Logical frameworks, reverse mathematics, and formal proof translation

Gilles Dowek

Is mathematical truth absolute?

Yes, then no, then yes, then no, then yes, then no again

- Yes
- No: "The sum of the angles of a triangle is π" is true in some geometries, not in others
- Yes: The judgement must include the theory: from $\vdash A$ to $\mathcal{T} \vdash A$, whose truth is absolute
- No: Bolzano-Weierstrass theorem true in some logic but not in others
- Yes: Deduction rules express the meaning of connectives and quantifiers, so $\exists$ and $\exists_c$ are different quantifiers
- No: Hales theorem has a HOL LIGHT proof while the four color theorem has a COQ proof

Problems: interoperability, sustainability

Is there a proof of (for instance) Hales theorem in Coq?

Can (for instance) the HOL Light proof be translated to Coq?

Not a new question

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

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

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: elimination of the axiom of choice, constructivization) or by hand

Reverse mathematics is the basis of interoperability but...

Reformulating the project of reverse mathematics

- Formal proofs, not pencil-paper-LaTeXones
- Expressive theories (Set theory, Type theory...) and not fragments of arithmetic
- Analyze proofs before (possibly) transforming them

Logical Frameworks

The interoperability ZF / ZFC possible because ZF and ZFC 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 π?

Would not be possible 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): a common predicate logic

A common framework for geometry, arithmetic, set theory...
Sharing connectives, deduction rules...

A theory: symbols and axioms

But a short revolution

Already in 1928: no reformulation of Type theory (*Principia Mathematica*) in predicate logic

Then (1940) Church: a new formulation of Type theory (based on λ-calculus) impossible to express in predicate logic (λ binds)

1970, 1985... Martin-Löf's type theory, the Calculus of constructions... not in predicate logic

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

New logical frameworks

- No bound variables ($\lambda x\ x$): λ-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

Defining a theory in DEDUKTI

No universal method
But several paradigmatic examples

- Any (finite) theory expressed in Predicate logic
- Axiom schemes
- Simple type theory (without and with polymorphism)
- Pure type systems (CoC...)
- Inductive types
- Universes

Ongoing: universe polymorphism, proof irrelevance, predicate subtyping

Simple type theory in Dedukti

$$
\begin{array}{rcl}
\mathit{type} & : & \mathit{Type} \\
\eta & : & \mathit{type} \to \mathit{Type} \\
o & : & \mathit{type} \\
\mathit{nat} & : & \mathit{type} \\
\mathit{arrow} & : & \mathit{type} \to \mathit{type} \to \mathit{type} \\
\varepsilon & : & (\eta\ o) \to \mathit{Type} \\
\Rightarrow & : & (\eta\ o) \to (\eta\ o) \to (\eta\ o) \\
\forall & : & \Pi x : \mathit{type}\ (((\eta\ x) \to (\eta\ o)) \to (\eta\ o))
\end{array}
$$

$$
\begin{array}{rcl}
(\eta\ (\mathit{arrow}\ x\ y)) & \longrightarrow & (\eta\ x) \to (\eta\ y) \\
(\varepsilon\ (\Rightarrow\ x\ y)) & \longrightarrow & (\varepsilon\ x) \to (\varepsilon\ y) \\
(\varepsilon\ (\forall\ x\ y)) & \longrightarrow & \Pi z : (\eta\ x)\ (\varepsilon\ (y\ z))
\end{array}
$$

Examples

Types: $nat \to nat$ expressed as $(arrow\ nat\ nat)$ of type $type$
Then to $(\eta\ (arrow\ nat\ nat))$ of type $Type$ that reduces to
$(\eta\ nat) \to (\eta\ nat)$

Terms: $\lambda x : nat\ x$ expressed as $\lambda x : (\eta\ nat)\ x$ of type
$(\eta\ nat) \to (\eta\ nat)$

