \mapsto\!\!\!\twoheadrightarrow \Box[\mathcal{C}]$$

where $\mathcal{C}$ is a unifiable set of constraints, then

$$P_1, \ldots, P_n \vdash_{\mathcal{RE}} Q_1, \ldots, Q_m$$

Proof By induction on the structure of the derivation of

$$c\ell(P_1, \ldots, P_n, \neg Q_1, \ldots, \neg Q_m)[\emptyset] \mapsto\!\!\!\twoheadrightarrow \Box[\mathcal{C}]$$

we build a derivation of

$$c\ell(P_1, \ldots, P_n, \neg Q_1, \ldots, \neg Q_m) \hookrightarrow \Box$$

ENAR completeness

Consider

- a set $\mathcal{K}$ of constrained clauses,
- a set $\hat{\mathcal{K}}$ of constrained clauses that contains renamed copies of constrained clauses of $\mathcal{K}$,
- a substitution θ that is a $\mathcal{E}$-unifier of the constraints of all the clauses of $\hat{\mathcal{K}}$,
- a set $\mathcal{K}'$ of (non constrained) clauses such that $\theta\hat{\mathcal{K}} =_{\mathcal{E}} \mathcal{K}'$ and $\mathcal{K}' \hookrightarrow \Box$.

Then $\mathcal{K} \mapsto\!\!\!\twoheadrightarrow \Box[\mathcal{C}]$ where $\mathcal{C}$ is a $\mathcal{E}$-unifiable set of constraints.

We have $\mathcal{K}' \hookrightarrow \Box$ hence there exists a derivation $U_1, \ldots, U_n$ of the clause $\Box$ under the set $\mathcal{K}'$. We reason by induction on n.

About cut elimination

Cut elimination is a key point of the completeness proof which difficulty is not surprising:

For example, when considering the congruence defining higher-order logic, the completeness of ENAR implies the consistency of higher-order logic:

$\triangleright$ there is trivially no ENAR derivation to prove the contradiction $\perp$.

Related concepts and topics

- Equational completion: Peterson & Stickel (81), Jouannaud & H. Kirchner (85), Bachmair & Dershowitz (94), ...
- Theory resolution: Stickel (85)
- Theorem proving with constraints: Kirchner & Kirchner & Rusinowitch (90), Vigneron (92+), Nieuvenhuis & Rubio (92 +), Bachmair & Ganzinger & Lynch & Snyder (95 +), ...
- Narrowing (or Surreduction) modulo and with constraints: Hullot (80), Jouannaud & Kirchner & Kirchner (83), ...

Reference documents

Rewriting, Solving, Proving Claude & Hélène Kirchner

`http://www.loria.fr/~ckirchne/rsp.ps.gz`

Theorem Proving Modulo Gilles Dowek, Thérèse Hardin, Claude Kirchner

`http://www.loria.fr/~ckirchne/TPM.html`

PART III

HOL-$\lambda\sigma$

HOL-λ: An expression of Higher-Order Logic using λ-calculus

Two base types are considered ι and o

Terms are those of a simply typed λ-calculus build using the constants:

— $\dot{\Rightarrow}$, $\dot{\wedge}$ and $\dot{\vee}$, of type $o \rightarrow o \rightarrow o$,

— $\dot{\neg}$ of type $o \rightarrow o$,

— $\dot{\bot}$ of type o,

— $\dot{\forall}_T$ and $\dot{\exists}_T$ of type $(T \rightarrow o) \rightarrow o$

Propositions are terms of type o.

The $\beta\eta$-normal form of a term a is written $a\downarrow$.

