What it takes to verify a kernel
And a new way to prove it

HoTTest seminar – September 24, 2026

Meven Lennon-Bertrand

ENS Rennes


Agda


ROCQ


LEAN


mikan

Pattern-matching

Fancy records

(Computational) univalence

Termination checking

(Co)Inductive types

Universes

Proof irrelevance

Pattern-matching

Fancy records

(Computational) univalence

Termination checking

(Co)Inductive types

Universes

Proof irrelevance

Modalities

Observational equality

ν equations

Subtyping

Pattern-matching

Fancy records

(Computational) univalence

Termination checking

(Co)Inductive types

Universes

Proof irrelevance

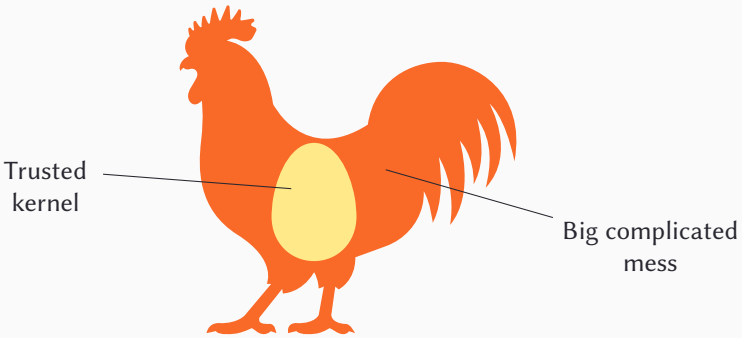
Modalities

Observational equality

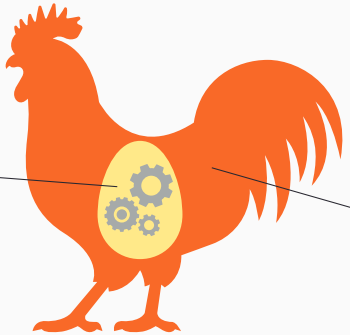
ν equations

Subtyping

Real proof assistants have curves


Trusted
kernel
Big complicated
mess

~~Trusted~~
Verified
kernel



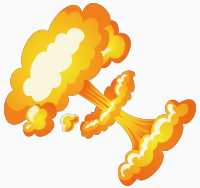
Big complicated
mess

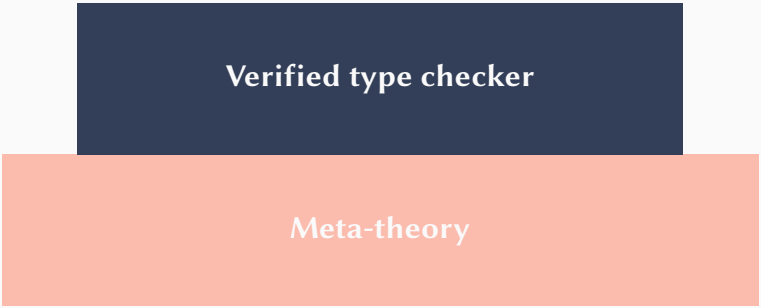
Kernels are only a few kLoC of pure functional code. So what's hard?

Kernels are only a few kLoC of pure functional code. So what's hard?

Invariants + Dependent type theory

=




Verified type checker
Meta-theory

Meta-theory of an **object** theory in an **ambient** theory

- 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

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

Which properties are needed to verify what?

How do we prove these properties?

How to verify a kernel

From *What Does It Take to Certify a Conversion Checker?* (FSCD '25)

Information must flow

Bidirectional typing

Separate $\Gamma \vdash t : A$ into

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

Similar “meaning”, different operational modes: input/subject/output (after McBride)

Information must flow

Bidirectional typing

Separate $\Gamma \vdash t : A$ into

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

Similar "meaning", different operational modes: input/subject/output (after McBride)

Constraints on dangerous rules (conversion, transitivity...)
Clear **information flow** guiding invariant preservation

Specifying a decision procedure

$$P : D \to \mathbb{P} \qquad \overset{?}{\Leftrightarrow} \qquad p : D \to \mathbb{B}$$

Specifying a decision procedure

$$P : D \to \mathbb{P} \qquad \overset{?}{\Leftrightarrow} \qquad p : D \to \mathbb{B}$$

0. decidability: $(p\ d = \text{true}) \vee (p\ d = \text{false})$

Specifying a decision procedure

$$P : D \to \mathbb{P} \qquad \overset{?}{\Leftrightarrow} \qquad p : D \to \mathbb{B}$$

0. decidability: $(p\ d = \text{true}) \vee (p\ d = \text{false})$

1. soundness: $p\ d = \text{true} \Rightarrow P\ d$

2. completeness: $P\ d \Rightarrow p\ d = \text{true}$

3. profit!

