# Formalised meta-theory for a verified type-theoretic kernel

GdT Malinca – October 22nd 2025

Meven Lennon-Bertrand


ROCQ

Agda


LEAN

Isabelle
HOL





Great successes!





**Great successes!**
**But what makes them usable?**





Pattern-matching

(Strong) records

(Computational) univalence

Termination checking

(Co)Inductive types

Universes

Proof irrelevance

Pattern-matching

(Strong) records

(Computational) univalence

Termination checking

(Co)Inductive types

Universes

Proof irrelevance

Gradual typing

Modalities

Observational equality

Subtyping

Pattern-matching

(Strong) records

(Computational) univalence

Termination checking

(Co)Inductive types

Universes

Proof irrelevance

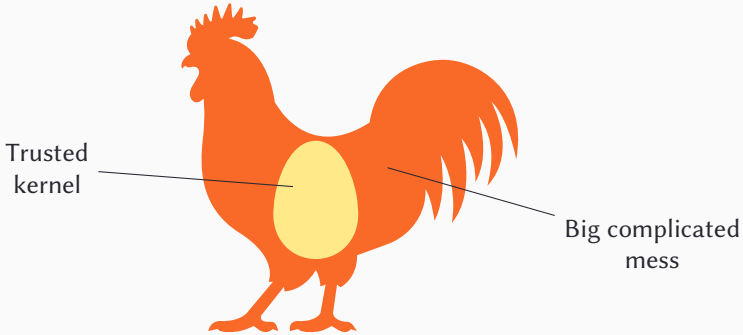
Gradual typing

Modalities

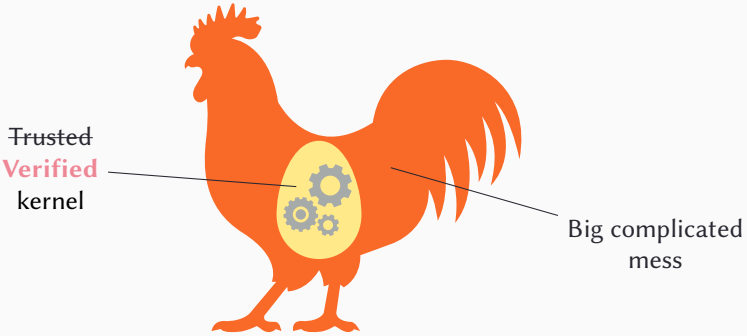
Observational equality

Subtyping

**Real proof assistants are complicated**




The de Bruijn architecture



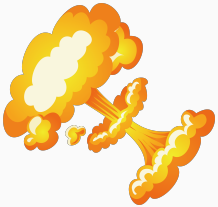

The de Bruijn architecture: a perfect target for **verification**

Rocq's kernel is only a few kLoC of pure functional code. *Surely* it can't be that difficult?

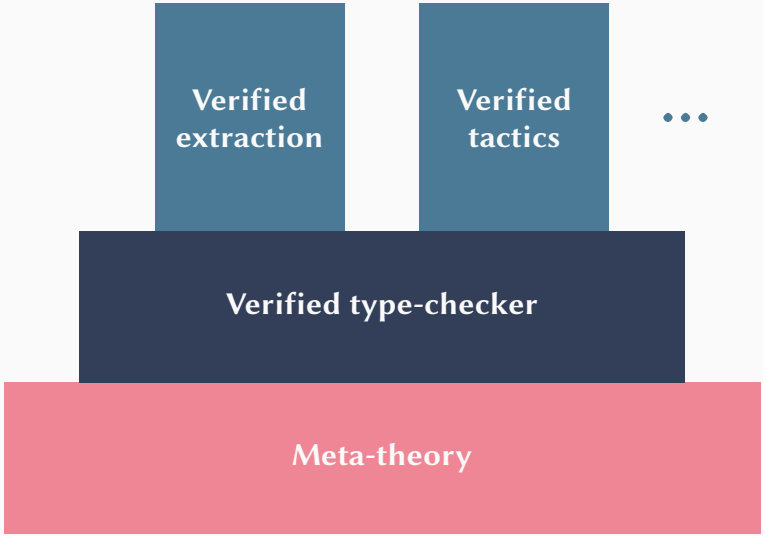
Rocq's kernel is only a few kLoC of pure functional code. *Surely* it can't be that difficult?

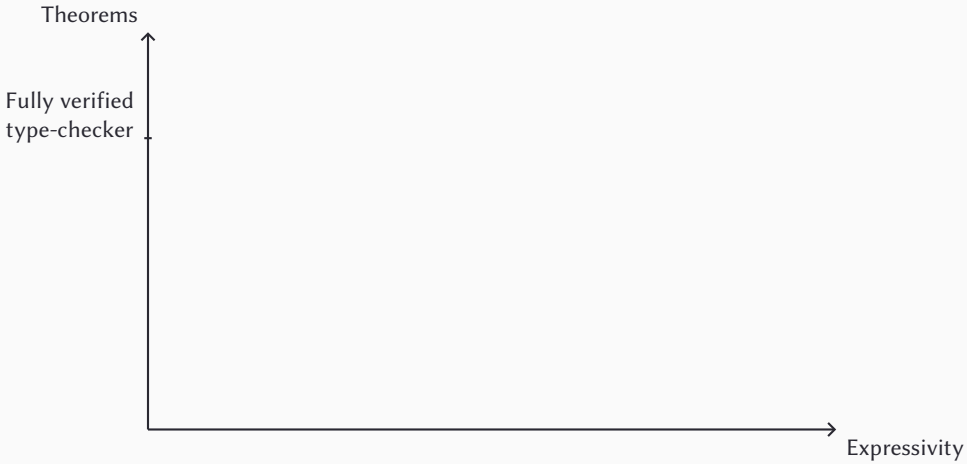
Dependent type theory + Invariants

=




Verified type-checker
Meta-theory


Verified extraction
Verified tactics
...
Verified type-checker
Meta-theory


Theorems
Fully verified
type-checker
Expressivity

# Expressivity?

*Coq in Coq* (Barras et al. 1997): verified type-checker for the CoC, in Coq.