The deduction rules of HOL-λ 1/2

$$\frac{}{P \vdash P}\text{axiom} \qquad \frac{\Gamma, P \vdash \Delta \quad \Gamma \vdash P, \Delta}{\Gamma \vdash \Delta}\text{cut}$$

$$\frac{\Gamma, P, P \vdash \Delta}{\Gamma, P \vdash \Delta}\text{contr-l} \qquad \frac{\Gamma \vdash P, P, \Delta}{\Gamma \vdash P, \Delta}\text{contr-r}$$

$$\frac{\Gamma \vdash \Delta}{\Gamma, P \vdash \Delta}\text{weak-l} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash P, \Delta}\text{weak-r}$$

$$\frac{\Gamma \vdash P, \Delta \quad \Gamma, Q \vdash \Delta}{\Gamma, (\dot{\Rightarrow}\ P\ Q) \vdash \Delta}\dot{\Rightarrow}\text{-l} \qquad \frac{P, \Gamma \vdash Q, \Delta}{\Gamma \vdash (\dot{\Rightarrow}\ P\ Q), \Delta}\dot{\Rightarrow}\text{-r}$$

The deduction rules of HOL-λ 2/2

$$\frac{\Gamma, P, Q \vdash \Delta}{\Gamma, (\dot{\wedge}\ P\ Q) \vdash \Delta}\,\dot{\wedge}\text{-l} \qquad \frac{\Gamma \vdash P, \Delta \quad \Gamma \vdash Q, \Delta}{\Gamma \vdash (\dot{\wedge}\ P\ Q), \Delta}\,\dot{\wedge}\text{-r}$$

$$\frac{\Gamma, P \vdash \Delta \quad \Gamma, Q \vdash \Delta}{\Gamma, (\dot{\vee}\ P\ Q) \vdash \Delta}\,\dot{\vee}\text{-l} \qquad \frac{\Gamma \vdash P, Q, \Delta}{\Gamma \vdash (\dot{\vee}\ P\ Q), \Delta}\,\dot{\vee}\text{-r}$$

$$\frac{\Gamma \vdash P, \Delta}{\Gamma, (\dot{\neg}\ P) \vdash \Delta}\,\dot{\neg}\text{-l} \qquad \frac{\Gamma, P \vdash \Delta}{\Gamma \vdash (\dot{\neg}\ P), \Delta}\,\dot{\neg}\text{-r}$$

$$\frac{}{\Gamma, \dot{\bot} \vdash \Delta}\,\dot{\bot}\text{-l}$$

$$\frac{\Gamma, (P\ t) \downarrow \vdash \Delta}{\Gamma, (\dot{\forall}\ P) \vdash \Delta}\,t\ \dot{\forall}\text{-l} \qquad \frac{\Gamma \vdash (P\ y) \downarrow, \Delta}{\Gamma \vdash (\dot{\forall}\ P), \Delta}\,\dot{\forall}\text{-r if } y \notin FV(\Gamma\Delta)$$

$$\frac{\Gamma, (P\ y) \downarrow \vdash \Delta}{\Gamma, (\dot{\exists}\ P) \vdash \Delta}\,\dot{\exists}\text{-l if } y \notin FV(\Gamma\Delta) \qquad \frac{\Gamma \vdash (P\ t) \downarrow, \Delta}{\Gamma \vdash (\dot{\exists}\ P), \Delta}\,t\ \dot{\exists}\text{-r}$$

Proof search method in HOL-λ

Higher-order constrained resolution. [Huet 1973]

Proof search method in HOL-λ

Higher-order constrained resolution. [Huet 1973]

? Could it be explained? as a first-order deduction procedure
in the same way as higher-order unification could be exactly expressed as equational unification in a theory of explicit substitution
[DHK, LICS95], [DHKP,ICLP96]

$$
\begin{tikzcd}
a =^?_{\beta\eta} b \arrow[d, "F"'] & & & R \\
a_F =^?_{\lambda\sigma} b_F \arrow[r, "\textbf{Unif}"] & P \arrow[rr, "\textbf{Back}"] & & Q \arrow[u, "F^{-1}"']
\end{tikzcd}
$$

Advantages of such a first-order approach to HOL

▶ Allows for a better abstract understanding of both frameworks

▶ Allows to represent HOL in first-order logical frameworks

▶ Permits to reuse the extensive works on first-order automated theorem proving (clever strategies, redundancy criteria, ...)

