CoqHoTT

A New Proof-Assistant that Revisits the Theoretical Foundations of Coq using Homotopy Type Theory

Nicolas Tabareau
The CoqHoTT project

Design and implement a brand-new proof assistant by revisiting the theoretical foundations of Coq.

Type Theory  ➔  Homotopy Type Theory
Coq: a success but ...

Based on the correspondence:

<table>
<thead>
<tr>
<th>Formula</th>
<th>⇔</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proof</td>
<td>⇔</td>
<td>Program</td>
</tr>
</tbody>
</table>

**Type Theory** has been developed, providing a common language for mathematics and computer science \(\Rightarrow\) \textbf{Coq}

- Program certification
- Theorem Proving

**CompCert Compiler**

**The 4 Colour Theorem**

CoqHoTT, a brand-new proof assistant based on Homotopy Type Theory
Coq: a success but ...

Program certification:

✓ First ever certified C-Compiler

Theorem proving:

✓ High impact in computer science as well as in mathematics

CompCert Compiler

The 4 Colour Theorem
... not the last word

Many weaknesses cannot be solved without changing the theoretical foundations of Coq:

- common operators/principles cannot be “constructed”
  (eg., general fixpoints or the law of excluded middle)
- the notion of equality is too weak
... not the last word

Many weaknesses cannot be solved without changing the theoretical foundations of Coq:

- common operators/principles cannot be “constructed” (eg., general fixpoints or the law of excluded middle)
- the notion of equality is too weak

Example: bounded integers

\[(n; \text{bounded}_n) \neq (n; \text{bounded}_n)\]
Only one way out

make use of axioms

- consistency issues
  - valid axioms can be wrong altogether
- breaks the extraction mechanism
  - axioms have no computational meaning
- limits possibility of automation
  - can not use reflection in the proof
The CoqHoTT project

Design and implement a brand-new proof assistant by revisiting the theoretical foundations of Coq.

Type Theory  ➔  Homotopy Type Theory
The CoqHoTT project

Design and implement a brand-new proof assistant by revisiting the theoretical foundations of Coq.

Type Theory $\rightarrow$ Homotopy Type Theory

Uniform equality (syntactic) $\rightarrow$ Relativized equality (semantic)
The Big Challenge

Revisit the theory behind Coq using HoTT and provide a brand-new proof assistant with:

- equality as a first-class citizen
- axiom-free extensions of the logic
The Big Challenge

Revisit the theory behind Coq using HoTT and provide a brand-new proof assistant with:
- equality as a first-class citizen
- axiom-free extensions of the logic

Apart from relaxing the drawbacks of using axioms, it will simplify a lot the development of new proofs for users.
State of the art. Homotopy Type Theory

Idea:

equality $\iff$ homotopy

- the univalence principle [Streicher, Voevodsky], which allows to derive equality principles used in mathematics:

Definition univalence := $\forall T U : \text{Type}, T \sim U \rightarrow T = U.$
State of the art.
Homotopy Type Theory

Idea:

\[
\text{equality } \iff \text{homotopy}
\]

+ the **univalence principle** [Streicher, Voevodsky], which allows to derive equality principles used in mathematics:

**Definition** univalence := \( \forall \ T \ U : \text{Type}, \ T \sim U \rightarrow T = U. \)

**Issue:**

Univalence is still stated as an axiom.
State of the art.
Logic Extension in Mathematical Logic

Idea:
Translating formulas of rich logic into formulas of a simpler logic by using complex proof transformations
State of the art.
Logic Extension in Mathematical Logic

Idea:
Translating formulas of rich logic into formulas of a simpler logic by using complex proof transformations

