Consolidating Equality in a Proof Irrelevant Universe

T. Felicissimo[2], R. Bocquet[1], K. Maillard[1], N. Tabareau[1], É. Tanter[1,3] & T. Winterhalter[1]

26 November 2026 @ TallCat seminar

[1]INRIA [2]University of Gothenburg [3]University of Chile

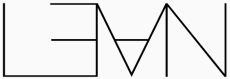
Preprint: `hal.science/hal-05688985`

Lean's equality

Last decade has seen the widespread adoption of a new proof assistant:

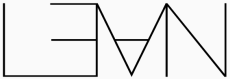
Lean's equality

Last decade has seen the widespread adoption of a new proof assistant:



Lean's equality

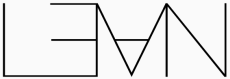
Last decade has seen the widespread adoption of a new proof assistant:



Its success suggests its type theory is well-adapted to formalizing mathematics

Lean's equality

Last decade has seen the widespread adoption of a new proof assistant:



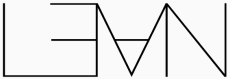
Its success suggests its type theory is well-adapted to formalizing mathematics

A key feature Universe Prop *impredicative* and *definitionally proof irrelevant*:

$$\frac{A : \mathsf{Type} \qquad P : \mathsf{Prop}}{\forall (x : A).P : \mathsf{Prop}} \qquad\qquad \frac{P : \mathsf{Prop} \qquad p : P \qquad q : P}{p \equiv q : P}$$

Lean's equality

Last decade has seen the widespread adoption of a new proof assistant:



Its success suggests its type theory is well-adapted to formalizing mathematics

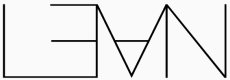
A key feature Universe Prop *impredicative* and *definitionally proof irrelevant*:

$$\frac{A : \mathsf{Type} \qquad P : \mathsf{Prop}}{\forall (x : A).P : \mathsf{Prop}} \qquad\qquad \frac{P : \mathsf{Prop} \qquad p : P \qquad q : P}{p \equiv q : P}$$

Importantly, Lean's equality type $a =_A b$ lives in Prop, yielding UIP *definitionally*

Lean's equality

Last decade has seen the widespread adoption of a new proof assistant:



Its success suggests its type theory is well-adapted to formalizing mathematics

A key feature Universe Prop *impredicative* and *definitionally proof irrelevant*:

$$\frac{A : \mathsf{Type} \qquad P : \mathsf{Prop}}{\forall (x : A).P : \mathsf{Prop}} \qquad\qquad \frac{P : \mathsf{Prop} \qquad p : P \qquad q : P}{p \equiv q : P}$$

Importantly, Lean's equality type $a =_A b$ lives in Prop, yielding UIP *definitionally*

Yet, can be eliminated into *relevant* types: from $e : n =_{\mathsf{Nat}} m$ and $l : \mathsf{Vec}_A\ n$ we get $l' : \mathsf{Vec}_A\ m$

Metatheoretic status

Recall the main feature of MLTT: it is both a *logic* and a *programming language*

Metatheoretic status

Recall the main feature of MLTT: it is both a *logic* and a *programming language*

Embodied by two main properties:

Metatheoretic status

Recall the main feature of MLTT: it is both a *logic* and a *programming language*

Embodied by two main properties:

- **Decidability of proof-checking**, which rests on the decidability of conversion

Metatheoretic status

Recall the main feature of MLTT: it is both a *logic* and a *programming language*

Embodied by two main properties:

- **Decidability of proof-checking**, which rests on the decidability of conversion
- **Canonicity**, ensuring that programs can be properly executed

Metatheoretic status

Recall the main feature of MLTT: it is both a *logic* and a *programming language*

Embodied by two main properties:

- **Decidability of proof-checking**, which rests on the decidability of conversion
- **Canonicity**, ensuring that programs can be properly executed

Unfortunately, for Lean's combination of equality and def. proof irrelevance, both are unclear

Metatheoretic status

Recall the main feature of MLTT: it is both a *logic* and a *programming language*