# Expressivity?

*Coq in Coq* (Barras et al. 1997): verified type-checker for the CoC, in Coq.

CoC is proof-theoretically stronger than Agda, close to Rocq. Time to change subject?

# Expressivity?

*Coq in Coq* (Barras et al. 1997): verified type-checker for the CoC, in Coq.

CoC is proof-theoretically stronger than Agda, close to Rocq. Time to change subject?

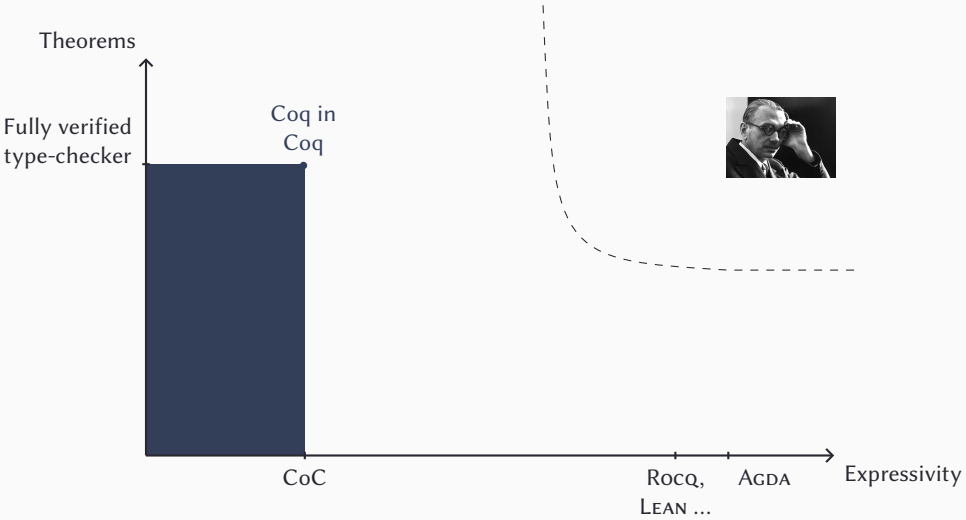
**Proof-theoretic strength does not measure expressivity**

Rocq in Rocq?

Rocq in Rocq?

~~Rocq in Rocq?~~

The meta-theory of an object theory $\mathcal{T}$ in a **(slightly) stronger ambient theory** $\mathcal{T}'$.
Or: **almost all** the meta-theory of a theory $\mathcal{T}$ in $\mathcal{T}$ itself.


Theorems
Fully verified type-checker
Coq in Coq
CoC
Rocq, Lean ...
Agda
Expressivity

A long shopping list:

- substitution for term variables
- substitution for universe variables
- weakening
- strengthening
- injectivity of type constructors
- confluence
- standardisation
- safety
- progress
- preservation/subject reduction
- uniqueness of types
- canonicity
- logical consistency
- normalisation
- decidability

A long shopping list:

- substitution for term variables
- substitution for universe variables
- weakening
- strengthening
- injectivity of type constructors
- confluence
- standardisation
- safety
- progress
- preservation/subject reduction
- uniqueness of types
- canonicity
- logical consistency
- normalisation
- decidability

**What properties are needed to verify what?**

**How do we prove these properties?**

# Dependent type theory

# What's type theory anyway?


Many answers

# What's type theory anyway?


Many answers, but let's say:

- a system of rules/a language
- intended to capture
  - valid logical/mathematical inferences
  - valid properties of programs

# What's type theory anyway?


Many answers, but let's say:

- a system of rules/a language
- intended to capture
  - valid logical/mathematical inferences
  - valid properties of programs

} Curry-Howard: **These are essentially the same!**

# A system of rules

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$$

$$\frac{\Gamma, x{:}\,A \vdash t : B}{\Gamma \vdash \lambda\, x{:}\,A.t : A \to B}$$

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B} \qquad \frac{\Gamma, x{:}\,A \vdash t : B}{\Gamma \vdash \lambda\, x{:}\,A.t : A \to B}$$

Formally read as:

- inductively defined predicates on tree-like structures
- directly defining an algebraic structure

Dependent types:

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathrm{Vect}\, A\, n}$$

Dependent types:

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathrm{Vect}\, A\, n}$$

$$\frac{\Gamma \vdash t : \Pi\, x{:}\, A.B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B[x := u]} \qquad \frac{\Gamma, x{:}\, A \vdash t : B}{\Gamma \vdash \lambda\, x{:}\, A.t : \Pi\, x{:}\, A.B}$$

Dependent types:

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathrm{Vect}\, A\, n}$$

$$\frac{\Gamma \vdash t : \Pi\, x{:}\, A.B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B[x := u]} \qquad \frac{\Gamma, x{:}\, A \vdash t : B}{\Gamma \vdash \lambda\, x{:}\, A.t : \Pi\, x{:}\, A.B}$$

The real beast is **conversion**/**definitional equality**:

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash A \cong B}{\Gamma \vdash t : B}$$

Dependent types:

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathrm{Vect}\, A\, n}$$

$$\frac{\Gamma \vdash t : \Pi\, x{:}\, A.B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B[x := u]} \qquad \frac{\Gamma, x{:}\, A \vdash t : B}{\Gamma \vdash \lambda\, x{:}\, A.t : \Pi\, x{:}\, A.B}$$

The real beast is **conversion**/**definitional equality**:

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash A \cong B}{\Gamma \vdash t : B}$$

$$\frac{\Gamma \vdash [1; 2; 7] : \mathrm{Vect}\, \mathbb{N}\, (1 + 1 + 1) \qquad \Gamma \vdash \mathrm{Vect}\, \mathbb{N}\, (1 + 1 + 1) \cong \mathrm{Vect}\, \mathbb{N}\, 3}{\Gamma \vdash [1; 2; 7] : \mathrm{Vect}\, \mathbb{N}\, 3}$$

# Definitional equality

The least **equivalence relation**

$$\textsc{Refl}\ \frac{\Gamma \vdash t : A}{\Gamma \vdash t \cong t : A} \qquad \textsc{Sym}\ \frac{\Gamma \vdash t \cong u : A}{\Gamma \vdash u \cong t : A} \qquad \textsc{Trans}\ \frac{\Gamma \vdash t \cong u : A \qquad \Gamma \vdash u \cong v : A}{\Gamma \vdash t \cong v : A}$$

# Definitional equality

The least **equivalence relation** which is **congruent**,

$$\textsc{Refl}\ \frac{\Gamma \vdash t : A}{\Gamma \vdash t \cong t : A} \qquad \textsc{Sym}\ \frac{\Gamma \vdash t \cong u : A}{\Gamma \vdash u \cong t : A} \qquad \textsc{Trans}\ \frac{\Gamma \vdash t \cong u : A \qquad \Gamma \vdash u \cong v : A}{\Gamma \vdash t \cong v : A}$$

$$\textsc{AppCong}\ \frac{\Gamma \vdash t \cong t' : \Pi\, x{:}\, A.B \qquad \Gamma \vdash u \cong u' : A}{\Gamma \vdash t\ u \cong t'\ u' : B[x := u]} \qquad \ldots$$

# Definitional equality

The least **equivalence relation** which is **congruent**, contains **computation**,

$$\textsc{Refl}\ \frac{\Gamma \vdash t : A}{\Gamma \vdash t \cong t : A} \qquad \textsc{Sym}\ \frac{\Gamma \vdash t \cong u : A}{\Gamma \vdash u \cong t : A} \qquad \textsc{Trans}\ \frac{\Gamma \vdash t \cong u : A \qquad \Gamma \vdash u \cong v : A}{\Gamma \vdash t \cong v : A}$$

$$\textsc{AppCong}\ \frac{\Gamma \vdash t \cong t' : \Pi\, x{:}\, A.B \qquad \Gamma \vdash u \cong u' : A}{\Gamma \vdash t\ u \cong t'\ u' : B[x := u]} \qquad \ldots$$

$$\beta\textsc{Fun}\ \frac{\Gamma \vdash A \qquad \Gamma, x{:}\, A \vdash B \qquad \Gamma, x{:}\, A \vdash t : B \qquad \Gamma \vdash u : A}{\Gamma \vdash (\lambda\, x{:}\, A.t)\, u \cong t[x := u] : B[x := u]}$$

# Definitional equality

The least **equivalence relation** which is **congruent**, contains **computation**, and **some more**.

$$\textsc{Refl}\ \frac{\Gamma \vdash t : A}{\Gamma \vdash t \cong t : A} \qquad \textsc{Sym}\ \frac{\Gamma \vdash t \cong u : A}{\Gamma \vdash u \cong t : A} \qquad \textsc{Trans}\ \frac{\Gamma \vdash t \cong u : A \qquad \Gamma \vdash u \cong v : A}{\Gamma \vdash t \cong v : A}$$

$$\textsc{AppCong}\ \frac{\Gamma \vdash t \cong t' : \Pi\, x{:}\, A.B \qquad \Gamma \vdash u \cong u' : A}{\Gamma \vdash t\ u \cong t'\ u' : B[x := u]} \qquad \ldots$$

$$\beta\textsc{Fun}\ \frac{\Gamma \vdash A \qquad \Gamma, x{:}\, A \vdash B \qquad \Gamma, x{:}\, A \vdash t : B \qquad \Gamma \vdash u : A}{\Gamma \vdash (\lambda\, x{:}\, A.t)\, u \cong t[x := u] : B[x := u]} \qquad \eta\textsc{Fun}\ \frac{\Gamma \vdash f : \Pi\, x{:}\, A.B}{\Gamma \vdash f \cong \lambda x{:}\, A.f\ x : \Pi\, x{:}\, A.B} \qquad \ldots$$

# Definitional equality

The least **equivalence relation** which is **congruent**, contains **computation**, and **some more**.