Specifying a decision procedure

$$P : D \to \mathbb{P} \quad \overset{?}{\Leftrightarrow} \quad p : D \to \mathbb{B}$$

0. decidability: $(p\ d = \text{true}) \vee (p\ d = \text{false})$

1. soundness: $p\ d = \text{true} \Rightarrow P\ d$

2. completeness: $P\ d \Rightarrow p\ d = \text{true}$

3. **profit?**

Specifying a decision procedure

$$P : D \to \mathbb{P} \quad \overset{?}{\Leftrightarrow} \quad p : D \rightharpoonup \mathbb{B}$$

0. **decidability**: $(p\ d = \text{true}) \vee (p\ d = \text{false})$
How do you know that the type checker terminates?

1. soundness: $p\ d = \text{true} \Rightarrow P\ d$

2. completeness: $P\ d \Rightarrow p\ d = \text{true}$

3. profit?

Specifying a decision procedure

$$P : D \to \mathbb{P} \quad \overset{?}{\Leftrightarrow} \quad p : D \rightharpoonup \mathbb{B}$$

0. **decidability**: $(p\ d = \text{true}) \vee (p\ d = \text{false})$
How do you know that the type checker terminates?

1. **soundness**: $p\ d = \text{true} \Rightarrow P\ d$
Look at the trace of the type checker

2. completeness: $P\ d \Rightarrow p\ d = \text{true}$

3. profit?

Specifying a decision procedure

$$P : D \to \mathbb{P} \quad \overset{?}{\Leftrightarrow} \quad p : D \rightharpoonup \mathbb{B}$$

0. **decidability**: $(p\ d = \text{true}) \vee (p\ d = \text{false})$
How do you know that the type checker terminates?

1. **soundness**: $p\ d = \text{true} \Rightarrow P\ d$
Look at the trace of the type checker

2. **completeness**: $P\ d \Rightarrow p\ d = \text{true}$
reflexivity $\simeq$ normalisation

3. profit?

$$P : D \to \mathbb{P} \qquad \overset{?}{\Leftrightarrow} \qquad p : D \rightharpoonup \mathbb{B}$$

1. **positive soundness**: $p\ d = \text{true} \Rightarrow P\ d$
   Look hard at the trace of the type checker

A better organisation

$$P : D \to \mathbb{P} \quad \overset{?}{\Leftrightarrow} \quad p : D \rightharpoonup \mathbb{B}$$

1. **positive soundness**: $p\ d = \text{true} \Rightarrow P\ d$
   Look hard at the trace of the type checker
2. **negative soundness**: $p\ d = \text{false} \Rightarrow \neg(P\ d)$
   Look harder at the trace of the type checker

A better organisation

$$P : D \to \mathbb{P} \qquad \overset{?}{\Leftrightarrow} \qquad p : D \rightharpoonup \mathbb{B}$$

1. **positive soundness**: $p\ d = \text{true} \Rightarrow P\ d$
   Look hard at the trace of the type checker
2. **negative soundness**: $p\ d = \text{false} \Rightarrow \neg(P\ d)$
   Look harder at the trace of the type checker

} Partial correctness

A better organisation

$$P : D \to \mathbb{P} \quad \overset{?}{\Leftrightarrow} \quad p : D \rightharpoonup \mathbb{B}$$

1. **positive soundness**: $p\ d = \text{true} \Rightarrow P\ d$
   Look hard at the trace of the type checker
2. **negative soundness**: $p\ d = \text{false} \Rightarrow \neg(P\ d)$
   Look harder at the trace of the type checker

} Partial correctness

3. **termination**: $(p\ d = \text{true}) \vee (p\ d = \text{false})$
   Still hard, of course...