Embodied by two main properties:

- **Decidability of proof-checking**, which rests on the decidability of conversion
- **Canonicity**, ensuring that programs can be properly executed

Unfortunately, for Lean's combination of equality and def. proof irrelevance, both are unclear

Alternative approaches

Metatheoretic status

Recall the main feature of MLTT: it is both a *logic* and a *programming language*

Embodied by two main properties:

- **Decidability of proof-checking**, which rests on the decidability of conversion
- **Canonicity**, ensuring that programs can be properly executed

Unfortunately, for Lean's combination of equality and def. proof irrelevance, both are unclear

Alternative approaches

- Limit how proof-irrelevant equality can be eliminated (Rocq): severely hurts usability

Metatheoretic status

Recall the main feature of MLTT: it is both a *logic* and a *programming language*

Embodied by two main properties:

- **Decidability of proof-checking**, which rests on the decidability of conversion
- **Canonicity**, ensuring that programs can be properly executed

Unfortunately, for Lean's combination of equality and def. proof irrelevance, both are unclear

Alternative approaches

- Limit how proof-irrelevant equality can be eliminated (Rocq): severely hurts usability
- *Observational equality* [Altenkirch et al. 2007]: radical change to implementations

Metatheoretic status

Recall the main feature of MLTT: it is both a *logic* and a *programming language*

Embodied by two main properties:

- **Decidability of proof-checking**, which rests on the decidability of conversion
- **Canonicity**, ensuring that programs can be properly executed

Unfortunately, for Lean's combination of equality and def. proof irrelevance, both are unclear

Alternative approaches

- Limit how proof-irrelevant equality can be eliminated (Rocq): severely hurts usability
- *Observational equality* [Altenkirch et al. 2007]: radical change to implementations

In short, all have drawbacks...

Metatheoretic status

Recall the main feature of MLTT: it is both a *logic* and a *programming language*

Embodied by two main properties:

- **Decidability of proof-checking**, which rests on the decidability of conversion
- **Canonicity**, ensuring that programs can be properly executed

Unfortunately, for Lean's combination of equality and def. proof irrelevance, both are unclear

Alternative approaches

- Limit how proof-irrelevant equality can be eliminated (Rocq): severely hurts usability
- *Observational equality* [Altenkirch et al. 2007]: radical change to implementations

In short, all have drawbacks...

Our goal Clarify the status of Lean's proof-irrelevant equality, used in practice

Problem 1: Decidability of conversion

Lean's equality can be eliminated into relevant types using *transport*:

$$\mathsf{transp} : (P : A \to \mathsf{Type}) \to a =_A b \to P\ a \to P\ b$$

$$\mathsf{transp}\ P\ \mathsf{refl}\ p \equiv p$$

Problem 1: Decidability of conversion

Lean's equality can be eliminated into relevant types using *transport*:

$$\mathsf{transp} : (P : A \to \mathsf{Type}) \to a =_A b \to P\ a \to P\ b$$

$$\mathsf{transp}\ P\ \mathsf{refl}\ p \equiv p$$

With definitional proof irrelevance, any $e : a =_A b$ is convertible to refl when $a \equiv b$, so:

$$\mathsf{transp}\ P\ (e : a =_A b)\ p \equiv p \qquad \text{whenever } a \equiv b \qquad (\textsc{TranspIrr})$$

Problem 1: Decidability of conversion

Lean's equality can be eliminated into relevant types using *transport*:

$$\mathsf{transp} : (P : A \to \mathsf{Type}) \to a =_A b \to P\ a \to P\ b$$

$$\mathsf{transp}\ P\ \mathsf{refl}\ p \equiv p$$

With definitional proof irrelevance, any $e : a =_A b$ is convertible to refl when $a \equiv b$, so:

$$\mathsf{transp}\ P\ (e : a =_A b)\ p \equiv p \qquad \text{whenever } a \equiv b \qquad (\textsc{TranspIrr})$$

Danger Abel & Coquand [2020] proved that the reduction rule

$$\mathsf{transp}\ P\ (e : a =_A b)\ p \longrightarrow p \qquad \text{whenever } a \equiv b$$

breaks termination of proof reduction in presence of impredicativity!