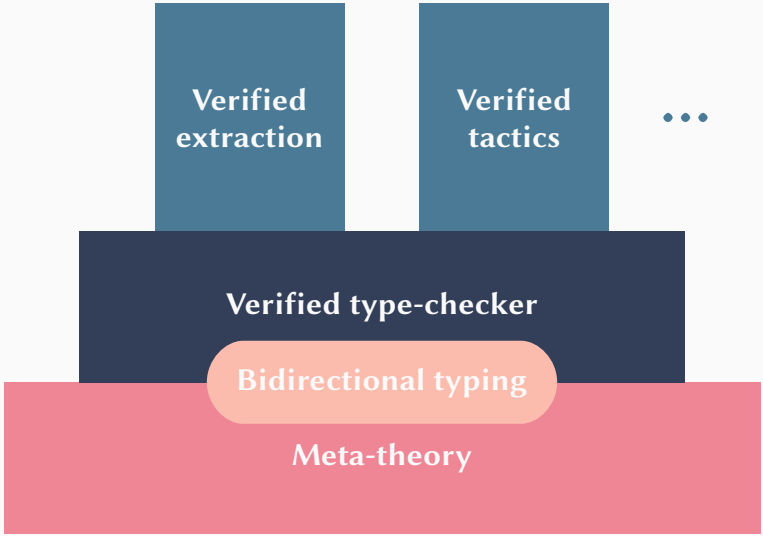
$$\textsc{Refl}\ \frac{\Gamma \vdash t : A}{\Gamma \vdash t \cong t : A} \qquad \textsc{Sym}\ \frac{\Gamma \vdash t \cong u : A}{\Gamma \vdash u \cong t : A} \qquad \textsc{Trans}\ \frac{\Gamma \vdash t \cong u : A \qquad \Gamma \vdash u \cong v : A}{\Gamma \vdash t \cong v : A}$$

$$\textsc{AppCong}\ \frac{\Gamma \vdash t \cong t' : \Pi\, x{:}\, A.B \qquad \Gamma \vdash u \cong u' : A}{\Gamma \vdash t\ u \cong t'\ u' : B[x := u]} \qquad \ldots$$

$$\beta\textsc{Fun}\ \frac{\Gamma \vdash A \qquad \Gamma, x{:}\, A \vdash B \qquad \Gamma, x{:}\, A \vdash t : B \qquad \Gamma \vdash u : A}{\Gamma \vdash (\lambda\, x{:}\, A.t)\, u \cong t[x := u] : B[x := u]} \qquad \eta\textsc{Fun}\ \frac{\Gamma \vdash f : \Pi\, x{:}\, A.B}{\Gamma \vdash f \cong \lambda x{:}\, A.f\ x : \Pi\, x{:}\, A.B} \qquad \ldots$$

If we orient β-rules, we get **reduction** $\to$.

# Bidirectional typing


Verified extraction
Verified tactics
...
Verified type-checker
Bidirectional typing
Meta-theory

# Information Must Flow

**Inference and checking**

$\Gamma \vdash t : A$ separates into

inference: $\Gamma \vdash t \triangleright A$ checking: $\Gamma \vdash t \triangleleft A$

Similar “meaning”, different modes: input/subject/output.

# Structure!

$$\frac{\vdash \Gamma \qquad (x : A) \in \Gamma}{\Gamma \vdash x : A}$$

$$\frac{\Gamma \vdash t : \Pi\, x : A.B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B[x := u]}$$

- Globally enforce well-formation

# Structure!

$$\frac{\vdash \Gamma \qquad (x : A) \in \Gamma}{\Gamma \vdash x : A}$$

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x \triangleright A}$$

$$\frac{\Gamma \vdash t : \Pi\, x : A.B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B[x := u]}$$

$$\frac{\Gamma \vdash t \triangleright_{\mathrm{h}} \Pi\, x : A.B \qquad \Gamma \vdash u \triangleleft A}{\Gamma \vdash t\ u \triangleright B[x := u]}$$

- ~~Globally enforce well-formation~~ → **Well-formation as an invariant**
- Clear **information flow**

# Structure!

$$\frac{\vdash \Gamma \qquad (x : A) \in \Gamma}{\Gamma \vdash x : A}$$

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x \triangleright A}$$

$$\frac{\Gamma \vdash t : \Pi\, x : A.B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B[x := u]}$$

$$\frac{\Gamma \vdash t \triangleright_{\mathrm{h}} \Pi\, x : A.B \qquad \Gamma \vdash u \triangleleft A}{\Gamma \vdash t\ u \triangleright B[x := u]}$$

$$\frac{\Gamma \vdash t : T \qquad \Gamma \vdash T \cong T'}{\Gamma \vdash t : T'}$$

- ~~Globally enforce well-formation~~ → Well-formation as an invariant
- Clear information flow
- Unconstrained conversion

# Structure!

$$\frac{\vdash \Gamma \qquad (x\colon A) \in \Gamma}{\Gamma \vdash x : A} \qquad \frac{(x\colon A) \in \Gamma}{\Gamma \vdash x \triangleright A}$$

$$\frac{\Gamma \vdash t : \Pi\, x\colon A.B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B[x := u]} \qquad \frac{\Gamma \vdash t \triangleright_{\mathrm{h}} \Pi\, x\colon A.B \qquad \Gamma \vdash u \triangleleft A}{\Gamma \vdash t\ u \triangleright B[x := u]}$$

$$\frac{\Gamma \vdash t : T \qquad \Gamma \vdash T \cong T'}{\Gamma \vdash t : T'} \qquad \frac{\Gamma \vdash t \triangleright T \qquad \Gamma \vdash T \cong T' \triangleleft}{\Gamma \vdash t \triangleleft T'} \qquad \frac{\Gamma \vdash t \triangleright T \qquad \Gamma \vdash T \to^{\star} T'}{\Gamma \vdash t \triangleright_{\mathrm{h}} T'}$$

- ~~Globally enforce well-formation~~ → Well-formation as an invariant
- Clear information flow
- ~~Unconstrained conversion~~ → Computation ($\to^{\star}$ or $\cong$) **guided by the mode**

# Information Must Flow

**Inference and checking**

$\Gamma \vdash t : A$ separates into

inference: $\Gamma \vdash t \triangleright A$ checking: $\Gamma \vdash t \triangleleft A$

Similar "meaning", different modes: input/subject/output.