A better organisation

$$P : D \to \mathbb{P} \qquad \overset{?}{\Leftrightarrow} \qquad p : D \rightharpoonup \mathbb{B}$$

1. **positive soundness**: $p\ d = \text{true} \Rightarrow P\ d$
   Look hard at the trace of the type checker
2. **negative soundness**: $p\ d = \text{false} \Rightarrow \neg(P\ d)$
   Look harder at the trace of the type checker

} Partial correctness

3. **termination**: $(p\ d = \text{true}) \vee (p\ d = \text{false})$
   Still hard, of course...
4. $\Rightarrow$ **completeness, decidability...**

A better organisation

$$P : D \to \mathbb{P} \quad \overset{?}{\Leftrightarrow} \quad p : D \rightharpoonup \mathbb{B}$$

1. **positive soundness**: $p\ d = \text{true} \Rightarrow P\ d$
   Look hard at the trace of the type checker
2. **negative soundness**: $p\ d = \text{false} \Rightarrow \neg(P\ d)$
   Look harder at the trace of the type checker

} Partial correctness

3. **termination**: $(p\ d = \text{true}) \vee (p\ d = \text{false})$
   Still hard, of course...
4. $\Rightarrow$ **completeness, decidability...**

Now we have a plan

A very innocuous (?) rule

zwarich
@zwarich@hachyderm.io

@carloangiuli @mevenlennonbertrand @markusde Meven's posts here have convinced me that any basic syntactic property of a type theory I dream up in my head are nontrivial and might indeed be false.

20 Aug 2026, 22:38 ·

1 boost · **0** quotes · **5** favourites

A very innocuous (?) rule

$$\textsc{AppCong}\ \frac{\Gamma \vdash f \equiv f' : \Pi\, x{:}\, A.B \qquad \Gamma \vdash u \equiv u' : A}{\Gamma \vdash f\, u \equiv f'\, u' : B[u]}$$

A very innocuous (?) rule

$$\textsc{AppCong}\ \frac{\Gamma \vdash f \equiv f' : \Pi\, x{:}\, A.B \qquad \Gamma \vdash u \equiv u' : A}{\Gamma \vdash f\, u \equiv f'\, u' : B[u]}$$

What are the pre- and post-conditions of definitional equality?
Are the input terms already well typed? Do they have the same type?

A very innocuous (?) rule

$$\textsc{AppCong}\ \frac{\Gamma \vdash f \equiv f' : \Pi\, x{:}\, A.B \qquad \Gamma \vdash u \equiv u' : A}{\Gamma \vdash f\, u \equiv f'\, u' : B[u]}$$

What are the pre- and post-conditions of definitional equality?
Are the input terms already well typed? Do they have the same type?

Definitional equality is bidirectional too

$$\frac{\Gamma \vdash f \equiv f' \triangleright \Pi\, x{:}\, A.B \qquad \Gamma \vdash u \equiv u' \triangleleft A}{\Gamma \vdash f\, u \equiv f'\, u' \triangleright B[u]}$$

$$\frac{\Gamma \vdash f \equiv f' \triangleright \Pi\, x{:}\, A.B \qquad \Gamma \vdash u \equiv u' \triangleleft A}{\Gamma \vdash f\, u \equiv f'\, u' \triangleright B[u]}$$

How do we ensure the second premise is ok?

$$\left.\begin{array}{l}\Gamma \vdash f : \Pi\, x{:}\, A.B \\ \Gamma \vdash f\, u : T\end{array}\right\} \overset{??}{\Rightarrow} \quad \Gamma \vdash u : A$$

$$\frac{\Gamma \vdash f \equiv f' \triangleright \Pi\, x\colon A.B \qquad \Gamma \vdash u \equiv u' \triangleleft A}{\Gamma \vdash f\, u \equiv f'\, u' \triangleright B[u]}$$

How do we ensure the second premise is ok?

$$\left.\begin{array}{l} \Gamma \vdash f : \Pi\, x\colon A.B \\ \Gamma \vdash f\, u : T \\ \quad \Rightarrow \Gamma \vdash f : \Pi\, x\colon A'.B' \\ \quad \wedge \quad \Gamma \vdash u : A' \end{array}\right\} \overset{??}{\Rightarrow} \quad \Gamma \vdash u : A$$

The $1000 properties: definitional inversion

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

$$\Pi\, x{:}\, A.B \equiv \Pi\, x{:}\, A'.B' \quad \overset{?}{\Rightarrow} \quad A \equiv A' \wedge B \equiv B'$$