Problem 1: Decidability of conversion

Lean's equality can be eliminated into relevant types using *transport*:

$$\mathsf{transp} : (P : A \to \mathsf{Type}) \to a =_A b \to P\ a \to P\ b$$

$$\mathsf{transp}\ P\ \mathsf{refl}\ p \equiv p$$

With definitional proof irrelevance, any $e : a =_A b$ is convertible to refl when $a \equiv b$, so:

$$\mathsf{transp}\ P\ (e : a =_A b)\ p \equiv p \qquad \text{whenever } a \equiv b \qquad (\textsc{TranspIrr})$$

Danger Abel & Coquand [2020] proved that the reduction rule

$$\mathsf{transp}\ P\ (e : a =_A b)\ p \longrightarrow p \qquad \text{whenever } a \equiv b$$

breaks termination of proof reduction in presence of impredicativity!

But proofs are irrelevant anyway!

Open problem [Abel & Coquand 2020] Is conversion still decidable?

Problem 2: Canonicity

To make it convenient for users, Lean fully embraces axioms tacitly assumed in mathematics: FunExt, PropExt, LEM, Hilbert's ε, ...

Problem 2: Canonicity

To make it convenient for users, Lean fully embraces axioms tacitly assumed in mathematics: FunExt, PropExt, LEM, Hilbert's ε, ...

Problem Axioms interact badly with computation

Problem 2: Canonicity

To make it convenient for users, Lean fully embraces axioms tacitly assumed in mathematics: FunExt, PropExt, LEM, Hilbert's ε, ...

Problem Axioms interact badly with computation

E.g., with FunExt (functional extensionality) we can prove

$$e : (\lambda x.x + 0) = (\lambda x.0 + x)$$

and so the closed term $t_{\mathsf{stuck}} : \mathsf{Nat} := \mathsf{transp}\ (\lambda_.\mathsf{Nat})\ e\ 0$ does not evaluate to 0

Problem 2: Canonicity

To make it convenient for users, Lean fully embraces axioms tacitly assumed in mathematics: FunExt, PropExt, LEM, Hilbert's ε, ...

Problem Axioms interact badly with computation

E.g., with FunExt (functional extensionality) we can prove

$$e : (\lambda x.x + 0) = (\lambda x.0 + x)$$

and so the closed term $t_{\mathsf{stuck}} : \mathsf{Nat} := \mathsf{transp}\ (\lambda_.\mathsf{Nat})\ e\ 0$ does not evaluate to 0

Lean's workaround Tactic `native_decide` checks computation after transp erasure, proves

$$e' : t_{\mathsf{stuck}} = 0$$

However, e' uses an unverified axiom: no formal justification that this equation holds!

Problem 2: Canonicity

To make it convenient for users, Lean fully embraces axioms tacitly assumed in mathematics: FunExt, PropExt, LEM, Hilbert's ε, ...

Problem Axioms interact badly with computation

E.g., with FunExt (functional extensionality) we can prove

$$e : (\lambda x.x + 0) = (\lambda x.0 + x)$$

and so the closed term $t_{\mathsf{stuck}} : \mathsf{Nat} := \mathsf{transp}\ (\lambda_.\mathsf{Nat})\ e\ 0$ does not evaluate to 0

Lean's workaround Tactic `native_decide` checks computation after transp erasure, proves

$$e' : t_{\mathsf{stuck}} = 0$$

However, e' uses an unverified axiom: no formal justification that this equation holds!

Natural questions Weaker form of canonicity, e.g. up to propositional equality?
Can the mechanism underlying `native_decide` be justified?

Towards a stronger computation rule?

Recall that Lean's rule removes transp as soon as $a \equiv b$:

$$\mathsf{transp}\ P\ (e : a =_A b)\ p \equiv p \qquad \text{whenever } a \equiv b \qquad (\textsc{TranspIrr})$$

Towards a stronger computation rule?

