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) 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”, “A HOL Light proof of Hales’ theorem”
Problems

- interoperability
- sustainability
- universality of logical truth

But not the first time: are the axiom of parallels, the excluded middle, $\exists x \ (2x = 17)$ true or not?

But a solution: different theories speaking of different things
“$A$ is true” relative
“A is true in $\mathcal{T}$” absolute
Why are proofs checking and text processing different?

Because, with proofs, we cannot go too far

Euclidean geometry \( \not\leftrightarrow \) Hyperbolic geometry

ZF \( \not\leftrightarrow \) ZFC
But...

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

A proof in \textit{ZFC} that does not use the axiom of choice can be “translated” to \textit{ZF}
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

Reverse mathematics 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
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 $\pi$. 
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

Geometry, arithmetic, set theory expressed in predicate logic, but **not** type theory (*Principia Mathematica*)

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... **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 ($\lambda x \ x$): $\lambda$-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

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

Ongoing: universe polymorphism, proof irrelevance, predicate subtyping
Simple type theory in Dedukti

\[
\text{type} : \quad \text{Type} \\
\eta : \quad \text{type} \to \text{Type} \\
o : \quad \text{type} \\
nat : \quad \text{type} \\
\text{arrow} : \quad \text{type} \to \text{type} \to \text{type} \\
\varepsilon : \quad (\eta \circ o) \to \text{Type} \\
\Rightarrow : \quad (\eta \circ o) \to (\eta \circ o) \to (\eta \circ o) \\
\forall : \quad \Pi x : \text{type} (((\eta x) \to (\eta o)) \to (\eta o))
\]

\[
(\eta (\text{arrow} \times y)) \quad \rightarrow \quad (\eta x) \to (\eta y) \\
(\varepsilon (\Rightarrow \times y)) \quad \rightarrow \quad (\varepsilon x) \to (\varepsilon y) \\
(\varepsilon (\forall \times y)) \quad \rightarrow \quad \Pi z : (\eta x) (\varepsilon (y z))
\]
The Calculus of constructions in **Dedukti**

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

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

- *arrow dependent* in the Calculus of constructions but not in Simple type theory
- Same for \( \Rightarrow \)
- 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...)
Proof translation

T

\[ D[U] \]

\[ D[T] \]

\[ D[V] \]

U

V
But also
An example
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
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

\( \forall p, a, \text{prime } p \rightarrow (p | a) \rightarrow (a^n (p-1)) = 1 \mod p) \)

Main Dependencies

Theory

---

http://logipedia.science
Towards concept alignment in Logipedia

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

Making formal the saying: Cauchy sequences or Dedekind cuts immaterial (isomorphic and only structural statements)

But classical and constructive disjunctions (ecumenical logic)
Current work

Proofs of Matita, Lean, Coq, HOL Light, HOL4, Isabelle/HOL, Agda, Rodin, Atelier-B (easy) PVS, Mizar (doable)

But what is next: SAT solvers, SMT solvers, Automated theorem provers, model checkers?
“We do not need proofs in model checking”

In general: “$A$ is true” undecidable, “$\pi$ is a proof of $A$” decidable
But propositional modal logics: “$A$ is true” decidable

Proofs are useless: instead of checking that $\pi$ is a proof of $A$, check directly that $A$ is true
And indeed

\[(c_1/x)A \ldots (c_n/x)A \]
\[\forall x \ A\]

A proof of

\[\forall x \in [0, 999] (\text{even}(x) \lor \text{odd}(x))\]

is a tree with 1000 branches

Checking the 1000 branches of the proof not easier than checking the 1000 numbers
But what about the complexity

Can it be that
- “$A$ is true” decidable with a high complexity
- “$\pi$ is a proof of $A$” decidable with a low complexity?

With $\forall$: no

But with $\exists$?

\[
\frac{(c_i/x)A}{\exists x \ A}
\]

$P$ predicate that holds for 999 only
Model checking $\exists x \ P(x)$: 1000 operations
Checking

\[
\frac{P(999)}{\exists x \ P(x)}
\]

two operations
A more serious example

Checking trace of deterministic program as complex as executing it
Checking trace of non deterministic program much faster

Finding a solution to a SAT problem \textit{NP}-complete
Checking a solution to a SAT problem in \textit{P}
Once you have found a proof keep it for future use

The market of rechecking with small checkers

Model-checking is proof construction, not proof checking
Proofs already exist in model-checking

But they are called counter examples

Example in SCTL (Jiang-D): two rules for the CTL modality EF

\[
\begin{align*}
(s/x)A & \frac{EF_x A(s)}{EF_x A(s)} \\
EF_x A(s) & \frac{EF_x A(s')}{EF_x A(s)} s' \in \text{Next}(s)
\end{align*}
\]

A proof of \( EF_x P(x)(s) \): almost a finite sequence \( s_0, \ldots, s_n \) such that \( s_0 = s, s_{i+1} \in \text{Next}(s_i) \), and \( P(s_n) \) holds

An “example” of \( EF_x P(x)(s) \)

(often called a counter example to \( AG_x \lnot P(x)(s) \))
Yet are proofs slightly more general

$$EF_x \ (P(x) \land EF_y \ Q(y)(x))(s)$$

Same but $P(s_n) \land EF_y \ Q(y)(s_n)$ must then be justified with another sequence

SCTL proofs smoothly integrate these two sequences
Towards propositional modal logic proofs in Logipedia?

A formulation of SCTL in Polarized Deduction modulo theory (Ji)

\texttt{Dedukti} supports Deduction modulo theory but not Polarized Deduction modulo theory

(For other reasons) a polarized extension of \texttt{Dedukti} (Burel)
To be done

Define SCTL in (polarized) Dedukti

Translate the SCTL proofs we have (Liu, et al.) to Logipedia

So that they can be used by everyone