Analyzing individual proofs as the basis of interoperability between proof systems

Gilles Dowek

Maurice Nivat

1937-2017

Among other things

- Pioneer of automata theory
- Created the french informatics community
- The advisor of my advisor
- Created ICALP, EATCS and TCS
- Activist for informatics in primary and secondary education
- A friend of mine

With Maurice Gross

Analyzing individual proofs as the basis of interoperability between proof systems

Gilles Dowek

Studying generic objects and specific ones

Mass of Mercury vs. Kepler's laws

In Mathematics: mostly generic objects, but also π

In Computer science: algorithms that apply to generic data vs. specific pieces of data

In Proof theory

Mostly generic proofs
Gentzen's cut elimination theorem: all proofs of Predicate logic

Study the proofs we have instead of those we may build

Changes the perspective:
Cut elimination for a logic $\mathcal{L}$: the stronger the logic $\mathcal{L}$, the stronger the theorem
The proof π can be expressed in $\mathcal{L}$: the weaker the logic, the stronger the theorem

Focus on ~~stronger and stronger~~ weaker and weaker logics

A program

Analyze the formal proofs that have been developed in Coq, Matita, HOL Light, Isabelle/HOL, PVS, FoCaLiZe...

In which logics can each of these proofs be expressed?

Reverse mathematics

An important source of inspiration

But some differences:

- analyze proofs not theorems
- focus on formal proofs expressed and checked in computerized proof systems
- less ambitious: the Calculus of constructions, Simple type theory... rather than fragments of Second-order arithmetic

In which logics can this proof be expressed?

A fundamental question
Part of our understanding of this proof

But also a practical one: interoperability between computerized proof systems
Coq proofs, HOL Light proofs: no (little) exchange
Not just a translation problem: different logics

I. From logics to theories

Logical framework

A proof π expressed in a logic $\mathcal{L}$

In which (other) logics can it be expressed?

Which ingredients of $\mathcal{L}$ does it use?

Decompose $\mathcal{L}$ into a number of ingredients, e.g. axioms

Express it in a logical framework, e.g. Predicate logic

Example: set theory, does π use the axiom of choice?

Logical frameworks

Predicate logic, Pure type systems, $\lambda\Pi$-calculus...

$\lambda\Pi$-calculus modulo theory aka Martin-Löf logical framework
Ingredients: axioms and rewrite rules
An implementation: DEDUKTI

II. Translating proofs expressed in the Calculus of constructions into Simple type theory

Simple type theory as a theory in the $\lambda\Pi$-calculus modulo theory

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

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

Examples

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

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

Propositions: $\forall X : o\ (X \Rightarrow X)$ 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
$\Pi X : (\eta\ o)\ ((\varepsilon\ X) \to (\varepsilon\ X))$

(A slight extension of) the Calculus of constructions as a theory in the $\lambda\Pi$-calculus modulo theory

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

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

(A slight extension of) the Calculus of constructions as a theory in the $\lambda\Pi$-calculus modulo theory

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

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

Comparing the theories

arrow in Simple type theory

$$\Pi x : type\ (type \to type)$$

in the Calculus of constructions

$$\Pi x : type\ (((\eta\ x) \to type) \to type)$$

In the Calculus of constructions, dependent *arrow*: in $A \to B$ (written $\Pi x : A\ B$), B can contain a variable x of type A

Same for $\Rightarrow$
($\forall$ is dependent is both theories)

An extra constant π in the Calculus of constructions: typing functions mapping proofs to terms

Analyzing proofs expressed in the Calculus of constructions

A subset S of the proofs expressed in the Calculus of constructions

- do not use the dependency of *arrow*
- do not use the dependency of the symbol $\Rightarrow$,
- do not use the symbol π

Many proofs expressed in the Calculus of constructions in S

Translating proofs to Simple type theory

A proof in the Calculus of constructions

In S

Translation to Simple type theory:
replace $(\mathit{arrow}\ A\ \lambda x : (\eta\ A)\ B)$ with $(\mathit{arrow}\ A\ B)$
(similar for $\Rightarrow$)

Not in S

Genuinely uses a feature of the Calculus of constructions that does not exist in Simple type theory
Cannot be expressed in Simple type theory
Same as in ZFC: genuinely uses the axiom of choice: not in ZF

Same proof (across theories): to be extended

III. An arithmetic library

Coq and Matita

The Calculus of constructions + inductive types, universes...

Boespflug, Burel, Assaf: inductive types, universes in the $\lambda\Pi$-calculus modulo theory

The arithmetic library of Matita in Dedukti, including a proof of Fermat's little theorem

Thiré: dependency of *arrow* and $\Rightarrow$, π, and universes can be eliminated from this library
Inductive types : replaced by a induction on natural numbers

Fermat's little theorem

A proof in constructive Simple type theory

Novelty: a formal proof in a theory weaker than MATITA
Also weaker than HOL LIGHT (excluded middle, extensionality, choice...)

Exporting this library

From Dedukti
To HOL Light, Isabelle/HOL, HOL4 (using OpenTheory)
To Coq

1.5 Mo, 340 lemmas

IV. Abstracting enough

Natural numbers

Both in Matita and HOL Light
Proving propositions by induction / defining functions by induction
But justified in different ways
Inductive type vs. impredicative definition of finite cardinals

Ignored by the library
Any system containing a notion of natural number and an induction principle

Connectives and quantifiers

Same for 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)

V. Mixing classical and constructive logics

From agnosticism to ecumenism

A logical framework neither constructive of classical
Permits to express both constructive and classical theories
A possibility: not to assume the excluded middle, include it in some theories

Alternative: not a question of theory, but of meaning of the connectives and quantifiers
Two existential quantifiers: constructive ($\exists$) and classical ($\exists_c$), two disjunctions...
Connective once for all, use various connectives and quantifiers in various theorems
Again: analysis, translations

VI. Future work

Arithmetic library: the beginning of a shared library

A formal proof of Fermat's little theorem, in constructive Simple type theory: weaker theories (predicative, PA...)