universität freiburg

Which Assistant is the Right One?

Mathias Fleury

Tutorial Vienna 2023

Outline

The PA can Express what You Need

Warning

I am not a specialist of type theory!

The PA can Express what You Need

LCF PA

LCF Philosophy

Have a small kernel where everything goes through the kernel

```
module kernel =
  abstract type thm
  thm axiom1 ... thm axiomn
  fn combine_theorems: thm -> thm -> thm
end
```

Everything else is an oracle, included verified code! (Except for raw computation done by the compiler)

Name

There is an ongoing attempt to find a non-offensive (for US people) name
`https://github.com/coq/coq/wiki/Alternative-names` (for 2.5 years)

Trivia: french people all learn bit in school with the same meaning

Coq

1. Started 1989
2. Now based on the calculus of inductive construction
3. 2013 ACM Software System Award

Two major nearly incompatible versions, Coq and Coq/ssreflect

Proper separation between different axiom systems (classical, HoTT, ...)

Coq

1. Started 1989
2. Now based on the calculus of inductive construction
3. 2013 ACM Software System Award

Two major nearly incompatible versions, Coq and Coq/ssreflect

Proper separation between different axiom systems (classical, HoTT, ...)

Thomas Hales on his blog

> *Coq is built of modular components on a foundation of dependent type theory. This system has grown one PhD thesis at a time.*

Coq: Past Bugs

- Fixpoint with more than 256 cases
- ... and various other, partially due to interference of features

Lean

- Started 2013, now under the umbrella of the Lean FRO

Lean

- Started 2013, now under the umbrella of the Lean FRO
- strong focus on performance (at the cost of the kernel size)

Lean

- Started 2013, now under the umbrella of the Lean FRO
- strong focus on performance (at the cost of the kernel size)
- community expected automation to appear early (since 2015), never happened

Lean

- Started 2013, now under the umbrella of the Lean FRO
- strong focus on performance (at the cost of the kernel size)
- community expected automation to appear early (since 2015), never happened

- every major update breaks everything (we are at Lean 4!)

Lean

- Started 2013, now under the umbrella of the Lean FRO
- strong focus on performance (at the cost of the kernel size)
- community expected automation to appear early (since 2015), never happened

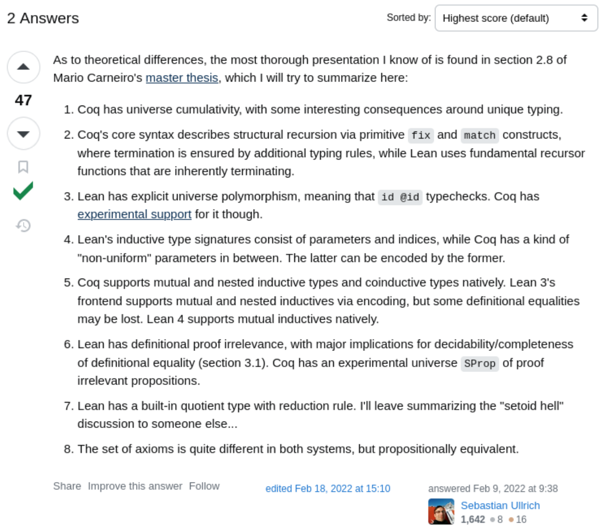
- every major update breaks everything (we are at Lean 4!)
- Main development: mathlib

Lean

- Started 2013, now under the umbrella of the Lean FRO
- strong focus on performance (at the cost of the kernel size)
- community expected automation to appear early (since 2015), never happened

- every major update breaks everything (we are at Lean 4!)
- Main development: mathlib
- $\sim$ Coq done properly

Lean vs Coq



2 Answers

Sorted by: Highest score (default)

47

As to theoretical differences, the most thorough presentation I know of is found in section 2.8 of Mario Carneiro's master thesis, which I will try to summarize here:

1. Coq has universe cumulativity, with some interesting consequences around unique typing.
2. Coq's core syntax describes structural recursion via primitive `fix` and `match` constructs, where termination is ensured by additional typing rules, while Lean uses fundamental recursor functions that are inherently terminating.
3. Lean has explicit universe polymorphism, meaning that `id @id` typechecks. Coq has experimental support for it though.
4. Lean's inductive type signatures consist of parameters and indices, while Coq has a kind of "non-uniform" parameters in between. The latter can be encoded by the former.
5. Coq supports mutual and nested inductive types and coinductive types natively. Lean 3's frontend supports mutual and nested inductives via encoding, but some definitional equalities may be lost. Lean 4 supports mutual inductives natively.
6. Lean has definitional proof irrelevance, with major implications for decidability/completeness of definitional equality (section 3.1). Coq has an experimental universe `SProp` of proof irrelevant propositions.
7. Lean has a built-in quotient type with reduction rule. I'll leave summarizing the "setoid hell" discussion to someone else...
8. The set of axioms is quite different in both systems, but propositionally equivalent.

