EuroProofNet

Introduction to Proof Systems Interoperability

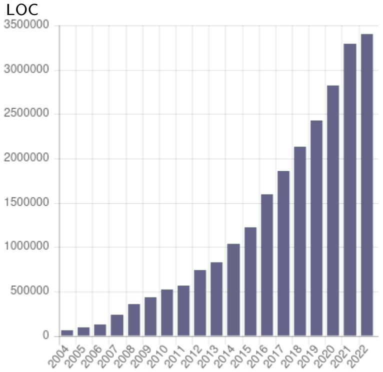
Frédéric Blanqui

Deduc⊢eam

Libraries of formal proofs today

Library	Nb files	Nb objects*
Coq Opam	35,000	1,200,000
Isabelle AFP	7,500	280,000
Lean Mathlib	4,200	210,000
Mizar Mathlib	1,400	77,000
HOL-Light	600	35,000
...	...	...

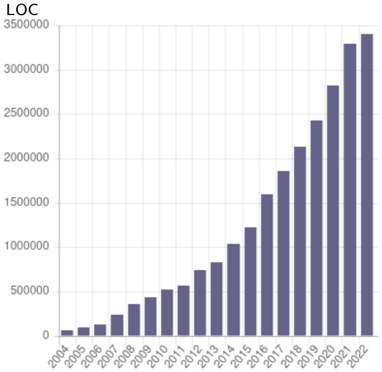
* type, definition, theorem, ... in 2023



Libraries of formal proofs today

Library	Nb files	Nb objects*
Coq Opam	35,000	1,200,000
Isabelle AFP	7,500	280,000
Lean Mathlib	4,200	210,000
Mizar Mathlib	1,400	77,000
HOL-Light	600	35,000
...	...	...

* type, definition, theorem, ... in 2023

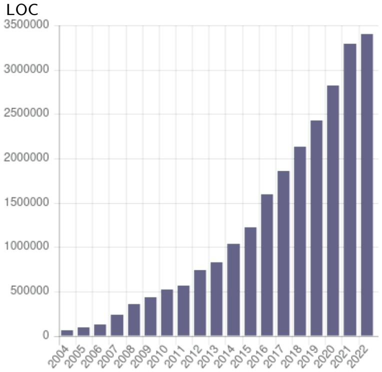


- Every system has basic libraries on integers, lists, ...
- Some definitions/theorems are available in one system only

Libraries of formal proofs today

Library	Nb files	Nb objects*
Coq Opam	35,000	1,200,000
Isabelle AFP	7,500	280,000
Lean Mathlib	4,200	210,000
Mizar Mathlib	1,400	77,000
HOL-Light	600	35,000
...	...	...

* type, definition, theorem, ... in 2023



- Every system has basic libraries on integers, lists, ...
- Some definitions/theorems are available in one system only

$\Rightarrow$ Can't we translate a proof between two systems automatically?

Interest of proof systems interoperability

- Avoid duplicating developments and losing time
- Facilitate development of new proof systems
- Increase reliability of formal proofs (cross-checking)
- Facilitate validation by certification authorities
- Relativize the choice of a system (school, industry)
- Provide multi-system data to machine learning

Difficulties of interoperability

- Each system is based on different axioms and deduction rules
- It is usually non trivial and sometimes impossible to translate a proof from one system to the other (e.g. a proof using impredicativity or proof irrelevance in a system not allowing these features)

Difficulties of interoperability

- Each system is based on different axioms and deduction rules
- It is usually non trivial and sometimes impossible to translate a proof from one system to the other (e.g. a proof using impredicativity or proof irrelevance in a system not allowing these features)
- Is it reasonable to have $n(n-1)$ translators for n systems?

$$\begin{tikzcd}
 & 1 \arrow[dr] \arrow[ddr] \arrow[dddr] & 1 \\
n(n-1) & 2 \arrow[ur] \arrow[dr] \arrow[ddr] & 2 \\
 & 3 \arrow[uur] \arrow[ur] \arrow[dr] & 3 \\
 & 4 \arrow[uuur] \arrow[uur] \arrow[ur] & 4
\end{tikzcd}$$

Difficulties of interoperability

- Each system is based on different axioms and deduction rules
- It is usually non trivial and sometimes impossible to translate a proof from one system to the other (e.g. a proof using impredicativity or proof irrelevance in a system not allowing these features)
- Is it reasonable to have $n(n-1)$ translators for n systems?