▶ Facilitates extension to e.g. equational higher-order theories

BUT

such translations (e.g. HOL-C, using combinators like S and K) are already known and are not really adequate since:

– In order to preserve provability the extentionality axioms should be used in HOL-C as they are not necessary in HOL-λ

– Moreover, the *structures* of proofs in the two systems are quite different.

HOWEVER

a good system exists: HOL-$\lambda\sigma$

It allows in particular an exact correspondance between the proof structures and more ...

HOL-$\lambda\sigma$

HOL-$\lambda\sigma$ is a many-sorted first-order theory with

sorts of the form

$$\Gamma \vdash T \text{ for term}$$
$$\Gamma \vdash \Delta \text{ for substitutions}$$

where Γ and Δ are sequences of simple types and T is a simple type.

formulas build using a single unary predicate symbol:

$$\varepsilon \quad \text{of rank} \quad (\vdash o)$$

terms build on the following function symbols:

$\mathtt{1}_A^\Gamma$	constant of sort	$A.\Gamma \vdash A$
$\alpha^\Gamma_{A\to B,A}$	binary function of rank	$(\Gamma \vdash A \to B, \Gamma \vdash A)\Gamma \vdash B$
$\lambda^\Gamma_{A,B}$	unary function of rank	$(A.\Gamma \vdash B)\Gamma \vdash A \to B$
$\_[\_]^{\Gamma,\Gamma'}_A$	binary function of rank	$(\Gamma' \vdash A, \Gamma \vdash \Gamma')\Gamma \vdash A$
id^Γ	constant of sort	$\Gamma \vdash \Gamma$
$\uparrow^\Gamma_A$	constant of sort	$A.\Gamma \vdash \Gamma$
$\cdot^{\Gamma,\Gamma'}_A$	binary function of rank	$(\Gamma \vdash A, \Gamma \vdash \Gamma')\Gamma \vdash A.\Gamma'$
$\circ^{\Gamma,\Gamma',\Gamma''}$	binary function of rank	$(\Gamma \vdash \Gamma'', \Gamma'' \vdash \Gamma')\Gamma \vdash \Gamma'$
$\dot{\Rightarrow}$	constant of sort	$\vdash o \to o \to o$
$\dot{\wedge}$	constant of sort	$\vdash o \to o \to o$
$\dot{\vee}$	constant of sort	$\vdash o \to o \to o$
$\dot{\neg}$	constant of sort	$\vdash o \to o$
$\dot{\bot}$	constant of sort	$\vdash o$
$\dot{\forall}_A$	constant of sort	$\vdash (A \to o) \to o$
$\dot{\exists}_A$	constant of sort	$\vdash (A \to o) \to o$

$\lambda\sigma$: A presentation of λ-calculus with explicit substitutions

β-reduction and η-reduction:

$$
\begin{array}{llll}
\textbf{Beta} & (\lambda a)b & \to & a[b.id] \\
\textbf{Eta} & \lambda(a\ \mathsf{1}) & \to & b \\
 & & & \text{if } a =_{\sigma} b[\uparrow]
\end{array}
$$

σ-reduction:

App	$(a\ b)[s]$	$\to$	$(a[s]\ b[s])$
VarCons	$\mathtt{1}[a.s]$	$\to$	a
Id	$a[id]$	$\to$	a
Abs	$(\lambda a)[s]$	$\to$	$\lambda(a[\mathtt{1}.(s \circ \uparrow)])$
Clos	$(a[s])[t]$	$\to$	$a[s \circ t]$
IdL	$id \circ s$	$\to$	s
ShiftCons	$\uparrow \circ\ (a.s)$	$\to$	s
AssEnv	$(s_1 \circ s_2) \circ s_3$	$\to$	$s_1 \circ (s_2 \circ s_3)$
MapEnv	$(a.s) \circ t$	$\to$	$a[t].(s \circ t)$
IdR	$s \circ id$	$\to$	s
VarShift	$\mathtt{1}.\uparrow$	$\to$	id
Scons	$\mathtt{1}[s].(\uparrow \circ\ s)$	$\to$	s