Recall that Lean's rule removes transp as soon as $a \equiv b$:

$$\mathsf{transp}\ P\ (e : a =_A b)\ p \equiv p \qquad \text{whenever } a \equiv b \qquad (\textsc{TranspIrr})$$

Observation The rule is well-typed as soon as $P\ a \equiv P\ b$: we don't need $a \equiv b$

Towards a stronger computation rule?

Recall that Lean's rule removes transp as soon as $a \equiv b$:

$$\mathsf{transp}\ P\ (e : a =_A b)\ p \equiv p \qquad \text{whenever } a \equiv b \qquad (\textsc{TranspIrr})$$

Observation The rule is well-typed as soon as $P\ a \equiv P\ b$: we don't need $a \equiv b$

We can consider a stronger rule that applies whenever the typing condition is met:

$$\mathsf{transp}\ P\ (e : a =_A b)\ p \equiv p \qquad \text{whenever } P\ a \equiv P\ b \qquad (\textsc{TranspMotive})$$

Towards a stronger computation rule?

Recall that Lean's rule removes transp as soon as $a \equiv b$:

$$\mathsf{transp}\ P\ (e : a =_A b)\ p \equiv p \qquad \text{whenever } a \equiv b \qquad (\textsc{TranspIrr})$$

Observation The rule is well-typed as soon as $P\ a \equiv P\ b$: we don't need $a \equiv b$

We can consider a stronger rule that applies whenever the typing condition is met:

$$\mathsf{transp}\ P\ (e : a =_A b)\ p \equiv p \qquad \text{whenever } P\ a \equiv P\ b \qquad (\textsc{TranspMotive})$$

Removes spurious transports, useful in practice (see examples in the paper):

- constant motives, e.g. now $t_{\mathsf{stuck}} \equiv 0$
- motives only inspecting head constructors, e.g. along $e : \mathsf{Some}\ x = \mathsf{Some}\ y$

Towards a stronger computation rule?

Recall that Lean's rule removes transp as soon as $a \equiv b$:

$$\mathsf{transp}\ P\ (e : a =_A b)\ p \equiv p \qquad \text{whenever } a \equiv b \qquad (\textsc{TranspIrr})$$

Observation The rule is well-typed as soon as $P\ a \equiv P\ b$: we don't need $a \equiv b$

We can consider a stronger rule that applies whenever the typing condition is met:

$$\mathsf{transp}\ P\ (e : a =_A b)\ p \equiv p \qquad \text{whenever } P\ a \equiv P\ b \qquad (\textsc{TranspMotive})$$

Removes spurious transports, useful in practice (see examples in the paper):

- constant motives, e.g. now $t_{\mathsf{stuck}} \equiv 0$
- motives only inspecting head constructors, e.g. along $e : \mathsf{Some}\ x = \mathsf{Some}\ y$

(Not enough to recover canonicity in general though)

Towards a stronger computation rule?

Recall that Lean's rule removes transp as soon as $a \equiv b$:

$$\mathsf{transp}\ P\ (e : a =_A b)\ p \equiv p \qquad \text{whenever } a \equiv b \qquad (\textsc{TranspIrr})$$

Observation The rule is well-typed as soon as $P\ a \equiv P\ b$: we don't need $a \equiv b$

We can consider a stronger rule that applies whenever the typing condition is met:

$$\mathsf{transp}\ P\ (e : a =_A b)\ p \equiv p \qquad \text{whenever } P\ a \equiv P\ b \qquad (\textsc{TranspMotive})$$

Removes spurious transports, useful in practice (see examples in the paper):

- constant motives, e.g. now $t_{\mathsf{stuck}} \equiv 0$
- motives only inspecting head constructors, e.g. along $e : \mathsf{Some}\ x = \mathsf{Some}\ y$

(Not enough to recover canonicity in general though)

Question What about the previous questions in this stronger setting?