**McBride:** ***A rule is a server for its conclusion and a client for its premises.***

- Modes guide invariant preservation
- In a conclusion, you assume inputs are well-formed, and ensure outputs are
- In a premise, you ensure inputs are well-formed, and assume outputs are

# Specifying bidirectional typing

**Positive soundness**

If the algorithm answers **yes** (*i.e.* $\Gamma \vdash t \triangleright T$) and its preconditions are met, then $\Gamma \vdash t : T$.

**Negative soundness**

If the algorithm answers **no** (*i.e.* $\Gamma \vdash t \triangleright \lightning$) and its preconditions are met, then $t$ is not typable.

**Totality**

If its preconditions are met, the algorithm always answers.

# Specifying bidirectional typing

**Positive soundness**

If the algorithm answers **yes** (*i.e.* $\Gamma \vdash t \triangleright T$) and its preconditions are met, then $\Gamma \vdash t : T$.

**Negative soundness**

If the algorithm answers **no** (*i.e.* $\Gamma \vdash t \triangleright \lightning$) and its preconditions are met, then $t$ is not typable.

**Totality**

If its preconditions are met, the algorithm always answers.

Positive soundness + Negative soundness + Totality $\Longrightarrow$

**Completeness** the answer is always yes on typable terms

**Decidability**

# MetaRocq

MetaCoq is developed by (left to right) Abhishek Anand, Danil Annenkov, Simon Boulier, Cyril Cohen, Yannick Forster, Jason Gross, Meven Lennon-Bertrand, Kenji Maillard, Gregory Malecha, Jakob Botsch Nielsen, Matthieu Sozeau, Nicolas Tabareau and Théo Winterhalter.

MetaCoq is developed by (left to right) Abhishek Anand, Danil Annenkov, Simon Boulier, Cyril Cohen, Yannick Forster, Jason Gross, Meven Lennon-Bertrand, Kenji Maillard, Gregory Malecha, Jakob Botsch Nielsen, Matthieu Sozeau, Nicolas Tabareau and Théo Winterhalter.

**The Predicative Calculus of Universe-Polymorphic Inductive Constructions (PCUIC)**

A dependent type theory with

- Very general (co-)inductive types
- Pattern-matching and fixed-points
- Complex universes + cumulativity

} What real users need!

**The Predicative Calculus of Universe-Polymorphic Inductive Constructions (PCUIC)**

A dependent type theory with

- Very general (co-)inductive types
- Pattern-matching and fixed-points
- Complex universes + cumulativity

} What real users need!

**Rocq, in Rocq**

- Formalised meta-theory of PCUIC

**The Predicative Calculus of Universe-Polymorphic Inductive Constructions (PCUIC)**

A dependent type theory with

- Very general (co-)inductive types
- Pattern-matching and fixed-points
- Complex universes + cumulativity

} What real users need!

**Rocq, in Rocq**

- Formalised meta-theory of PCUIC
- Normalisation axiom to implement a verified type-checker
- Verified extraction, meta-programming...

**The Predicative Calculus of Universe-Polymorphic Inductive Constructions (PCUIC)**

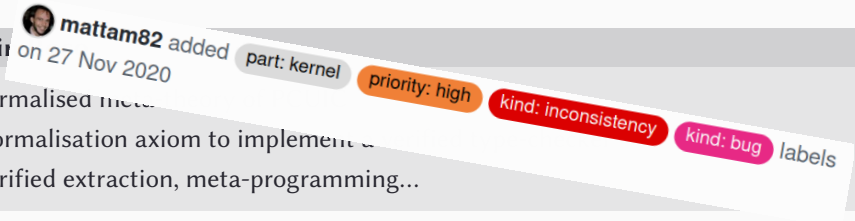
A dependent type theory with

- Very general (co-)inductive types
- Pattern-matching and fixed-points
- Complex universes + cumulativity

What real users need!

**Rocq, i**

- Formalised
- Normalisation axiom to implement
- Verified extraction, meta-programming...



**The Predicative Calculus of Universe-Polymorphic Inductive Constructions (PCUIC)**

A dependent type theory with

- Very general (co-)inductive types
- Pattern-matching and fixed-points
- Complex universes + cumulativity

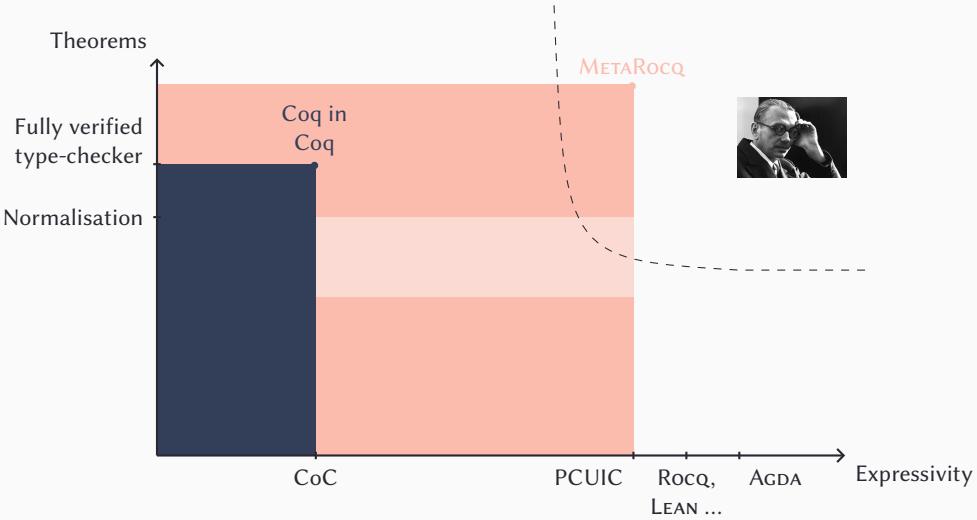
} What real users need!