Share Improve this answer Follow

edited Feb 18, 2022 at 15:10

answered Feb 9, 2022 at 9:38

Sebastian Ullrich

1,642 ● 8 ● 16

Isabelle

Named after Gérard Huet's daughter

Isabelle

- Started in 1986
- Based on intuitionistic higher-order logic with rank-1 polymorphism (“Pure”)
- No real motivation to reduce the kernel size
- A readable way of writing proofs

Variants: Isabelle/HOL (most developed), Isabelle/ZF, Isabelle/FOL, ...

Isabelle: Past Bugs

- Printing the exception took too long (did not impact `isabelle build`)
- Accidental missing check in dependency between types and constants

```
consts a
type 'c = {a, true}
define a = if card('c) = 2 then false else true
```

Isabelle and Friends

HOL4 (famous for CakeML), HOL Light (many math theorems), ... same logic than Isabelle/HOL. Without readable proof language.

Tom Hales on HOL Light:

> *HOL Light has an exquisite minimal design. It has the smallest kernel of any system. John Harrison is the sole permanent occupant, but he is very welcoming to guests.*

HOL vs Coq/Lean

HOL	Coq/Lean
Bool=Prop	Bool for computable Prop for non-computable

```
if Rieman_hypothesis then 1 else 0
```

HOL vs Coq/Lean

HOL	Coq/Lean
Bool=Prop	Bool for computable Prop for non-computable
`if Rieman_hypothesis then 1 else 0`	
Total functions	

HOL vs Coq/Lean

HOL	Coq/Lean
Bool=Prop	Bool for computable Prop for non-computable
`if Rieman_hypothesis then 1 else 0`	
Total functions	
	Easier subtyping

HOL vs Coq/Lean

HOL	Coq/Lean
Bool=Prop	Bool for computable
	Prop for non-computable
`if Rieman_hypothesis then 1 else 0`	
Total functions	
	Easier subtyping
groups all based on + for whole type	
properties	dependent types

HOL vs Coq/Lean

HOL	Coq/Lean

Bool=Prop
`if Rieman_hypothesis then 1 else 0`
	Bool for computable
	Prop for non-computable

Total functions	
groups all based on + for whole type	Easier subtyping
properties	dependent types
easier to automate	

HOL vs Coq/Lean

HOL	Coq/Lean
Bool=Prop	Bool for computable Prop for non-computable
`if Rieman_hypothesis then 1 else 0`	
Total functions	
groups all based on + for whole type	Easier subtyping
properties	dependent types
easier to automate (Sledgehammer, SMT)	

HOL vs Coq/Lean

HOL	Coq/Lean

Bool=Prop
`if Rieman_hypothesis then 1 else 0`
	Bool for computable
	Prop for non-computable

Total functions	
groups all based on + for whole type	Easier subtyping
properties	dependent types
easier to automate (Sledgehammer, SMT)	
simplifier	frowned upon as large proof terms

HOL vs Coq/Lean

HOL	Coq/Lean

Bool=Prop
 	
`if Rieman_hypothesis then 1 else 0`
	Bool for computable
	Prop for non-computable

Total functions	
groups all based on + for whole type	Easier subtyping
properties	dependent types
easier to automate (Sledgehammer, SMT)	
simplifier	frowned upon as large proof terms

The PA can Express what You Need

Beyond LCF

ACL2

- very US-centric, pragmatic approach: verified tool can be used
- started in 90
- untipped based logic

Others include PVS, Metamath, Mizra

Proof Exchange

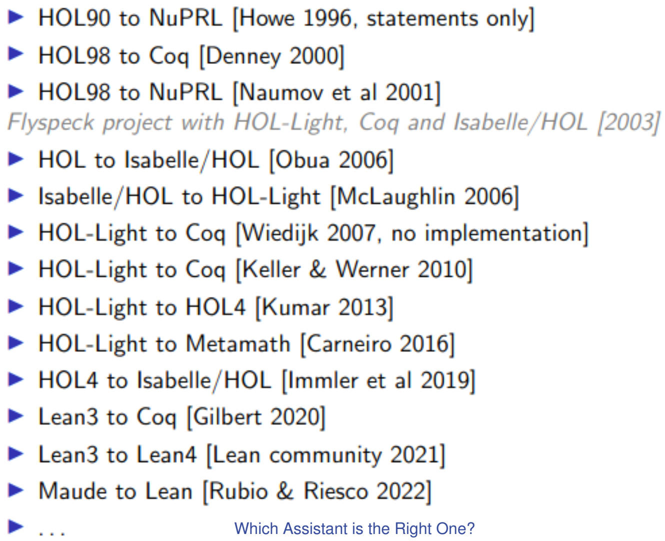
Exchanging Proof is Possible?