The $\mathcal{L}$-rewrite rules

$$\varepsilon(\dot{\Rightarrow}\ x\ y) \longrightarrow \varepsilon(x) \Rightarrow \varepsilon(y)$$

$$\varepsilon(\dot{\wedge}\ x\ y) \longrightarrow \varepsilon(x) \wedge \varepsilon(y)$$

$$\varepsilon(\dot{\vee}\ x\ y) \longrightarrow \varepsilon(x) \vee \varepsilon(y)$$

$$\varepsilon(\dot{\neg}\ x) \longrightarrow \neg\varepsilon(x)$$

$$\varepsilon(\dot{\bot}) \longrightarrow \bot$$

$$\varepsilon(\dot{\forall}_T\ x) \longrightarrow \forall y\ \varepsilon(x\ y)$$

$$\varepsilon(\dot{\exists}_T\ x) \longrightarrow \exists y\ \varepsilon(x\ y)$$

HOL-$\lambda\sigma$

We denote $\lambda\sigma\mathcal{L}$ the rewrite rules of $\lambda\sigma$-calculus together with the logical rules $\mathcal{L}$ given above.

HOL-$\lambda\sigma$ is the theory defined by the previous language and which deduction rules are the one of the sequent calculus modulo $\lambda\sigma\mathcal{L}$.

Theorems:

— Termination: The system $\lambda\sigma\mathcal{L}$ is weakly terminating.

— Confluence: The system $\lambda\sigma\mathcal{L}$ is confluent on terms containing only term variables.

— Consistency: The theory HOL-$\lambda\sigma$ is consistent.

— Proof normalization: Proofs modulo $\lambda\sigma\mathcal{L}$ normalize.

Embedding HOL-λ in HOL-$\lambda\sigma$

Pre-Cooking

Let a be a λ-term. The pre-cooking of a is the $\lambda\sigma$-term defined by $a_F = F(a, [\,])$ where $F(a, l)$ is defined using the list of variable l ($[\,]$ being the empty list) by:

- $F((\lambda x.a), l) = \lambda(F(a, x.l))$,
- $F((a\ b), l) = F(a, l)F(b, l)$,
- $F(x, l) = 1[\uparrow^{k-1}]$, if x is the k-th variable of l
- $F(x, l) = x[\uparrow^{n}]$ where n is the length of l if x is a variable not occurring in l or a constant.

Example

With $a : A, X : A \to A$,

$$\lambda_A y.(X\ a) =^?_{\beta\eta} \lambda_A y.a$$

This is encoded in de Bruijn terms, using the context $\Gamma = A.nil$

$$\lambda(X\ 2) =^?_{\beta\eta} \lambda 2$$

and then pre-cooked into:

$$\lambda(X[\uparrow]\ 2) =^?_{\lambda\sigma} \lambda 2.$$

We get it right!

1– Main Result:

If $p_1, \ldots, p_n, q$ are propositions in HOL-λ then

$$p_1, \ldots, p_n \vdash q \text{ is provable in HOL-}\lambda$$
$$\text{if and only if}$$
$$\varepsilon(p_{1F}), \ldots, \varepsilon(p_{nF}) \vdash_{\mathcal{RE}} \varepsilon(q_F) \text{ is provable in HOL-}\lambda\sigma.$$

Moreover the proof structure is preserved.

2– We can use ENAR as a proof search method in HOL-$\lambda\sigma$ and it mimics higher-order resolution in a full first-order framework.