This work

Contribution A thorough metatheoretic study of equality in a proof irrelevant universe

This work

Contribution A thorough metatheoretic study of equality in a proof irrelevant universe

For both Lean's TranspIrr and the stronger TranspMotive, we prove (formally in Rocq):

This work

Contribution A thorough metatheoretic study of equality in a proof irrelevant universe

For both Lean's TranspIrr and the stronger TranspMotive, we prove (formally in Rocq):

1. *Decidability of conversion*, settling the open problem of Abel & Coquand [2020]

This work

Contribution A thorough metatheoretic study of equality in a proof irrelevant universe

For both Lean's TranspIrr and the stronger TranspMotive, we prove (formally in Rocq):

1. *Decidability of conversion*, settling the open problem of Abel & Coquand [2020]
2. *Propositional canonicity*: If $\vdash t : \mathsf{Nat}$ then $\vdash e : t = \mathsf{S}^k(\mathsf{0})$ for some e, k,
   even in the presence of axioms: FunExt, PropExt, LEM, ... (but not Hilbert's ε)

This work

Contribution A thorough metatheoretic study of equality in a proof irrelevant universe

For both Lean's TranspIrr and the stronger TranspMotive, we prove (formally in Rocq):

1. *Decidability of conversion*, settling the open problem of Abel & Coquand [2020]
2. *Propositional canonicity*: If $\vdash t : \mathsf{Nat}$ then $\vdash e : t = \mathsf{S}^k(\mathsf{0})$ for some e, k,
   even in the presence of axioms: FunExt, $\mathsf{PropExt}$, LEM, ... (but not Hilbert's ε)
3. Correctness of *evaluation-via-erasure*: evaluating after erasing transports is sound

This work

Contribution A thorough metatheoretic study of equality in a proof irrelevant universe

For both Lean's TranspIrr and the stronger TranspMotive, we prove (formally in Rocq):

1. *Decidability of conversion*, settling the open problem of Abel & Coquand [2020]
2. *Propositional canonicity*: If $\vdash t : \mathsf{Nat}$ then $\vdash e : t = \mathsf{S}^k(\mathsf{0})$ for some e, k,
   even in the presence of axioms: FunExt, $\mathsf{PropExt}$, LEM, ... (but not Hilbert's ε)
3. Correctness of *evaluation-via-erasure*: evaluating after erasing transports is sound

Justified by these results, we also provide practical contributions:

This work

Contribution A thorough metatheoretic study of equality in a proof irrelevant universe

For both Lean's TranspIrr and the stronger TranspMotive, we prove (formally in Rocq):

1. *Decidability of conversion*, settling the open problem of Abel & Coquand [2020]
2. *Propositional canonicity*: If $\vdash t : \mathsf{Nat}$ then $\vdash e : t = \mathsf{S}^k(\mathsf{0})$ for some e, k,
   even in the presence of axioms: FunExt, PropExt, LEM, ... (but not Hilbert's ε)
3. Correctness of *evaluation-via-erasure*: evaluating after erasing transports is sound

Justified by these results, we also provide practical contributions:

1. TranspMotive implemented in Rocq 9.3 (via SProp's `Definitional UIP` flag)

This work

Contribution A thorough metatheoretic study of equality in a proof irrelevant universe

For both Lean's TranspIrr and the stronger TranspMotive, we prove (formally in Rocq):

1. *Decidability of conversion*, settling the open problem of Abel & Coquand [2020]
2. *Propositional canonicity*: If $\vdash t : \mathsf{Nat}$ then $\vdash e : t = \mathsf{S}^k(\mathsf{0})$ for some e, k, even in the presence of axioms: FunExt, PropExt, LEM, ... (but not Hilbert's ε)
3. Correctness of *evaluation-via-erasure*: evaluating after erasing transports is sound

Justified by these results, we also provide practical contributions:

1. TranspMotive implemented in Rocq 9.3 (via SProp's `Definitional UIP` flag)
2. An evaluation-via-erasure tactic for Rocq, analogue of Lean's `native_decide`

This work

Contribution A thorough metatheoretic study of equality in a proof irrelevant universe