What you want:

- use the existing standard library to get native proofs
- translation of the theorems
- do not introduce extra axioms

Reality: it just never works beside trivial application (no, transferring HOL is not enough)

From Blanqui:



- HOL90 to NuPRL [Howe 1996, statements only]
- HOL98 to Coq [Denney 2000]
- HOL98 to NuPRL [Naumov et al 2001]

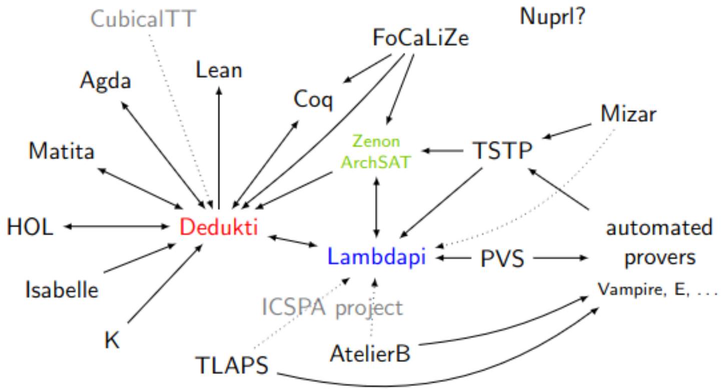
Flyspeck project with HOL-Light, Coq and Isabelle/HOL [2003]

- HOL to Isabelle/HOL [Obua 2006]
- Isabelle/HOL to HOL-Light [McLaughlin 2006]
- HOL-Light to Coq [Wiedijk 2007, no implementation]
- HOL-Light to Coq [Keller & Werner 2010]
- HOL-Light to HOL4 [Kumar 2013]
- HOL-Light to Metamath [Carneiro 2016]
- HOL4 to Isabelle/HOL [Immler et al 2019]
- Lean3 to Coq [Gilbert 2020]
- Lean3 to Lean4 [Lean community 2021]
- Maude to Lean [Rubio & Riesco 2022]
- ...

Best at PR

But wait, there is a EuroproofNet about exchanging proofs!

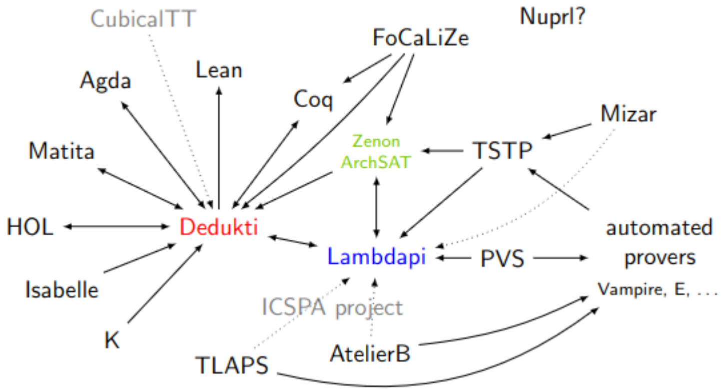
Best at PR



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

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

Best at PR



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

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

Can you get Help from Colleagues?

Communities

Most ITP have

a Zulip-chat exists for all ITP, for Isabelle: mostly for beginners

a mailing-list where development happens

Does It have the Right Libraries?

Coq

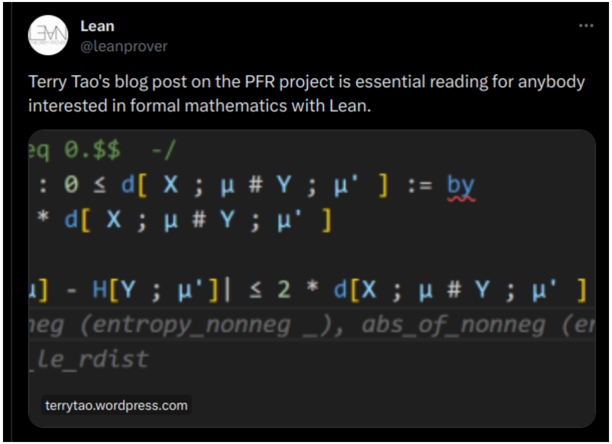
- 4-color theorems
- math-comp
- CompCert: formally verified compiler (similar to `gcc -O1`)
- imperative code extraction

Isabelle

- Archive of Formal Proofs contains most developments (+ distribution with HOL/Analysis)
- Everything older than 5 years is hopelessly outdated
- seL4: verified microkernel
- java bytecode semantics
- imperative code extraction and refinement library

Lean

- mathlib (Kevin Buzzard)


Lean
@leanprover

Terry Tao's blog post on the PFR project is essential reading for anybody interested in formal mathematics with Lean.

```
eq 0.$$  -/
: 0 ≤ d[ X ; μ # Y ; μ' ] := by
* d[ X ; μ # Y ; μ' ]

μ] - H[Y ; μ']| ≤ 2 * d[X ; μ # Y ; μ' ]
neg (entropy_nonneg _), abs_of_nonneg (en
_le_rdist
```

terrytao.wordpress.com