Example 1/2

In HOL-$\lambda\sigma$, $\exists x\ \varepsilon(x)$ is provable:

$$
\dfrac{\dfrac{\dfrac{\varepsilon(y) \vdash \varepsilon(y)}{\vdash \varepsilon(y), \neg\varepsilon(y)}\ \neg\text{-r}}{\boxed{\begin{array}{c}\vdash \varepsilon(y) \vee \neg\varepsilon(y) \\ =_{\mathcal{RE}} \\ \vdash \varepsilon(\alpha(\alpha(\dot{\vee}, y), \alpha(\dot{\neg}, y)))\end{array}}}\ \vee\text{-r}}{\vdash \exists x\ \varepsilon(x)}\ \exists\text{-r}
$$

Example 2/2

$$
\begin{array}{ll}
\neg\varepsilon(X)\,[\emptyset] & \rightarrow \textbf{Extended Narrowing} \\
& c\ell(\neg\neg\varepsilon(Y))\,[\varepsilon(X) =^{?} \varepsilon(\alpha(\dot{\neg}, Y))] \\
& = \ \varepsilon(Y)\,[\varepsilon(X) =^{?} \varepsilon(\alpha(\dot{\neg}, Y))] \\
& \rightarrow \textbf{Extended Resolution} \\
& \Box\,[\varepsilon(X) =^{?} \varepsilon(\alpha(\dot{\neg}, Y)), \varepsilon(X') =^{?} \varepsilon(Y)]
\end{array}
$$

which constraint is trivially satisfiable.

And another proof could be

$$\neg\varepsilon(x)\,[\emptyset] \quad \to_{\textbf{Narrowing}} \qquad (\text{using } \varepsilon(\dot{\vee}\ y\ z) \to \varepsilon(y) \vee \varepsilon(z))$$

$$c\ell(\neg(\varepsilon(y) \vee \varepsilon(z)))\,[\varepsilon(x) =^? \varepsilon(\alpha(\alpha(\dot{\vee}, y), z))]$$

$$= \neg\varepsilon(y)\,[\varepsilon(x) =^? \varepsilon(\alpha(\alpha(\dot{\vee}, y), z))] \quad \neg\varepsilon(z)\,[\varepsilon(x) =^? \varepsilon(\alpha(\alpha(\dot{\vee}, y), z))]$$

$$\to_{\textbf{Narrowing}} \qquad (\text{using } \varepsilon(\dot{\neg}\ u) \to \neg\varepsilon(u))$$

$$c\ell(\neg\neg\varepsilon(u))\,[\varepsilon(x) =^? \varepsilon(\alpha(\alpha(\dot{\vee}, y), z)) \wedge \varepsilon(\alpha(\dot{\neg}, u)) =^? \varepsilon(y)] \quad \neg\varepsilon(z)\,[\ldots]$$

$$= \varepsilon(u)\,[\varepsilon(x) =^? \varepsilon(\alpha(\alpha(\dot{\vee}, y), z)) \wedge \varepsilon(\alpha(\dot{\neg}, u)) =^? \varepsilon(y)] \quad \neg\varepsilon(z)\,[\ldots]$$

$$\to \textbf{Resolution}$$

$$\Box\,[\varepsilon(x) =^? \varepsilon(\alpha(\alpha(\dot{\vee}, y), z)) \wedge \varepsilon(\alpha(\dot{\neg}, u)) =^? \varepsilon(y) \wedge u =^? z]$$

and since the constraint is trivially satisfiable, this terminates the proof.

An unexpected consequence of encoding HOL in HOL-$\lambda\sigma$

Skolemization in HOL is know to be complicated [Miller83]

The naive first-order skolemization applied on the translation from HOL to HOL-$\lambda\sigma$ is correct and give a new understanding of Miller's conditions.

TPM: to sum up

- Success in the use of HOL-$\lambda\sigma$ for encoding HOL.
- Higher-order versus first-order resolution.
- Implementations.
- Extensions (equational, ...).
- CoC modulo.

References

G. Dowek, Th. Hardin, and C. Kirchner.
Theorem proving modulo
Rapport de Recherche 3400, INRIA, April 1998.
Soon in JAR

G. Dowek, Th. Hardin, and C. Kirchner.
HOL-$\lambda\sigma$ an intentional first-order expression of higher-order logic.
Rapport de Recherche 3556, INRIA, November 1998.
and
RTA 99, soon in MSCS

G. Dowek and B. Werner.
Proof normalization modulo.
Rapport de Recherche 3542, INRIA, November 1998.
Types for Proofs and Programs, LNCS 1657

All on: `http://www.inria.fr/`