**Rocq, in Rocq**

- Formalised meta-theory of PCUIC
- **Normalisation axiom** to implement a verified type-checker
- Verified extraction, meta-programming...

**Evaluation terminates**. Consequences:

- totality ♥
- logical consistency ⚠


Theorems
Fully verified type-checker
Normalisation
Coq in Coq
MetaRocq
CoC
PCUIC
Rocq, Lean ...
Agda
Expressivity

# The $1000 question: inversion

If you know $T \cong T'$, **what can you conclude**?

- $T \cong \Pi\, x{:}\, A.B \overset{?}{\Rightarrow} T \to^\star \Pi\, x{:}\, A'.B'$?
- $\Pi\, x{:}\, A.B \cong \Pi\, x{:}\, A'.B' \overset{?}{\Rightarrow} A \cong A' \wedge B \cong B'$?
- $\Pi\, x{:}\, A.B \cong \mathbb{N} \overset{?}{\Rightarrow} \bot$?
- $\mathrm{S}\, m \cong \mathrm{S}\, n \overset{?}{\Rightarrow} m \cong n$?
- $\mathrm{S}\, m \cong 0 \overset{?}{\Rightarrow} \bot$?
- ...

# The $1000 question: inversion

If you know $T \cong T'$, what can you conclude?

- $T \cong \Pi\, x\colon A.B \overset{?}{\Rightarrow} T \to^\star \Pi\, x\colon A'.B'$?
- $\Pi\, x\colon A.B \cong \Pi\, x\colon A'.B' \overset{?}{\Rightarrow} A \cong A' \wedge B \cong B'$?
- $\Pi\, x\colon A.B \cong \mathbb{N} \overset{?}{\Rightarrow} \bot$?
- $\mathrm{S}\, m \cong \mathrm{S}\, n \overset{?}{\Rightarrow} m \cong n$?
- $\mathrm{S}\, m \cong 0 \overset{?}{\Rightarrow} \bot$?
- ...

The derivation of $T \cong T'$ might be quite complicated...

# The $1000 question: inversion

If you know $T \cong T'$, what can you conclude?

- $T \cong \Pi\, x\colon A.B \overset{?}{\Rightarrow} T \to^\star \Pi\, x\colon A'.B'$?
- $\Pi\, x\colon A.B \cong \Pi\, x\colon A'.B' \overset{?}{\Rightarrow} A \cong A' \wedge B \cong B'$?
- $\Pi\, x\colon A.B \cong \mathbb{N} \overset{?}{\Rightarrow} \bot$?
- $\mathrm{S}\, m \cong \mathrm{S}\, n \overset{?}{\Rightarrow} m \cong n$?
- $\mathrm{S}\, m \cong 0 \overset{?}{\Rightarrow} \bot$?
- ...

The derivation of $T \cong T'$ might be quite complicated...

**Confluence**

(after Tait, Martin-Löf, Takahashi)

# The $1000 question: inversion

If you know $T \cong T'$, what can you conclude?

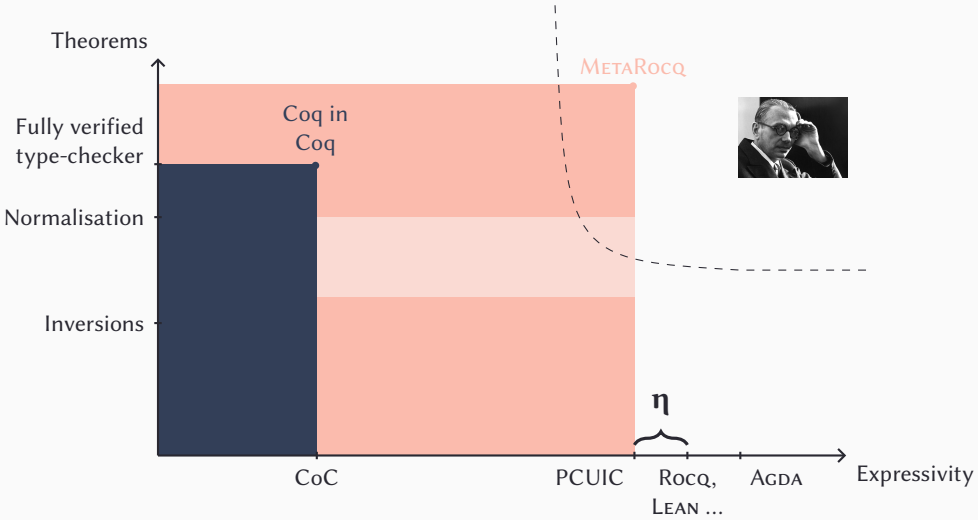
- $T \cong \Pi\, x\colon A.B \overset{?}{\Rightarrow} T \to^\star \Pi\, x\colon A'.B'$?
- $\Pi\, x\colon A.B \cong \Pi\, x\colon A'.B' \overset{?}{\Rightarrow} A \cong A' \wedge B \cong B'$?
- $\Pi\, x\colon A.B \cong \mathbb{N} \overset{?}{\Rightarrow} \bot$?
- $\mathrm{S}\, m \cong \mathrm{S}\, n \overset{?}{\Rightarrow} m \cong n$?
- $\mathrm{S}\, m \cong 0 \overset{?}{\Rightarrow} \bot$?
- ...