The $1000 properties: definitional inversion

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

$$\begin{aligned}
\Pi\, x{:}\, A.B \equiv \Pi\, x{:}\, A'.B' &\overset{?}{\Rightarrow} A \equiv A' \wedge B \equiv B' \\
\Pi\, x{:}\, A.B \equiv \mathcal{U} &\overset{?}{\Rightarrow} \bot \\
\mathsf{S}\, m \equiv \mathsf{S}\, n &\overset{?}{\Rightarrow} m \equiv n \\
x\, u \equiv x\, u' &\overset{?}{\Rightarrow} u \equiv u' \\
&\dots
\end{aligned}$$

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

$$\begin{aligned}
\Pi\, x{:}\, A.B \equiv \Pi\, x{:}\, A'.B' &\overset{?}{\Rightarrow} A \equiv A' \wedge B \equiv B' \\
\Pi\, x{:}\, A.B \equiv \mathcal{U} &\overset{?}{\Rightarrow} \bot \\
\mathsf{S}\, m \equiv \mathsf{S}\, n &\overset{?}{\Rightarrow} m \equiv n \\
x\, u \equiv x\, u' &\overset{?}{\Rightarrow} u \equiv u' \\
&\overset{?}{\Rightarrow} \dots
\end{aligned}$$

The most important properties of all

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

$$\begin{aligned}
\Pi\, x{:}\, A.B \equiv \Pi\, x{:}\, A'.B' &\overset{?}{\Rightarrow} A \equiv A' \wedge B \equiv B' \\
\Pi\, x{:}\, A.B \equiv \mathcal{U} &\overset{?}{\Rightarrow} \bot \\
\mathrm{S}\, m \equiv \mathrm{S}\, n &\overset{?}{\Rightarrow} m \equiv n \\
x\, u \equiv x\, u' &\overset{?}{\Rightarrow} u \equiv u' \\
&\dots
\end{aligned}$$

The most important properties of all

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

Very specific to the initial model!

How to prove definitional inversions?

	Rewriting/Confluence	Logical relation	Gluing/Nf Model
Weak ambiant theory	✓	×	×
Scaling	✓	×	∼
η laws	×	✓	✓

Theory Comput Syst (2019) 63:647–665
https://doi.org/10.1007/s00224-018-9879-9

CrossMark

An Adequacy Theorem for Dependent Type Theory

Thierry Coquand[1] · **Simon Huber**[1]

Published online: 28 July 2018

Abstract We present a domain model of dependent type theory and use it to prove basic metatheoretic properties. In particular, we prove that two convertible terms have the same Böhm tree. The method used is reminiscent of the use of "inclusive predicates" in domain theory.

Keywords Dependent type theory · Domain theory · Finitary projections

ng/Nf Model

Weak ambia
theo

Scalir

η lav

×

∼

✓

How to prove definitional inversions?

	Rewriting/Confluence	Domain theory	Logical relation	Gluing/Nf Model
Weak ambiant theory	✓	✓	×	×
Scaling	✓	?	×	∼
η laws	×	✓	✓	✓

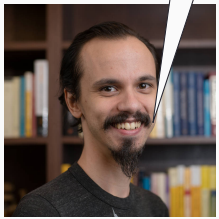
How to prove definitional inversions?

	Rewriting/Confluence	Domain theory	Logical relation	Gluing/Nf Model
Weak ambiant theory	✓	✓	×	×
Scaling	✓	?	×	~
η laws	×	✓	✓	✓

Some domain theory

From *Definitional Inversion, Without Normalisation* (Preprint)

I want to verify Lean's kernel!

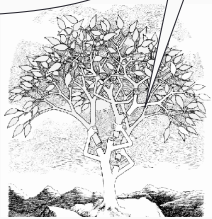


Mario Carneiro

I want dependent types and non-termination!

Stephanie Weirich

What about linear dependent types?



Adrien Frabetti
Mathieu

Domains might help...

Paul-André Melliès

Thierry Coquand

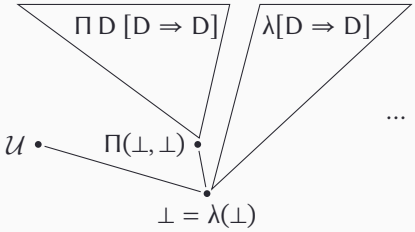
I've lost my intuitions,
is my 2018 proof correct?

