Logipedia: a system-independent encyclopedia of formal proofs

Gilles Dowek
Formats

In the early ages: write a piece of software (for example: text processing system) choose 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”, “A HOL Light proof of Hales’ theorem”

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

Because we cannot go too far

Euclidean geometry $\nleftrightarrow$ Hyperbolic geometry

$ZF \nleftrightarrow ZFC$

No way to choose a standard for proofs
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 $\text{ZF}$
A little bit better: proof transformation

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

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

by hand or automatically (for example: constructivization)

Reverse mathematics (transforming proofs so that they squeeze in smaller theories) as the basis of interoperability
Reformulating the project of reverse mathematics

- **Formal** proofs, not pencil-paper-$\LaTeX$ ones
- **Expressive** theories (Set theory, Type theory...) and not fragments of arithmetic
- **Analyze** proofs before (possibly) transforming them
I. Logical frameworks
The interoperability $\text{ZF} / \text{ZFC}$ possible because $\text{ZF}$ and $\text{ZFC}$ expressed in the same **logical framework**: Predicate logic

No standard for proofs because they are expressed in different theories, but a standard for language in which we define theories

Moreover, a theory in Predicate logic: **several** axioms

Permits to raise the question: which axioms are used in a proof $\pi$
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 framework Predicate logic for geometry, arithmetic, set theory...

Sharing connectives, deduction rules...

A theory: symbols and axioms
But a short revolution

Predicate logic: at that time another theory: Type theory
(Principia Mathematica)
No expression in Predicate logic

Soon (1940) Church: a new formulation of Type theory (based on \(\lambda\)-calculus) impossible to express in Predicate logic (\(\lambda\) binds)

1970, 1985... Martin-Löf’s type theory, the Calculus of constructions... Type theory with predicate subtyping 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...)
- Transform 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 proof reduction
- Classical and not constructive
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
II. Minimal, Constructive, and Ecumenical Predicate logic in Dedukti
Terms and propositions of Minimal predicate logic

A weak framework: no notion of term, proposition, proof...
But tools to build them

\[ I : Type \]
\[ 0 : I, S : I \rightarrow I \ldots \]

\[ prop : Type \]
\[ \Rightarrow : prop \rightarrow prop \rightarrow prop \]
\[ \forall : (I \rightarrow prop) \rightarrow prop \]
\[ even : I \rightarrow prop \]
\[ (\forall x A = \forall \lambda x : I A) \]
Proofs of Minimal predicate logic

Brouwer-Heyting-Kolmogorov: a proof of $A \Rightarrow B$ is a typed term expressing a function mapping proofs of $A$ to proofs of $B$

$\text{prf} : \text{prop} \rightarrow \text{Type}$

$(\text{prf } A)$ the type of the proofs of $A$

$\lambda x : (\text{prf } P) \times \text{maps proofs of } P \text{ to proofs of } P$

Should be a proof of $P \Rightarrow P$, that is a term of type $\text{prf} (P \Rightarrow P)$

But has type $(\text{prf } P) \rightarrow (\text{prf } P)$

$\text{prf} (x \Rightarrow y) \rightarrow (\text{prf } x) \rightarrow (\text{prf } y)$

$\text{prf} (\forall p) \rightarrow \Pi z : I (\text{prf} (p z))$
Constructive Predicate logic

In the theory $\forall$ quantifier on elements of $I$
No quantification over propositions in particular no $\forall C ((A \Rightarrow B \Rightarrow C) \Rightarrow C)$

But, in the framework $\Pi x : A \ B$ for any $A$ and $B$ of type $Type$, in particular $\Pi z : prop (((prf x) \rightarrow (prf y) \rightarrow (prf z)) \rightarrow (prf z))$

Use this to define $\wedge$
$\wedge : prop \rightarrow prop \rightarrow prop$
$(prf (x \wedge y))$
$\rightarrow \Pi z : prop (((prf x) \rightarrow (prf y) \rightarrow (prf z)) \rightarrow (prf z))$

Same for $\top$, $\bot$, $\neg$, $\lor$, and $\exists$
Ecumenical predicate logic

Why does the Deduction rule

\[
\frac{A}{A \lor B}
\]

hold?

- Because it holds in nature
- Because our brain is wired in this way
- Because it defines the meaning of the symbol $\lor$

Constructive and classical rules define different meanings. Fighting about the excluded middle is like fighting about “disjunction is exclusive”: depends on what you call “disjunction”

Different symbols: $\lor$ and $\lor_c$ that can coexist in the same language just like $\lor$ and $\oplus$

How can we define the classical connectives in Dedukti?
Gödel, Gentzen, Kolmogorov...

A has a classical proof if and only if $|A|$ has a constructive proof

For instance

$|(P \land Q) \Rightarrow P| = \neg \neg(\neg \neg((\neg \neg P) \land (\neg \neg Q))) \Rightarrow (\neg \neg P))$

Use it as a definition: find $\Rightarrow_c, \land_c ...$ such that $(P \land_c Q) \Rightarrow_c P$ is

$\neg \neg(\neg \neg((\neg \neg P) \land (\neg \neg Q))) \Rightarrow (\neg \neg P))$

$\land_c$ should be $\land + \neg \neg$ somewhere
Three possibilities

1. (Prawitz) \( A \land_c B = \neg \neg (A \land B) \)
2. (Hermant, Allali) \( A \land_c B = (\neg \neg A) \land (\neg \neg B) \)
3. (D) \( A \land_c B = \neg \neg ((\neg \neg A) \land (\neg \neg B)) \)

