Logipedia: a system-independent encyclopedia of formal proofs

Gilles Dowek
In the early ages: write a piece of software (for example: text processing system) chose a representation for the data
Involuntarily defined a format

In modern times: define a format first
ASCII, TCP / IP, HTTP, HTML, UNICODE...
Software has to comply to the format

But not yet in the realm of formal proofs: “A Coq proof of the four color theorem”

Problems: interoperability, sustainability, cross-verification
Why is it more difficult with formal proofs?

Because we cannot go too far

Euclidean geometry $\not\leftrightarrow$ Hyperbolic geometry

ZF $\not\leftrightarrow$ ZFC
But...

A proof in $\text{ZF}$ can be “translated” to $\text{ZFC}$

A proof in $\text{ZFC}$ that does not use the axiom of choice can be “translated” to $\text{ZF}$
The interoperability $\text{ZF} / \text{ZFC}$ possible because $\text{ZF}$ and $\text{ZFC}$ 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 $\pi$. 
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 proof reduction: Deduction modulo theory
- Classical and not constructive: Ecumenical logic

The $\lambda\Pi$-calculus modulo theory that generalizes them all

**Dedukti**: an implementation of it
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
- Universe polymorphism
- Proof irrelevance
- Predicate subtyping
Simple type theory in **Dedukti**

\[ \begin{align*}
Set & : \text{TYPE} \\
\iota & : Set \\
El & : Set \to \text{TYPE} \\
Prop & : \text{TYPE} \\
Prf & : Prop \to \text{TYPE} \\
\Rightarrow & : Prop \to Prop \to Prop \\
(Prf (\Rightarrow x y)) & \to (Prf x) \to (Prf y) \\
\forall & : \Pi x : Set (((El x) \to Prop) \to Prop) \\
(Prf (\forall x y)) & \to \Pi z : (El x) (Prf (y z)) \\
o & : Set \\
(El o) & \to Prop \\
\rightsquigarrow & : Set \to Set \to Set \\
(El (\rightsquigarrow x y)) & \to (El x) \to (El y)
\end{align*} \]
The Calculus of constructions in Dedukti

\[
\begin{align*}
Set & : \text{TYPE} \\
\iota & : Set \\
El & : Set \to \text{TYPE} \\
Prop & : \text{TYPE} \\
Prf & : Prop \to \text{TYPE} \\
\Rightarrow & : \Pi x : Prop (((Prf x) \to Prop) \to Prop) \\
(Prf (\Rightarrow x y)) & \to \Pi z : (Prf x) (Prf (y z)) \\
\forall & : \Pi x : Set (((El x) \to Prop) \to Prop) \\
(Prf (\forall x y)) & \to \Pi z : (El x) (Prf (y z)) \\
o & : Set \\
(El o) & \to Prop \\
\leadsto & : \Pi x : Set (((El x) \to Set) \to Set) \\
(El (\leadsto x y)) & \to \Pi z : (El x) (El (y z)) \\
\pi & : \Pi x : Prop (((Prf x) \to Set) \to Set) \\
(El (\pi x y)) & \to \Pi z : (Prf x) (El (y z)) 
\end{align*}
\]
A comparison

- dependent in the Calculus of constructions but not in Simple type theory
- Same for $\Rightarrow$
- An extra symbol $\pi$ in the Calculus of constructions: express functions mapping proofs to terms

Have them all (ecumenical)
\[
\begin{align*}
\text{Set} & : \text{TYPE} \\
\iota & : \text{Set} \\
\text{El} & : \text{Set} \rightarrow \text{TYPE} \\
\text{Prop} & : \text{TYPE} \\
\text{Prf} & : \text{Prop} \rightarrow \text{TYPE} \\
\Rightarrow & : \text{Prop} \rightarrow \text{Prop} \rightarrow \text{Prop} \\
(\text{Prf} \Rightarrow x y) & \rightarrow (\text{Prf} x) \rightarrow (\text{Prf} y) \\
\Rightarrow_d & : \Pi x : \text{Prop} (((\text{Prf} x) \rightarrow \text{Prop}) \rightarrow \text{Prop}) \\
(\text{Prf} \Rightarrow_d x y) & \rightarrow \Pi z : (\text{Prf} x)(\text{Prf} y z) \\
\forall & : \Pi x : \text{Set} (((\text{El} x) \rightarrow \text{Prop}) \rightarrow \text{Prop}) \\
(\text{Prf} \forall x y) & \rightarrow \Pi z : (\text{El} x)(\text{Prf} y z) \\
o & : \text{Set} \\
(\text{El} o) & \rightarrow \text{Prop} \\
\sim \Rightarrow & : \text{Set} \rightarrow \text{Set} \rightarrow \text{Set} \\
(\text{El} \sim x y) & \rightarrow (\text{El} x) \rightarrow (\text{El} y) \\
\sim \Rightarrow_d & : \Pi x : \text{Set} (((\text{El} x) \rightarrow \text{Set}) \rightarrow \text{Set}) \\
(\text{El} \sim \Rightarrow_d x y) & \rightarrow \Pi z : (\text{El} x)(\text{El} y z) \\
\pi & : \Pi x : \text{Prop} (((\text{Prf} x) \rightarrow \text{Set}) \rightarrow \text{Set}) \\
(\text{El} \pi x y) & \rightarrow \Pi z : (\text{Prf} x)(\text{El} y z) 
\end{align*}
\]
Fragments

Minimal logic: \( \text{Set, } \iota, \text{ El, Prop, Prf, } \forall + \Rightarrow \)

Simple type theory: \( \ldots + \Rightarrow\), \( o, \sim \Rightarrow \)

Simple type theory: \( \ldots + \Rightarrow_d, o, \sim \Rightarrow_d, \pi \)

- A proof that does not use \( \Rightarrow \) and \( \sim \Rightarrow \) is expressed in the Calculus of constructions

- A proof that does not use \( \Rightarrow_d, \sim \Rightarrow_d \) and \( \pi \) is a proof in Simple type theory
Translations

All proofs in Simple type theory can be translated to the Calculus of constructions: translate $\Rightarrow$ to $\Rightarrow_d$ and $\sim$ to $\sim_d$

Some proofs in the Calculus of constructions can be translated to Simple type theory: if dependency not used, translate $\Rightarrow_d$ to $\Rightarrow$ and $\sim_d$ to $\sim$

(not the others: genuine Calculus of constructions proofs)
An example

The same proofs of Fermat’s little theorem in six systems
Why does it work so well?

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.
Collecting all the proofs in a single data base

**Logipedia**: an encyclopedia of proofs expressed
- in various theories
- in Dedukti
Theorem

fermat.congruent_exp_pred_SO

Statement

\forall p, a, \text{prime } p \rightarrow \neg (p | a) \rightarrow a^p \equiv 1 \pmod{p}

Main Dependencies

Theory

http://logipedia.science
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 these theories
- Analyzing the proofs before we share them (partial translations)
Interoperability is not just a question of committees, negotiations, and standards: it is a research problem.