Issue:
Type Theory ≠ Mathematical Logic
CoqHoTT Challenges

C1  Type Theory with a **built-in notion of univalence**

C2  Implement CoqHoTT **without overhead**

C3  Define and implement **Higher Inductive Types**

C4  Extend Type Theory **without axioms**
CoqHoTT Challenges

C1 Type Theory with a **built-in notion of univalence**

C4 Extend Type Theory **without axioms**
Type Theory with Univalence

- **Type T is a space**

- **Programs** $a : T$ are points

- **Proofs of equality** $p : a = b$ are paths
Type Theory with Univalence

Path operations:
- \text{id} : a =_T a
- p^{-1} : b =_T a
- q \circ p : a =_T c

Homotopies:
- \text{left-id} : \text{id} \circ p =_{a=b} p
- \text{right-id} : p \circ \text{id} =_{a=b} p
- \text{assoc} : r \circ (q \circ p) =_{a=d} (r \circ q) \circ p

Proofs of equality:
- \text{p} : a = b are paths

Programs:
a:T are points

type T is a space
Type Theory with Univalence and Higher Homotopies

The main novelty of this approach is to realize that homotopies between homotopies cannot be omitted, and this up to infinite dimension.
Type Theory with Univalence and Higher Homotopies

This compilation phase uses higher algebraic structure (e.g., $\infty$-groupoids, cubical sets [Coquand et al.])

CoqHoTT, a brand-new proof assistant based on Homotopy Type Theory
Extend Type Theory without Axioms

We will use the fact that:

Homotopy Type Theory = Higher Mathematical Logic
Extend Type Theory without Axioms Using Compilation Phases

We will use the fact that:

**Homotopy Type Theory = Higher Mathematical Logic**

Consider logical transformations (forcing, sheafification) of mathematical logic as compilation phases in HoTT:

- increase the power of the logic **without axioms**
- change the logic **at compile time**, according to a trade-off between efficiency and logical expressivity
Extend Type Theory without Axioms Using Compilation Phases

We will use the fact that:

Homotopy Type Theory = Higher Topos Theory [Lurie]

Consider logical transformations (forcing, sheafification) of mathematical logic as compilation phases in HoTT:

- increase the power of the logic **without axioms**
- change the logic **at compile time**, according to a trade-off between efficiency and logical expressivity
Extend Type Theory without axioms using a compilation phase

CoqHoTT + Classical Logic
added principle: Excluded Middle

CoqHoTT + Axiom of Choice
added principle: Dependent AC

CoqHoTT + General Induction
added principles: General Inductive Types Löb Rule

CoqHoTT + Kripke Semantics
added principle: Modal Logic

CoqHoTT
}

Sheaf Translation w/ Dense Topology
Forcing Translation w/ Natural Numbers

Sheaf Translation w/ Dense Topology
Forcing Translation w/ worlds

Sheaf Translation w/ Dense Topology

CoqHoTT, a brand-new proof assistant based on Homotopy Type Theory
Methodology.
Distinct compilation phases

Compile complex type theories into simpler ones.

- inherit consistency of Coq
- split the complexity of type checking
Feasability and Impacts

The CoqHoTT project is very ambitious—in particular Challenges 1 & 4.

- Background and Connections to the Coq community
- Connections to the emerging HoTT community
Feasability and Impacts

The CoqHoTT project is very ambitious—in particular Challenges 1 & 4.

- Background and Connections to the Coq community
- Connections to the emerging HoTT community

Our expectation is that CoqHoTT will become a popular proof assistant among:

- software engineers: faster and simpler developments
- computer scientists: integration of type isomorphisms
- mathematicians: new equality and logical expressivity
The CoqHoTT team ... 

The PI (75 % of his time) will take charge of the overall scientific direction.

Matthieu Sozeau (20%) is one of the main developers of Coq and its expertise on the subject is crucial to the CoqHoTT project.

Non-Permanent Staff (ERC):
3 PhD, 4 Post-Docs, 1 software engineer

Non-Permanent Staff (Host Institution, Inria):
1 PhD, 1 Post-Doc, 1 support engineer

and hopefully more!
... inside a very active community
CoqHoTT in a nutshell

New proof-assistant based on HoTT
Equality as a first-class citizen
Axiom-free extensions of the logic

Defined using distinct compilations phases

A world-class group on a new generation of proof assistants