Object theory $\approx$ MLTT '71 $= \mathcal{U} : \mathcal{U} + \Pi$ (β/η)

Object theory ≈ MLTT '71 = $\mathcal{U} : \mathcal{U}$ + Π (β/η)

Modelled in a universal domain D

D =



What we get out

$$\Pi\, x{:}\, A.B \equiv \Pi\, x{:}\, A'.B' \overset{?}{\Rightarrow} A \equiv A' \wedge B \equiv B' \qquad \text{“Injectivity for types”}$$

$$\Pi\, x{:}\, A.B \equiv \mathcal{U} \overset{\checkmark}{\Rightarrow} \bot \qquad \text{“No-confusion for types”}$$

$$\mathsf{S}\, m \equiv \mathsf{S}\, n \overset{\checkmark}{\Rightarrow} m \equiv n \qquad \text{“Inversion for datatypes”}$$

$$x\, u \equiv x\, u' \overset{?}{\Rightarrow} u \equiv u' \qquad \text{“Inversion for neutrals”}$$

$$\ldots$$

What we get out

$$\Pi\, x{:}\, A.B \equiv \Pi\, x{:}\, A'.B' \overset{?}{\Rightarrow} A \equiv A' \wedge B \equiv B' \qquad \text{“Injectivity for types”}$$

$$\Pi\, x{:}\, A.B \equiv \mathcal{U} \overset{\checkmark}{\Rightarrow} \bot \qquad \text{“No-confusion for types”}$$

$$\mathsf{S}\, m \equiv \mathsf{S}\, n \overset{\checkmark}{\Rightarrow} m \equiv n \qquad \text{“Inversion for datatypes”}$$

$$x\, u \equiv x\, u' \overset{?}{\Rightarrow} u \equiv u' \qquad \text{“Inversion for neutrals”}$$

$$\ldots$$

The injectivity problem

$$
\begin{array}{rrcl}
 & \Pi\, x{:}\, A.B & \equiv & \Pi\, x{:}\, A'.B' \\
\Rightarrow & [\![\Pi\, x{:}\, A.B]\!] & = & [\![\Pi\, x{:}\, A'.B']\!] \\
\Rightarrow & \Pi([\![A]\!], [\![B]\!]) & = & \Pi([\![A']\!], [\![B']\!]) \\
\Rightarrow & [\![A]\!] = [\![A']\!] & \wedge & [\![B]\!] = [\![B']\!] \\
\Rightarrow & & ?? &
\end{array}
$$

The injectivity problem

$$
\begin{array}{rrcl}
 & \Pi\, x\colon A.B & \equiv & \Pi\, x\colon A'.B' \\
\Rightarrow & [\![\Pi\, x\colon A.B]\!] & = & [\![\Pi\, x\colon A'.B']\!] \\
\Rightarrow & \Pi([\![A]\!], [\![B]\!]) & = & \Pi([\![A']\!], [\![B']\!]) \\
\Rightarrow & [\![A]\!] = [\![A']\!] & \wedge & [\![B]\!] = [\![B']\!] \\
\Rightarrow & & ?? &
\end{array}
$$

Two problems:

1. Coming back from the model
2. Functions $\neq$ open terms

Logical relations in a slide

Wanna show $\forall \Gamma, A, t \in \mathsf{Tm}(\Gamma, A).\ P(t)$?

Logical relations in a slide

Wanna show $\forall\, \Gamma, A, t \in \mathsf{Tm}(\Gamma, A).\ P(t)$?

Logical relations ≈ **displayed propositional model**

Logical relations in a slide

Wanna show $\forall\, \Gamma, A, t \in \mathsf{Tm}(\Gamma, A).\ P(t)$?

Logical relations ≈ **displayed propositional model** with a family P **defined by induction** on A

Logical relations in a slide

Wanna show $\forall \Gamma, A, t \in \mathsf{Tm}(\Gamma, A).\ P(t)$?

Logical relations ≈ displayed propositional model with a family P defined by induction on A

But: can't induct "on the type" if it's up to definitional equality!

Logical relations in a slide

Wanna show $\forall \Gamma, A, t \in \mathsf{Tm}(\Gamma, A).\ P(t)$?