Propositions: $\forall_o\ \lambda X : o\ (X \Rightarrow X)$ expressed as
$\forall\ o\ \lambda X : (\eta\ o)\ (\Rightarrow\ X\ X)$ of type $(\eta\ o)$
Then to $(\varepsilon\ (\forall\ o\ \lambda X : (\eta\ o)\ (\Rightarrow\ X\ X)))$ of type $Type$ that reduces to $\Pi X : (\eta\ o)\ ((\varepsilon\ X) \to (\varepsilon\ X))$

Proofs: well-know expressed as $\lambda X : (\eta\ o)\ \lambda \alpha : (\varepsilon\ X)\ \alpha$ of type
$\Pi X : (\eta\ o)\ ((\varepsilon\ X) \to (\varepsilon\ X))$

Simple type theory in Dedukti

$$
\begin{array}{rcl}
\mathit{type} & : & \mathit{Type} \\
\eta & : & \mathit{type} \to \mathit{Type} \\
o & : & \mathit{type} \\
\mathit{nat} & : & \mathit{type} \\
\mathit{arrow} & : & \mathit{type} \to \mathit{type} \to \mathit{type} \\
\varepsilon & : & (\eta\ o) \to \mathit{Type} \\
\Rightarrow & : & (\eta\ o) \to (\eta\ o) \to (\eta\ o) \\
\forall & : & \Pi x : \mathit{type}\ (((\eta\ x) \to (\eta\ o)) \to (\eta\ o))
\end{array}
$$

$$
\begin{array}{rcl}
(\eta\ (\mathit{arrow}\ x\ y)) & \longrightarrow & (\eta\ x) \to (\eta\ y) \\
(\varepsilon\ (\Rightarrow\ x\ y)) & \longrightarrow & (\varepsilon\ x) \to (\varepsilon\ y) \\
(\varepsilon\ (\forall\ x\ y)) & \longrightarrow & \Pi z : (\eta\ x)\ (\varepsilon\ (y\ z))
\end{array}
$$

The Calculus of constructions in DEDUKTI

$$\begin{array}{rcl}
type & : & Type \\
\eta & : & type \rightarrow Type \\
o & : & type \\
nat & : & type \\
arrow & : & \Pi x : type \; (((\eta \; x) \rightarrow type) \rightarrow type) \\
\varepsilon & : & (\eta \; o) \rightarrow Type \\
\Rightarrow & : & \Pi x : (\eta \; o) \; (((\varepsilon \; x) \rightarrow (\eta \; o)) \rightarrow (\eta \; o)) \\
\forall & : & \Pi x : type \; (((\eta \; x) \rightarrow (\eta \; o)) \rightarrow (\eta \; o)) \\
\pi & : & \Pi x : (\eta \; o) \; (((\varepsilon \; x) \rightarrow type) \rightarrow type)
\end{array}$$

$$\begin{array}{rcl}
(\eta \; (arrow \; x \; y)) & \longrightarrow & \Pi z : (\eta \; x) \; (\eta \; (y \; z)) \\
(\varepsilon \; (\Rightarrow \; x \; y)) & \longrightarrow & \Pi z : (\varepsilon \; x) \; (\varepsilon \; (y \; z)) \\
(\varepsilon \; (\forall \; x \; y)) & \longrightarrow & \Pi z : (\eta \; x) \; (\varepsilon \; (y \; z)) \\
(\eta \; (\pi \; x \; y)) & \longrightarrow & \Pi z : (\varepsilon \; x) \; (\eta \; (y \; z))
\end{array}$$

A comparison

- *arrow* dependent in the Calculus of constructions but not in Simple type theory
- Same for $\Rightarrow$
- An extra symbol π 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)

An even more general theory

- Take both *arrow* and arrow_d (and $\Rightarrow$ and $\Rightarrow_d$): 11 symbols, 6 rules
- Express the proof of the Calculus of constructions with arrow_d and $\Rightarrow_d$
- Replace arrow_d with *arrow* when dependency is not used (resp. $\Rightarrow_d$ with $\Rightarrow$)
- A proof can be expressed in Simple type theory if it uses *arrow*, $\Rightarrow$ and $\forall$ (but not arrow_d, $\Rightarrow_d$ and π)