The derivation of $T \cong T'$ might be quite complicated...

**Confluence**[*]

(after Tait, Martin-Löf, Takahashi)

[*] **Only works** for a **rather poor** notion of conversion...


Theorems
Fully verified type-checker
Normalisation
Inversions
Coq in Coq
MetaRocq
η
CoC
PCUIC
Rocq, Lean ...
Agda
Expressivity

# Martin-Löf *à la* Coq

Arthur Adjedj

Kenji Maillard

Pierre-Marie Pédrot

Loïc Pujet

$$\eta\textsc{Fun}\ \frac{\Gamma \vdash f : \Pi\, x{:}\, A.B}{\Gamma \vdash f \cong \lambda x{:}\, A.f\ x : \Pi\, x{:}\, A.B}$$

# Extensionality rules

$$\eta\textsc{Fun}\ \frac{\Gamma \vdash f : \Pi\, x{:}\, A.B}{\Gamma \vdash f \cong \lambda x{:}\, A.f\ x : \Pi\, x{:}\, A.B}$$

$$\eta\textsc{Prop}\ \frac{\Gamma \vdash P : \mathrm{SProp} \qquad \Gamma \vdash p : P \qquad \Gamma \vdash p' : P}{\Gamma \vdash p \cong p' : P}$$

# Extensionality rules

$$\eta\textsc{Fun}\ \frac{\Gamma \vdash f : \Pi\, x{:}\, A.B}{\Gamma \vdash f \cong \lambda x{:}\, A.f\ x : \Pi\, x{:}\, A.B} \qquad \eta\textsc{Prop}\ \frac{\Gamma \vdash P : \mathrm{SProp} \qquad \Gamma \vdash p : P \qquad \Gamma \vdash p' : P}{\Gamma \vdash p \cong p' : P}$$

Might look easy (?), but:

- crucial in practice
- does not really work with confluence
- a very typed notion

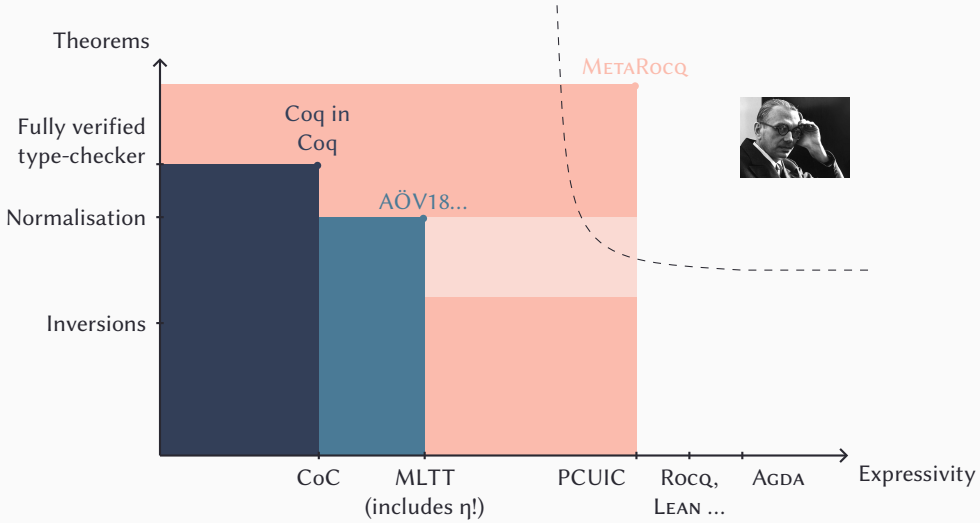
# Extensionality rules

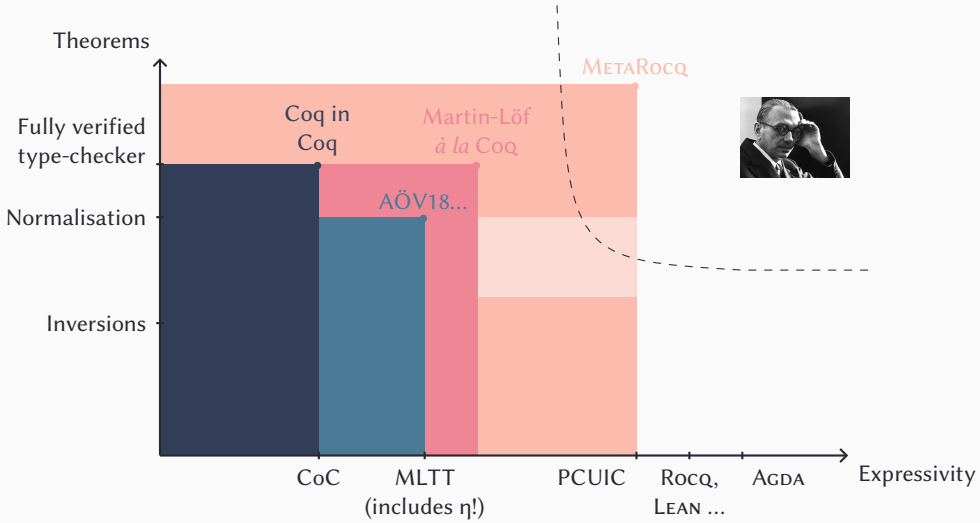
$$\eta\textsc{Fun}\ \frac{\Gamma \vdash f : \Pi\, x{:}\, A.B}{\Gamma \vdash f \cong \lambda x{:}\, A.f\ x : \Pi\, x{:}\, A.B} \qquad \eta\textsc{Prop}\ \frac{\Gamma \vdash P : \mathrm{SProp} \qquad \Gamma \vdash p : P \qquad \Gamma \vdash p' : P}{\Gamma \vdash p \cong p' : P}$$

Might look easy (?), but:

- crucial in practice
- does not really work with confluence
- a very typed notion

Rocq has a clever **untyped implementation**, but what's its **specification**?