Logical relations ≈ displayed propositional model with a family P defined by induction on A

But: can't induct "on the type" if it's up to definitional equality!

$$\Pi\, \Gamma, A \in \mathsf{Ty}(\Gamma). \quad P_{\mathsf{Ty}}(A)$$
$$\Pi\, \Gamma, A, t \in \mathsf{Tm}(\Gamma, A). \quad \mathcal{A} \in P_{\mathsf{Ty}}(A) \to P_{\mathsf{Tm}}(t)$$

where P_{Ty} is **structure** (= **data**) and P_{Tm} defined by induction on $\mathcal{A} \in P_{\mathsf{Ty}(A)}$
$P_{\mathsf{Ty}(A)}$ contains the normal form of A

Logical relations in a slide

Wanna show $\forall \Gamma, A, t \in \mathsf{Tm}(\Gamma, A).\ P(t)$?

Logical relations ≈ displayed propositional model with a family P defined by induction on A

But: can't induct "on the type" if it's up to definitional equality!

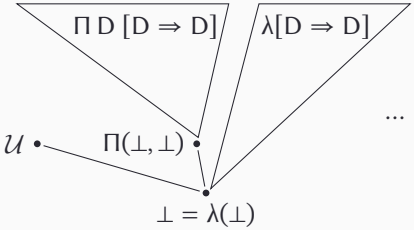
$$\Pi\, \Gamma, A \in \mathsf{Ty}(\Gamma). \quad P_{\mathsf{Ty}}(A)$$
$$\Pi\, \Gamma, A, t \in \mathsf{Tm}(\Gamma, A). \quad \mathcal{A} \in P_{\mathsf{Ty}}(A) \to P_{\mathsf{Tm}}(t)$$

where P_{Ty} is structure (= data) and P_{Tm} defined by induction on $\mathcal{A} \in P_{\mathsf{Ty}(A)}$
$P_{\mathsf{Ty}(A)}$ contains the normal form of A

But what if A has no normal form??

Compact approximations

D =

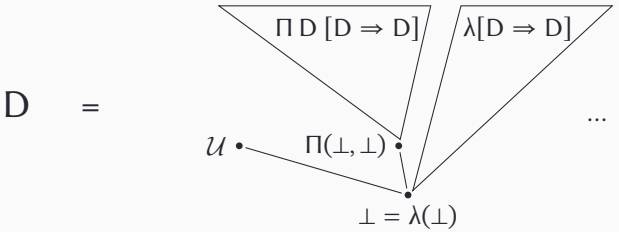


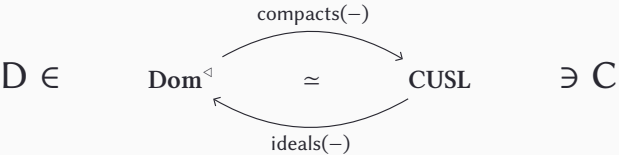
Compact approximations

$$\mathsf{D} = \mathcal{U} \bullet \quad \Pi(\bot,\bot) \bullet \quad \Pi\,\mathsf{D}\,[\mathsf{D} \Rightarrow \mathsf{D}] \quad \lambda[\mathsf{D} \Rightarrow \mathsf{D}] \quad \ldots \quad \bot = \lambda(\bot)$$

$$\mathsf{D} \in \mathbf{Dom}^{\triangleleft} \underset{\text{ideals}(-)}{\overset{\text{compacts}(-)}{\simeq}} \mathbf{CUSL} \ni \mathsf{C}$$

Compact approximations





Elements of D are characterised by a **set of compact approximations**
The set of compacts is **inductively generated**

Logical relations via domains, an intuition

- replace normal forms by domain elements
- use compacts to bootstrap an induction

$$[\![A]\!] \in \mathsf{D} \qquad \Rightarrow \qquad \mathrm{compacts}([\![A]\!]) \subseteq \mathsf{C}$$

Logical relations via domains, an intuition

- replace normal forms by domain elements
- use compacts to bootstrap an induction

$$
\begin{array}{rcl}
[\![A]\!] \in \mathsf{D} & \Rightarrow & \mathsf{compacts}([\![A]\!]) \subseteq \mathsf{C} \\
\\
P_{\mathsf{Ty}} : \mathsf{C} & \to & (\mathsf{Tm}(\Gamma, A) \to \mathbb{P}) \\
\bot & \mapsto & (\_ \mapsto \top) \\
 & \dots &
\end{array}
$$