\((P \land_c Q) \Rightarrow_c P\) then is

1. \( \neg \neg ((\neg \neg (P \land Q)) \Rightarrow P) \)
2. \( (\neg \neg((\neg \neg P) \land (\neg \neg Q))) \Rightarrow (\neg \neg P) \)
3. \( \neg \neg((\neg \neg \neg((\neg \neg P) \land (\neg \neg Q))) \Rightarrow (\neg \neg P)) \)

None of them exactly is

\( \neg \neg((\neg \neg((\neg \neg P) \land (\neg \neg Q))) \Rightarrow (\neg \neg P)) \)
Fixing definitions

Fixing 1.:
Prawitz: introduce a classical predicate symbol as well $P_c = \neg\neg P$
Gilbert: changing the syntax of propositions

\[
\begin{align*}
t &= x \mid f(t, \ldots, t) \\
a &= P(t, \ldots, t) \\
A &= \text{\texttt{\triangleright}}a \mid A \land A \mid A \Rightarrow A \mid \ldots \\
\text{\texttt{\triangleright c}} &= \neg\neg\text{\texttt{\triangleright}} \\
(\text{\texttt{\triangleright c}}P \land \text{\texttt{\triangleright c}}Q) \Rightarrow \text{\texttt{\triangleright c}}P \text{ is then exactly what you wanted}
\end{align*}
\]

Fixing 2.:
Grienenberger:

\[
\begin{align*}
t &= x \mid f(t, \ldots, t) \\
p &= P(t, \ldots, t) \mid p \land p \mid p \Rightarrow p \mid \ldots \\
A &= \text{\texttt{\triangleleft}}p \\
\text{\texttt{\triangleleft c}} &= \neg\neg\text{\texttt{\triangleleft}} \\
\text{\texttt{\triangleleft c}}((P \land \text{\texttt{\triangleleft c}}Q) \Rightarrow \text{\texttt{\triangleleft c}}P) \text{ is then exactly what you wanted}
\end{align*}
\]
Solution 2.
Remember $p : \text{prop}$ a proposition then $(\text{prf } p)$ type of its proofs
Incorporate $\triangleleft$ and $\triangleleft_c$ in $\text{prf}$: $\text{prf}$ and $\text{prf}_c$

\[
\text{prf}_c : \text{prop} \rightarrow \text{Type} \\
(\text{prf}_c \; x) \rightarrow (\text{prf} (\neg \neg x))
\]

\[
\Rightarrow_c : \text{prop} \rightarrow \text{prop} \rightarrow \text{prop} \\
x \Rightarrow_c y \rightarrow (\neg \neg x) \Rightarrow (\neg \neg y)
\]

\[
\land_c : \text{prop} \rightarrow \text{prop} \rightarrow \text{prop} \\
x \land_c y \rightarrow (\neg \neg x) \land (\neg \neg y)
\]

...
III. More theories
Ecumenical predicate logic is just one paradigmatic example
Several others

- Any (finite) theory expressed in Predicate logic
- Axiom schemes
- Simple type theory (without and with polymorphism) (Assaf)
- Pure type systems (The Calculus of constructions) (Cousineau, D)
- Inductive types (Boespflug, Burel)
- Universes (Assaf)
- Predicate subtyping (Hondet)
- Proof irrelevance (Thiré, Férey)
- Universe polymorphism (Férey, Genestier)
Simple type theory in Dedukti

set : Type

ι : set

el : set → Type

prop : Type

prf : prop → Type

⇒ : prop → prop → prop

(prf (⇒ x y)) → (prf x) → (prf y)

∀ : Πx : set (((el x) → prop) → prop)

(prf (∀ x y)) → Πz : (el x) (prf (y z))

o : set

(el o) → prop

arrow : set → set → set

(el (arrow x y)) → (el x) → (el y)
The Calculus of constructions in Dedukti

\[
\begin{align*}
\text{set} & : \ Type \\
\iota & : \ set \\
\text{el} & : \ set \to Type \\
\text{prop} & : \ Type \\
\text{prf} & : \ prop \to Type \\
\Rightarrow_d & : \ \Pi x : \ prop (((\text{prf} x) \to \ prop) \to \ prop) \\
(\text{prf} (\Rightarrow_d x y)) & \to \ \Pi z : (\text{prf} x) (\text{prf} y z) \\
\forall & : \ \Pi x : \ set (((\text{el} x) \to \ prop) \to \ prop) \\
(\text{prf} (\forall x y)) & \to \ \Pi z : (\text{el} x) (\text{prf} y z) \\
o & : \ set \\
(\text{el} o) & \to \ prop \\
\text{arrow}_d & : \ \Pi x : \ set (((\text{el} x) \to \ set) \to \ set) \\
(\text{el} (\text{arrow}_d x y)) & \to \ \Pi z : (\text{el} x) (\text{el} y z) \\
\pi & : \ \Pi x : \ prop (((\text{prf} x) \to \ set) \to \ set) \\
(\text{el} (\pi x y)) & \to \ \Pi z : (\text{prf} x) (\text{el} y z)
\end{align*}
\]
Very few differences

- $\rightarrow$ replaced by $\rightarrow_d$
- $\Rightarrow$ replaced by $\Rightarrow_d$
- an extra symbol $\pi$

And very few differences with Minimal predicate logic (propositional contents and functionality)

Again: be Ecumenical, have all these symbols and rules and write your proofs in the fragment you like
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 (Thiré): all the proofs of the arithmetic library of Matita

“First” proof of Fermat’s little theorem in constructive Simple type theory.
Collecting all the proofs in a single data base

**Logipedia**: an encyclopedia of proofs expressed

- in various theories (using various symbols and rules)
- in Dedukti

http://logipedia.science
Why does it work so well?

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

Two 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
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 (reverse mathematics) before we share them (partial translations)
Interoperability is not just a question of committees, negotiations, and standards: it is a research problem