Theorems
Fully verified type-checker
Normalisation
Inversions
Coq in Coq
AÖV18...
MetaRocq
CoC
MLTT (includes η!)
PCUIC
Rocq, Lean ...
Agda
Expressivity


Theorems
Fully verified type-checker
Normalisation
Inversions
MetaRocq
Coq in Coq
Martin-Löf à la Coq
AÖV18...
CoC
MLTT (includes η!)
PCUIC
Rocq, Lean ...
Agda
Expressivity

# Conversion is bidirectional too !

Conversion is bidirectional too ...

even if it does not compute types!

Conversion is bidirectional too ...

even if it does not compute types!

The important part is **invariant preservation**.

**Positive soundness**

Requires inversion for types.

**Negative soundness**

Requires inversion for types and terms.

In particular: **no normalisation**!
Conjecture: requires **low logical power**.

**Positive soundness**

Requires inversion for types.

**Negative soundness**

Requires inversion for types and terms.

In particular: **no normalisation**!
Conjecture: requires **low logical power**.

**Termination**

Requires inversion for types and normalisation.

# How to prove the properties?

| | Logical relation [AÖV17; Adj+24] | Gluing/Nf Model [Ste21; BKS23] | Rewriting/Confluence [Tak95]/MetaRocq | Domain model [CH18] |
|---|---|---|---|---|
| Weak ambiant theory | × | × | ✓ | ✓ |
| Normalisation | ✓ | ✓ | × | × |
| η laws | ✓ | ✓ | × | ✓ |
| Formalised | ✓ | ✓ / × | ✓ | × |
| | Most explored<br>Difficult to scale | Formalisation currently difficult | Insane scaling | Very unexplored |

# How to prove the properties?

| | Logical relation [AÖV17; Adj+24] | Gluing/Nf Model [Ste21; BKS23] | Rewriting/Confluence [Tak95]/MetaRocq | Domain model [CH18] |
|---|---|---|---|---|
| Weak ambiant theory | × | × | ✓ | ✓ |
| Normalisation | ✓ | | | × |
| η laws | ✓ | | | ✓ |
| Formalised | ✓ | | | × |
| | Most explored<br>Difficult to scale | Formalisation currently difficult | Insane scaling | Very unexplored |

**There is space for exploration!**

# How to prove the properties?

| | Logical relation [AÖV17; Adj+24] | Gluing/Nf Model [Ste21; BKS23] | Rewriting/Confluence [Tak95]/MetaRocq | Domain model [CH18] |
|---|---|---|---|---|
| Weak ambiant theory | × | × | ✓ | ✓ |
| Normalisation | ✓ | | | × |
| η laws | ✓ | | | ✓ |
| Formalised | ✓ | | | × |
| | Most explored<br>Difficult to scale | Formalisation currently difficult | Insane scaling | Very unexplored |

**There is space for exploration!**

... but I have peculiar requirements.

Thank you!

# Bibliography


[AÖV17] Andreas Abel, Joakim Öhman and Andrea Vezzosi. 'Decidability of Conversion for Type Theory in Type Theory'. In: *Proc. ACM Program. Lang.* POPL (Dec. 2017). DOI: 10.1145/3158111.

[WB18] Paweł Wieczorek and Dariusz Biernacki. 'A Coq Formalization of Normalization by Evaluation for Martin-Löf Type Theory'. In: *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs*. 2018. DOI: 10.1145/3167091.

[BW97] Bruno Barras and Benjamin Werner. 'Coq in Coq'. 1997.

[Adj+24] Arthur Adjedj et al. 'Martin-Löf à la Coq'. In: *Proceedings of the 13th ACM SIGPLAN International Conference on Certified Programs and Proofs*. 2024. DOI: 10.1145/3636501.3636951.

[Ste21] Jonathan Sterling. 'First Steps in Synthetic Tait Computability: The Objective Metatheory of Cubical Type Theory'. PhD thesis. Carnegie Mellon University, Nov. 2021. DOI: 10.5281/zenodo.6990769.

[BKS23] Rafaël Bocquet, Ambrus Kaposi and Christian Sattler. 'For the Metatheory of Type Theory, Internal Sconing Is Enough'. In: *8th International Conference on Formal Structures for Computation and Deduction, FSCD 2023*. 2023. DOI: 10.4230/LIPICS.FSCD.2023.18.

[Tak95] M. Takahashi. 'Parallel Reductions in λ-Calculus'. In: *Information and Computation* 1 (1995). DOI: 10.1006/inco.1995.1057.

[Soz+25] Matthieu Sozeau et al. 'Correct and Complete Type Checking and Certified Erasure for Coq, in Coq'. In: *Journal of the ACM* (Jan. 2025). DOI: 10.1145/3706056.

[CH18] Thierry Coquand and Simon Huber. 'An Adequacy Theorem for Dependent Type Theory'. In: *Theory of Computing Systems* 4 (July 2018). DOI: 10.1007/s00224-018-9879-9.