Logical relations via domains, an intuition

- replace normal forms by domain elements
- use compacts to bootstrap an induction

$$
\begin{array}{rcl}
[\![A]\!] \in \mathsf{D} & \Rightarrow & \mathsf{compacts}([\![A]\!]) \subseteq \mathsf{C} \\
\\
P_{\mathsf{Ty}} : \mathsf{C} & \to & (\mathsf{Tm}(\Gamma, A) \to \mathbb{P}) \\
\bot & \mapsto & (\_ \mapsto \top) \\
 & \dots & \\
\\
P_{\mathsf{Ty}} : \mathsf{D} & \to & (\mathsf{Tm}(\Gamma, A) \to \mathbb{P}) \\
u & \mapsto & \bigcap_{c \leq u} P_{\mathsf{Ty}}(c)
\end{array}
$$

A sort-of closed model

No open terms/types in the model
$\longrightarrow$ similar to canonicity proofs

A sort-of closed model

No open terms/types in the model
$\longrightarrow$ similar to canonicity proofs

However, we can set $[\![x]\!] := \bot$

- all neutrals intepreted as $\bot$
- equality in the model is coarser than in syntax
- but equality of the glued model is just right!

We can still obtain **results about open terms**!

Fundamental lemma

For any $a \in \mathsf{C}$ such that $a \leq [\![A]\!]_\perp$ and $a : \mathsf{U}$, and $u \in \mathsf{C}$ such that $u \leq [\![t]\!]_\perp$ and $u : a$,

$$\Gamma \vdash t \equiv t' : A \qquad \Longrightarrow \qquad \Gamma \vDash t \equiv t' : A \,\big|_{u:a}.$$

Fundamental lemma

For any $a \in \mathsf{C}$ such that $a \leq [\![A]\!]_\perp$ and $a : \mathsf{U}$, and $u \in \mathsf{C}$ such that $u \leq [\![t]\!]_\perp$ and $u : a$,

$$\Gamma \vdash t \equiv t' : A \qquad \Longrightarrow \qquad \Gamma \vDash t \equiv t' : A \,|_{u:a}.$$

In particular:

$$\Gamma \vDash \Pi_{x:A}\, B \equiv \Pi_{x:A'}\, B' \,|_{\Pi(\perp,\perp)}$$

Corollary

If $\Gamma \vdash \Pi_{x:A}\, B \equiv \Pi_{x:A'}\, B'$, then $\Gamma \vdash A \equiv A'$ and $\Gamma, x{:}\, A \vdash B \equiv B'$.

Fundamental lemma

For any $a \in \mathsf{C}$ such that $a \leq [\![A]\!]_\perp$ and $a : \mathsf{U}$, and $u \in \mathsf{C}$ such that $u \leq [\![t]\!]_\perp$ and $u : a$,

$$\Gamma \vdash t \equiv t' : A \qquad \Longrightarrow \qquad \Gamma \vDash t \equiv t' : A \,|_{u:a}.$$

In particular:

$$\Gamma \vDash \Pi_{x:A}\, B \equiv \Pi_{x:A'}\, B' \,|_{\Pi(\perp,\perp)}$$

Corollary

If $\Gamma \vdash \Pi_{x:A}\, B \equiv \Pi_{x:A'}\, B'$, then $\Gamma \vdash A \equiv A'$ and $\Gamma, x{:}\, A \vdash B \equiv B'$.

Technical details in the paper, even more gory details in the formalisations

What we get out

$$\Pi\, x{:}\, A.B \equiv \Pi\, x{:}\, A'.B' \overset{\checkmark}{\Rightarrow} A \equiv A' \wedge B \equiv B' \qquad \text{“Injectivity for types”}$$

$$\Pi\, x{:}\, A.B \equiv \mathcal{U} \overset{\checkmark}{\Rightarrow} \bot \qquad \text{“No-confusion for types”}$$

$$\mathrm{S}\, m \equiv \mathrm{S}\, n \overset{\checkmark}{\Rightarrow} m \equiv n \qquad \text{“Inversion for datatypes”}$$

$$x\, u \equiv x\, u' \overset{?}{\Rightarrow} u \equiv u' \qquad \text{“Inversion for neutrals”}$$

$$\ldots$$

What we get out

$$\Pi\, x{:}\, A.B \equiv \Pi\, x{:}\, A'.B' \overset{\checkmark}{\Rightarrow} A \equiv A' \wedge B \equiv B' \qquad \text{“Injectivity for types”}$$

$$\Pi\, x{:}\, A.B \equiv \mathcal{U} \overset{\checkmark}{\Rightarrow} \bot \qquad \text{“No-confusion for types”}$$

$$\mathsf{S}\, m \equiv \mathsf{S}\, n \overset{\checkmark}{\Rightarrow} m \equiv n \qquad \text{“Inversion for datatypes”}$$

$$x\, u \equiv x\, u' \overset{?}{\Rightarrow} u \equiv u' \qquad \text{“Inversion for neutrals”}$$

$$\ldots$$

For which type theories?