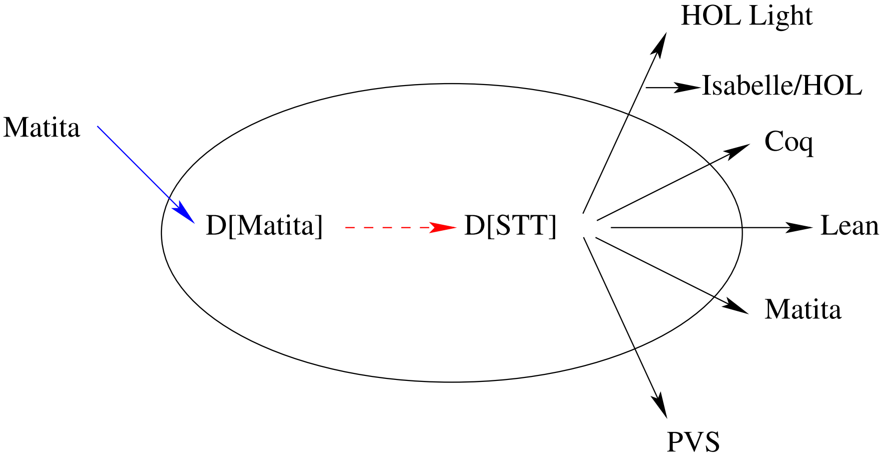
An example

All the proofs of the arithmetic library of MATITA

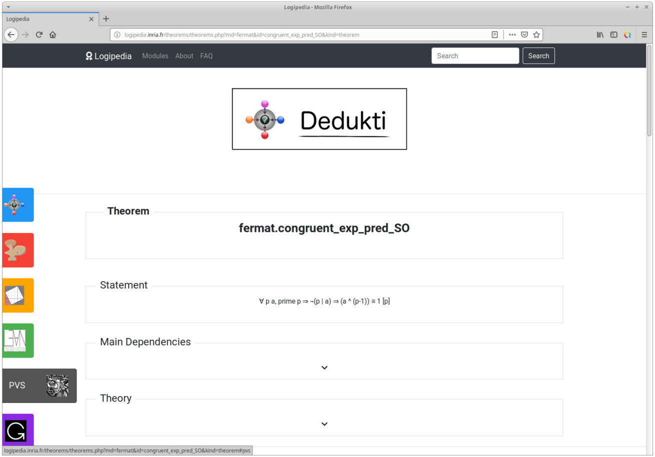
MATITA is not the Calculus of constructions: universes, inductive types and inductive definitions

"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



http://logipedia.science



Ongoing work: concept alignment in Logipedia

Connectives and quantifiers: inductive types / Q_0
Should be ignored by the library

Transporting structural statements between isomorphic structures

Making formal the saying: Cauchy sequences or Dedekind cuts immaterial

Ecumenical $\textsc{Dedukti}$

But classical and constructive disjunctions

Define $A \vee_c B$ as $\neg\neg(A \vee B)$
The double negation in front of the disjunction is carried by the $\neg\neg$

But no connective to carry the $\neg\neg$: various ecumenical logics (Girard, Prawitz, D, Pereira...)

Gilbert's connective $\rhd$
Not $P \Rightarrow Q$ but $(\rhd P) \Rightarrow (\rhd Q)$
Define $\rhd_c a$ as $\neg\neg \rhd a$

Already concrete results

While QED (1993) did not go very far

- Better understanding of the theories implemented in the various proof systems
- A new logical framework to express the these theories
- Analyzing the proofs (reverse mathematics) before we share them (partial translations)

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

Interoperability is not just a question of committees, negotiations, and standards: it is a proof theory problem