Expressing theories in the $\lambda\Pi$-calculus modulo theory and in the Dedukti system

Gilles Dowek

With Ali Assaf, Guillaume Burel, Raphaël Cauderlier, David Delahaye, Catherine Dubois, Frédéric Gilbert, Pierre Halmagrand, Olivier Hermant, and Ronan Saillard

Predicate logic

(Peano) arithmetic, (Euclide) geometry, (Zermelo) set theory...
Theories in Predicate logic (Hilbert and Ackermann, 1928)

A logical framework where formalisms can be defined as theories

- $\wedge$, $\vee$, $\forall$... defined once for all
- proof, model... defined once for all
- soundness, completeness... proved once for all
- $Z \subseteq ZF \subseteq ZFC$
- if $\mathcal{T} \vdash A \Rightarrow B$ and $\mathcal{T}' \vdash A$, then $\mathcal{T} \cup \mathcal{T}' \vdash B$

But...

The Theory of classes (aka Second-order logic)
Simple type theory (aka Higher-order logic)
The Calculus of constructions
The Calculus of inductive constructions

... not theories expressed in Predicate logic

A Babel tower

Before: a proof of xyz (rarely: using the axiom of choice)
Now: a Coq proof of the four color theorem", "an Isabelle/HOL proof of the correctness of seL4"

A proof of A in S cannot be used in S'

A proof of A in S, a proof of $A \Rightarrow B$ in S', a proof of B in nothing

Five limitations of Predicate logic

1. No bound variables (except $\forall$, $\exists$), no function symbol $\mapsto$
2. No proofs-as-terms principle
3. No computation: a proof of $2 + 2 = 4$
4. No theory-independent cut-elimination theorem
5. No constructive proofs

Partial solutions: more logical frameworks

1. λ-Prolog, Isabelle

1, 2. LF, aka $\lambda\Pi$-calculus, aka λ-calculus with dependent types

3, 4. Deduction modulo theory

Combine $\lambda\Pi$-calculus and Deduction modulo theory: $\lambda\Pi$-calculus modulo theory (variant of the Martin-Löf logical framework)
Solves 1., 2., 3., 4., and 5.

Implemented in DEDUKTI `http://dedukti.gforge.inria.fr/`

Simple type theory in DEDUKTI: 8 variables and 3 rules

$$type : Type$$

$$o : type$$

$$\iota : type$$

$$arrow : type \to type \to type$$

$$\eta : type \to Type$$

$$\eta(arrow\ a\ b) \longrightarrow \eta(a) \to \eta(b)$$

$$\Rightarrow : \eta(o) \to \eta(o) \to \eta(o)$$

$$\forall : \Pi a : type\ ((\eta(a) \to \eta(o)) \to \eta(o))$$

$$\varepsilon : \eta(o) \to Type$$

$$\varepsilon(\Rightarrow\ p\ q) \longrightarrow \varepsilon(p) \to \varepsilon(q)$$

$$\varepsilon(\forall\ a\ p) \longrightarrow \Pi x : \eta(a)\ \varepsilon(p\ x)$$

What does "expressing a logic in a framework" means?

Adequacy theorem (in principle)
Large library of formal proofs translated and checked (in facts)

Dedukti libraries (650 MB)

- Constructive predicate logic (Resolution proofs): The iProverModulo TPTP library (38.1 MB)
- Classical logic (tableaux proofs): The Zenon modulo Set Theory Library (595 MB)
- FoCaLiZe: The Focalide library (1.89 MB)
- Simple type theory: The Holide library (21.5 MB)
- The Calculus of constructions with universes: The Matita arithmetic library (1.11 MB)

Minimal logic in the $\lambda\Pi$-calculus

ι : *Type*
for each variable x, $\mathtt{x} : \iota$
for each function symbol f, $\mathtt{f} : \iota \to ... \to \iota \to \iota$
for each predicate symbol P, $\mathtt{P} : \iota \to ... \to \iota \to$ *Type*

- $|x| = \mathtt{x}$
- $|f(t_1, ..., t_n)| = (\mathtt{f}\ |t_1|\ ...\ |t_n|)$
- $|P(t_1, ..., t_n)| = (\mathtt{P}\ |t_1|\ ...\ |t_n|)$
- $|A \Rightarrow B| = |A| \to |B|$, i.e. $\Pi z : |A|\ |B|$
- $|\forall x\ A| = \Pi \mathtt{x} : \iota\ |A|$

A provable if and only if there exists π such that $\pi : |A|$

o aka *Prop*, *bool*...

$\iota : \mathit{Type}$, $o : \mathit{Type}$
for each predicate symbol P, $\mathtt{P} : \iota \to ... \to \iota \to o$

$\top$, $\bot$ of type o
$\Rightarrow$, $\wedge$, $\vee$ of type $o \to o \to o$
$\forall$, $\exists$ of type $(\iota \to o) \to o$
o embedded in *Type* with ε of type $o \to \mathit{Type}$

Meaning defined by rewrite rules e.g.

$$\varepsilon(\wedge\ x\ y) \longrightarrow \Pi z : o\ ((\varepsilon(x) \to \varepsilon(y) \to \varepsilon(z)) \to \varepsilon(z))$$

The ~~im~~predicative expression of connectives and quantifiers

$$\varepsilon(\wedge\ x\ y) \longrightarrow \Pi z : o\ ((\varepsilon(x) \to \varepsilon(y) \to \varepsilon(z)) \to \varepsilon(z))$$

$\Pi z : o$: a quantification over all propositions
But... yields a type (: $Type$) and not a proposition (: o)

Not even in the image of the embedding ε

Propositions-as-types: $o \sqsubseteq Type$ (ε) not $o = Type$

Ongoing work

More proofs: PVS (predicate subtyping), Coq (universe polymorphism: rewriting modulo AC), SMT-solvers

Reverse engineering of proofs: Half of the HOL-Light standard library is constructive *a posteriori*

Can we express (part of) the Matita arithmetic library in HA?