A proof library shared by different proof systems

Gilles Dowek

Sharing data

A C program can be executed on any computer

A jpg, png... photo can be seen on any telephone, computer...

A webpage can be displayed in any browser

Sharing proofs accross systems

A PVS proof of $x + y = y + x$

A HOL Light proof of $x + y = y + x$

A Coq proof of $x + y = y + x$

...

Although

We all would like to have a proof of

- Hales' theorem
- the correctness of ACCoRD
- the Four color theorem
- Heule's theorem

in PVS, Coq, HOL Light...

Why don't we have it?

Lack of standard

But also different logics

I. Building a shared library in five steps

Step 1: A logical framework

Different logics: nothing new

ZF $\vdash$ Every vector has a unique decomposition in a base

ZF $\nvdash$ Every vector space has a base
ZFC $\vdash$ Every vector space has a base

From logics to theories

But... ZF and ZFC are expressed in the same logical framework (Predicate logic)

Only the axioms differ

Easy to analyse if a proof uses the axiom of choice or not

Logical frameworks

- Predicate logic
- $\lambda\Pi$-calculus (Harper, Honsell, Plotkin, 1993): proof-terms, binders
- Deduction modulo theory (D, Hardin, Kircher, 2003): computations, cut elimination
- $\lambda\Pi$-calculus modulo theory (Cousineau, D, 2007) implemented in Dedukti (Boespflug, Saillard, et al.)
- Many others: LFSC, ProofCert...

Step 2: Expressing logics in DEDUKTI

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: $nat \to nat$ expressed as $(arrow\ nat\ nat)$ of type $type$
Then as $(\eta\ (arrow\ nat\ nat))$ 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: $\forall X : o\ (X \Rightarrow X)$ expressed as
$\forall\ o\ \lambda X : (\eta\ o)\ (\Rightarrow\ X\ X)$ of type $(\eta\ o)$
Then as $(\varepsilon\ (\forall\ o\ \lambda X : (\eta\ o)\ (\Rightarrow\ X\ X)))$ 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
$\Pi X : (\eta\ o)\ ((\varepsilon\ X) \to (\varepsilon\ X))$

The Calculus of constructions as a theory in the $\lambda\Pi$-calculus modulo theory

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

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

Coq and Matita

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

Calculus of constructions with inductive types, universes... in Dedukti (Boespflug, Burel, Assaf, 2015)

Step 3: Translating proofs to Dedukti

HOL Light proofs: Assaf (2015)
Matita proofs: Assaf (2015)
FoCaLiZe proofs: Cauderlier, Dubois (2016)

Zenon modulo proofs: Halmagrand (2016)
i-prover modulo proofs: Burel (2014)

On going: Coq proofs, SAT proofs, SMT proofs...

Step 4: Reverse mathematics

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

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

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

The Calculus of constructions as a theory in the $\lambda\Pi$-calculus modulo theory

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

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

The Calculus of constructions as a theory in the $\lambda\Pi$-calculus modulo theory

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

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

Analyzing proofs expressed in the Calculus of constructions

A subset 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

Can be translated to Simple type theory:
just replace ($arrow\ A\ \lambda x : (\eta\ A)\ B$) with ($arrow\ A\ B$)
(similar for $\Rightarrow$)

Otherwise

The proof genuinely uses a feature of the Calculus of constructions that does not exist in Simple type theory

Should be labeled as such

Same as in ZFC: genuinely uses the axiom of choice: not in ZF

The arithmetic library of Matita

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

Dependency of *arrow* and $\Rightarrow$, π, and universes can be eliminated from this library (Thiré, 2018)

Inductive types: replaced by a induction on natural numbers

Actual proofs are much simpler than what is allowed by the logic

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...)

(Genuine) reverse mathematics

Friedman, Simpson...
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

Step 5: Exporting from DEDUKTI

Exporting this library

From DEDUKTI

To HOL LIGHT, ISABELLE/HOL, HOL4 (using OPENTHEORY)
To COQ and (of course) to MATITA

1.5 Mo, 340 lemmas

https://github.com/francoisthire/SharingAnArithmeticLibrary

II. 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
Left to the host (the proof must land on a comfortable enough pillow)

Any system containing a notion of natural number and an induction principle

Connectives and quantifiers

Same as natural numbers
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)

III. What about PVS?

Using the library

PVS contains Simple type theory

The full arithmetic library can be translated to PVS (or has it been translated already?)

Contributing to the library

Express PVS in Dedukti

What is the logic of PVS already? (Gilbert, 2018)

Can it be expressed in Dedukti?

Future work

Arithmetic library: the beginning of a shared library

Label each lemma by the rewrite rules and axioms it requires

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