Logical frameworks, reverse mathematics, and formal proofs translation

Gilles Dowek
The question

Formal proofs developed in different systems: Coq, Matita, HOL Light... expressed in different theories

Can a proof expressed in a theory $\mathcal{T}$ be translated to $\mathcal{T}'$?

Can a proof in ZF be translated to ZFC? a proof in ZFC to ZF? a proof in Euclidean geometry to Hyperbolic geometry? a proof in Hyperbolic geometry to Euclidean geometry?
Proof transformation

There exists a basis of $\mathbb{R}^2$

- by the incomplete basis theorem (axiom of choice)
- $\langle 1, 0 \rangle, \langle 0, 1 \rangle$

automatically (for example: constructivization) or by hand
Reformulating the project of reverse mathematics

- **Formal** proofs, not pencil-paper-L\(\text{ATEX}\)ones
- **Expressive** theories (Set theory, Type theory...) and not fragments of arithmetic
- **Analyze** proofs before (possibly) transforming them
How is this possible?

Because ZF and ZFC, Euclidean geometry, Hyperbolic geometry... expressed in the same **logical framework**: predicate logic

In predicate logic, a theory: **several** axioms

Permits to raise the question: **which** axioms are used in a proof

Not if ZF and ZFC were defined by Turing machines terminating on their theorems
The revolution of predicate logic

Since Euclid: geometry, arithmetic, set theory... each system its syntax, its notion of proof...

Hilbert and Ackermann (1928): predicate logic
A common framework for geometry, arithmetic, set theory...
A theory: symbols and axioms

But a short revolution: not Russell’s Type theory, not Church’s Type theory, not Martin-Löf’s Type theory, not the Calculus of constructions...
Three attitudes

- Consider logical framework as a **dead** concept
- Express Russell’s type theory, Church’s, Martin-Löf’s, the Calculus of constructions... in predicate logic **by will of by force** (Henkin, Davis, D...)
- Extend predicate logic to a **better** logical framework
The limits of predicate logic

- No bound variables ($\lambda x \ x$)
- No syntax for proofs
- No notion of computation
- No good notion of cut
- Classical and not constructive

$\lambda\Pi$-calculus modulo theory ($\text{Dedukti}$) generalizes them all
New logical frameworks

- No bound variables ($\lambda x \ x$): $\lambda$-Prolog, Isabelle, $\lambda\Pi$-calculus
- No syntax for proofs: $\lambda\Pi$-calculus
- No notion of computation: Deduction modulo theory
- No good notion of cut: Deduction modulo theory
- Classical and not constructive: Ecumenical logic

$\lambda\Pi$-calculus modulo theory (DEDUKTI) generalizes them all
Simple type theory in \textbf{Dedukti}

\[
\begin{align*}
type & : \ Type \\
\eta & : \ type \to \ Type \\
o & : \ type \\
nat & : \ type \\
arrow & : \ type \to \ Type \\
\varepsilon & : \ (\eta \ o) \to \ Type \\
\Rightarrow & : \ (\eta \ o) \to (\eta \ o) \to (\eta \ o) \\
\forall & : \ \Pi x : \ type \ (((\eta \ x) \to (\eta \ o)) \to (\eta \ o)) \\
\end{align*}
\]

\[
\begin{align*}
(\eta \ (arrow \ x \ y)) & \to (\eta \ x) \to (\eta \ y) \\
(\varepsilon \ (\Rightarrow \ x \ y)) & \to (\varepsilon \ x) \to (\varepsilon \ y) \\
(\varepsilon \ (\forall \ x \ y)) & \to \ \Pi \ z : \ (\eta \ x) \ ((\varepsilon \ (y \ z)))
\end{align*}
\]
The Calculus of constructions in Dedukti

\[
\begin{align*}
\text{type} & : \quad \text{Type} \\
\eta & : \quad \text{type} \rightarrow \text{Type} \\
o & : \quad \text{type} \\
nat & : \quad \text{type} \\
\text{arrow} & : \quad \Pi x : \text{type} (((\eta x) \rightarrow \text{type}) \rightarrow \text{type}) \\
\varepsilon & : \quad (\eta o) \rightarrow \text{Type} \\
\Rightarrow & : \quad \Pi x : (\eta o) (((\varepsilon x) \rightarrow (\eta o)) \rightarrow (\eta o)) \\
\forall & : \quad \Pi x : \text{type} (((\eta x) \rightarrow (\eta o)) \rightarrow (\eta o)) \\
\pi & : \quad \Pi x : (\eta o) (((\varepsilon x) \rightarrow \text{type}) \rightarrow \text{type})
\end{align*}
\]

\[
\begin{align*}
(\eta (\text{arrow} x y)) & \rightarrow \Pi z : (\eta x) (\eta (y z)) \\
(\varepsilon (\Rightarrow x y)) & \rightarrow \Pi z : (\varepsilon x) (\varepsilon (y z)) \\
(\varepsilon (\forall x y)) & \rightarrow \Pi z : (\eta x) (\varepsilon (y z)) \\
(\eta (\pi x y)) & \rightarrow \Pi z : (\varepsilon x) (\eta (y z))
\end{align*}
\]
A comparison

- *arrow dependent* in the Calculus of constructions but not in Simple type theory
- Same for ⇒
- An extra symbol $\pi$ in the Calculus of constructions: express functions mapping proofs to terms
Reverse mathematics in DEDUKTI

- All proofs in Simple type theory can be translated to the Calculus of constructions
- The proofs in the Calculus of constructions that do not use these three features can be translated to Simple type theory (not the others: genuine Calculus of constructions proofs)

For example: all the proofs of the arithmetic library of MATITA

“First” proof of Fermat’s little theorem in constructive Simple type theory (further: predicative, PA, fragments of PA...
Reverse mathematics as the basis of interoperability
Theorem

fermat.congruent_exp_pred_SO

Statement

∀ p, a, prime p → ¬(p | a) → (a ^ (p-1)) ≡ 1 [p]

Main Dependencies


Theory


Why does it work?

Because proof systems implement very expressive theories and use only a tiny part of it.

Early empirical evidences

- Proof systems: very different theories, but very similar libraries.
- Mathematicians are not very interested in the axioms used in their proofs: any theory seems to fit.