Done:

- MLTT ’71, but with η

For which type theories?

Done:

- MLTT '71, but with η
- η for Σ (surjective pairing) & ⊤ (definitional singleton)
  - problematic for rewriting
  - non-confluent
  - very type-directed

For which type theories?

Done:

- MLTT '71, but with η
- η for Σ (surjective pairing) & ⊤ (definitional singleton)
  - problematic for rewriting
  - non-confluent
  - very type-directed
- general recursion (very easy!)
- hierarchy of universes
- datatypes (ℕ)
- equality (for now: proof relevant)
- strict propositions

For which type theories?

Done:

- MLTT '71, but with η
- η for Σ (surjective pairing) & ⊤ (definitional singleton)
  - problematic for rewriting
  - non-confluent
  - very type-directed
- general recursion (very easy!)
- hierarchy of universes
- datatypes (ℕ)
- equality (for now: proof relevant)
- strict propositions

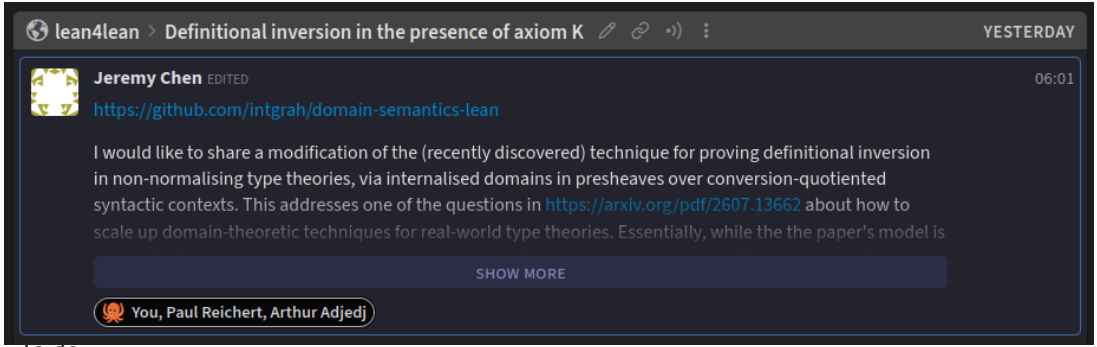
To do:

- observational equality (seems pretty straightforward?)
- irrelevant equality with Lean "K-like" reduction

For which type theories?

Done:

- MLTT ’71, but with η



To do:

- observational equality (seems pretty straightforward?)
- irrelevant equality with Lean “K-like” reduction

Wrapping up

We can (should!) **separate meta-theory and implementation**

Cannot beat Gödel, but can salvage a lot: **partial correctness** is within reach

Inversion properties are key, more so that normalisation

There is **space for new meta-theory** with different constraints

We can (should!) **separate meta-theory and implementation**

Cannot beat Gödel, but can salvage a lot: **partial correctness** is within reach

Inversion properties are key, more so that normalisation

There is **space for new meta-theory** with different constraints

- More features?
- Integration in MetaRocq/Lean4Lean?
- Neutral inversion?
- Typed rewriting?

We can (should!) **separate meta-theory and implementation**

Cannot beat Gödel, but can salvage a lot: **partial correctness** is within reach

Inversion properties are key, more so that normalisation

There is **space for new meta-theory** with different constraints

- More features?
- Integration in MetaRocq/Lean4Lean?
- Neutral inversion?
- Typed rewriting?

Thank you!

Bibliography

[Len25] Meven Lennon-Bertrand. 'What Does It Take to Certify a Conversion Checker?' In: *10th International Conference on Formal Structures for Computation and Deduction (FSCD 2025)*. 2025. DOI: 10.4230/LIPIcs.FSCD.2025.27.

[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.

[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.

[Car+26] Mario Carneiro et al. *Definitional Inversion, Without Normalisation*. 15th July 2026. DOI: 10.48550/ARXIV.2607.13662. arXiv: 2607.13662 [cs.LO].