$$\begin{tikzcd}
& 1 \arrow[rdd] \arrow[rdddd] \arrow[rdddddd] & 1 & 1 \arrow[rddd] & & 1 & \\
& & & & & & \\
& 2 \arrow[ruu] \arrow[rdd] \arrow[rdddd] & 2 & 2 \arrow[rd] & & 2 & \\
n(n-1) & & & & D \arrow[ruuu] \arrow[ru] \arrow[rd] \arrow[rddd] & & 2n \\
& 3 \arrow[ruuuu] \arrow[ruu] \arrow[rdd] & 3 & 3 \arrow[ru] & & 3 & \\
& & & & & & \\
& 4 \arrow[ruuuuuu] \arrow[ruuuu] \arrow[ruu] & 4 & 4 \arrow[ruuu] & & 4 &
\end{tikzcd}$$

A common language for proof systems?

Logical framework D

language for describing axioms, deduction rules and proofs of a system S as a theory $D(S)$ in D

Example: D = predicate calculus

allows one to represent S=geometry, S=arithmetic, S=set theory, ...
not well suited for computations and dependent types

A common language for proof systems?

Logical framework D

language for describing axioms, deduction rules and proofs of a system S as a theory $D(S)$ in D

Example: $D =$ predicate calculus

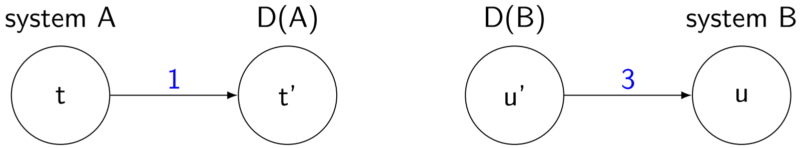
allows one to represent S=geometry, S=arithmetic, S=set theory, ...
not well suited for computations and dependent types

Better: $D = \lambda\Pi$-calculus modulo rewriting ($\lambda\Pi/\mathcal{R}$)

allows one to represent also:
S=HOL, S=Coq, S=Agda, S=PVS, ...

How to translate a proof $t \in A$ in a proof $u \in B$?

In a logical framework D:

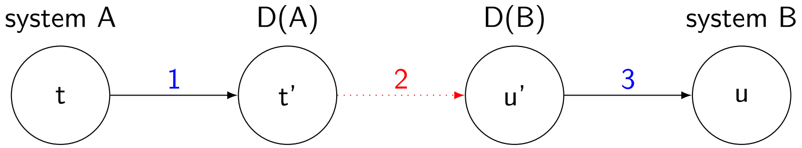


1. translate $t \in A$ in $t' \in D(A)$

3. translate $u' \in D(B)$ in $u \in B$

How to translate a proof $t \in A$ in a proof $u \in B$?

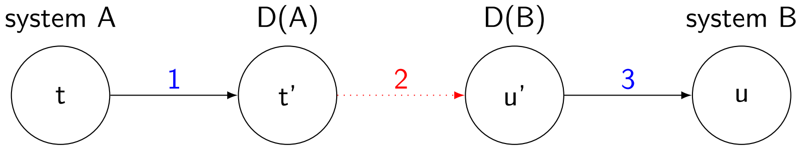
In a logical framework D:



1. translate $t \in A$ in $t' \in D(A)$
2. identify the axioms and deduction rules of A used in t'
   translate $t' \in D(A)$ in $u' \in D(B)$ if possible
3. translate $u' \in D(B)$ in $u \in B$

How to translate a proof $t \in A$ in a proof $u \in B$?

In a logical framework D:

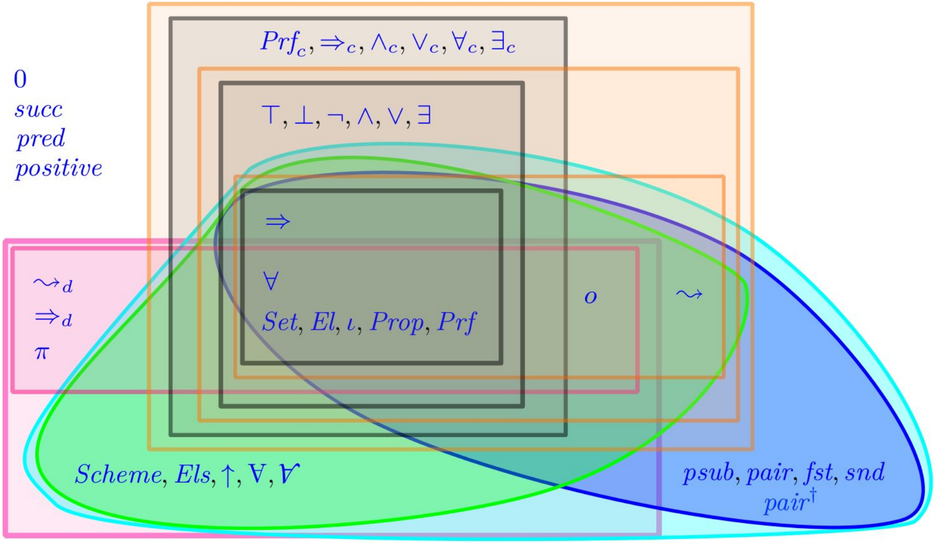


1. translate $t \in A$ in $t' \in D(A)$
2. identify the axioms and deduction rules of A used in t'
   translate $t' \in D(A)$ in $u' \in D(B)$ if possible
3. translate $u' \in D(B)$ in $u \in B$

$\Rightarrow$ represent in the same way functionalities common to A and B

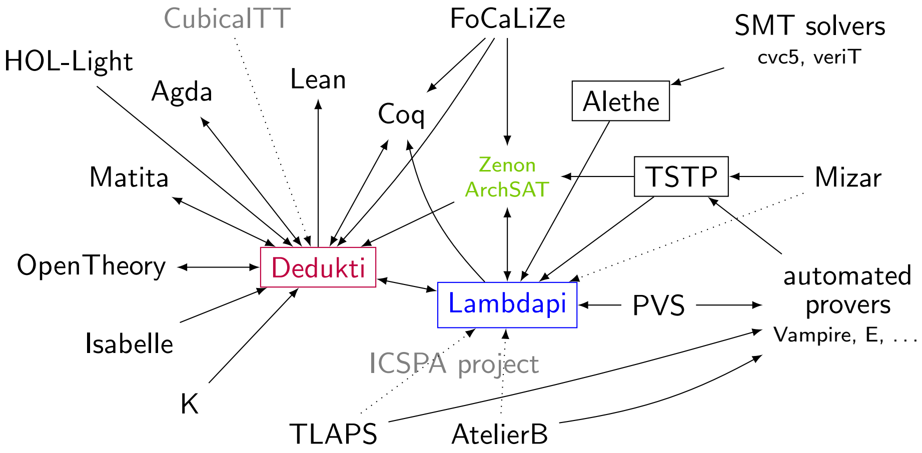
The modular $\lambda\Pi/\mathcal{R}$ theory U and its sub-theories

38 symbols, 28 rules, 13 sub-theories



0
$succ$
$pred$
$positive$
$Prf_c, \Rightarrow_c, \wedge_c, \vee_c, \forall_c, \exists_c$
$\top, \bot, \neg, \wedge, \vee, \exists$
$\Rightarrow$
$\forall$
$Set, El, \iota, Prop, Prf$
o
$\leadsto$
$\leadsto_d$
$\Rightarrow_d$
π
$Scheme, Els, \uparrow, \dot{\forall}, \forall\!\!\!/$
$psub, pair, fst, snd$
$pair^\dagger$

Dedukti, an assembly language for proof systems



Lambdapi = Dedukti + implicit arguments/coercions, tactics, ...

`https://github.com/Deducteam/Dedukti`
`https://github.com/Deducteam/lambdapi`

Libraries currently available in Dedukti

System	Libraries
HOL-Light	all libraries
Matita	Arith
Coq	Stdlib parts, GeoCoq
Isabelle	HOL.Complex_Main (*AFP soon?*)
Agda	Stdlib parts ($\pm$ 25%)
PVS	Stdlib parts (statements only)
TPTP	E 69%, Vampire 83%

Examples of library translations

- `https://logipedia.inria.fr`:
  Matita/Arith $\longrightarrow$ OpenTheory, Coq, PVS, Lean
- `https://github.com/Deducteam/matita_lib_in_agda`:
  Matita/Arith $\longrightarrow$ Agda
- `https://github.com/Deducteam/hol2dk`:
  `https://github.com/Deducteam/coq-hol-light/`:
  HOL-Light $\longrightarrow$ Coq