For both Lean's TranspIrr and the stronger TranspMotive, we prove (formally in Rocq):

1. *Decidability of conversion*, settling the open problem of Abel & Coquand [2020]
2. *Propositional canonicity*: If $\vdash t : \mathsf{Nat}$ then $\vdash e : t = \mathsf{S}^k(\mathsf{0})$ for some e, k, even in the presence of axioms: FunExt, PropExt, LEM, ... (but not Hilbert's ε)
3. Correctness of *evaluation-via-erasure*: evaluating after erasing transports is sound

Justified by these results, we also provide practical contributions:

1. TranspMotive implemented in Rocq 9.3 (via SProp's `Definitional UIP` flag)
2. An evaluation-via-erasure tactic for Rocq, analogue of Lean's `native_decide`

Rest of the talk We focus only on the theory with TranspMotive, henceforth called $\mathsf{TT}_{\mathsf{sEq}}$

Decidability of conversion: the challenge

Decidability of conversion: the challenge

Usual proof strategy Define logical relation predicate $\Gamma \Vdash t : A$, ensuring t has normal form

Decidability of conversion: the challenge

Usual proof strategy Define logical relation predicate $\Gamma \Vdash t : A$, ensuring t has normal form

Then prove the logical relation "correctly models" every typing rule (fundamental theorem)

Decidability of conversion: the challenge

Usual proof strategy Define logical relation predicate $\Gamma \Vdash t : A$, ensuring t has normal form

Then prove the logical relation "correctly models" every typing rule (fundamental theorem)

Main case The typing rule for transp:

$$\frac{\Gamma, x : A \vdash P : \mathsf{U}_\ell \qquad \Gamma \vdash e : a =_A b \qquad \Gamma \vdash p : P[a/x]}{\Gamma \vdash \mathsf{transp}\ P\ e\ p : P[b/x]}$$

Decidability of conversion: the challenge

Usual proof strategy Define logical relation predicate $\Gamma \Vdash t : A$, ensuring t has normal form

Then prove the logical relation "correctly models" every typing rule (fundamental theorem)

Main case The typing rule for transp:

$$\frac{\Gamma, x : A \vdash P : \mathsf{U}_\ell \qquad \Gamma \vdash e : a =_A b \qquad \Gamma \vdash p : P[a/x]}{\Gamma \vdash \mathsf{transp}\ P\ e\ p : P[b/x]}$$

IH for P provides normal forms P_a for $P[a/x]$ and P_b for $P[b/x]$

We proceed with a case analysis:

Decidability of conversion: the challenge

Usual proof strategy Define logical relation predicate $\Gamma \Vdash t : A$, ensuring t has normal form

Then prove the logical relation "correctly models" every typing rule (fundamental theorem)

Main case The typing rule for transp:

$$\frac{\Gamma, x : A \vdash P : \mathsf{U}_\ell \qquad \Gamma \vdash e : a =_A b \qquad \Gamma \vdash p : P[a/x]}{\Gamma \vdash \mathsf{transp}\ P\ e\ p : P[b/x]}$$

IH for P provides normal forms P_a for $P[a/x]$ and P_b for $P[b/x]$
We proceed with a case analysis:

- If $P_a \neq P_b$: transp $P\ e\ p$ is stuck, and all stuck terms are in the logical relation ☺

Decidability of conversion: the challenge

Usual proof strategy Define logical relation predicate $\Gamma \Vdash t : A$, ensuring t has normal form

Then prove the logical relation "correctly models" every typing rule (fundamental theorem)

Main case The typing rule for transp:

$$\frac{\Gamma, x : A \vdash P : \mathsf{U}_\ell \qquad \Gamma \vdash e : a =_A b \qquad \Gamma \vdash p : P[a/x]}{\Gamma \vdash \mathsf{transp}\ P\ e\ p : P[b/x]}$$

IH for P provides normal forms P_a for $P[a/x]$ and P_b for $P[b/x]$
We proceed with a case analysis:

- If $P_a \neq P_b$: transp $P\ e\ p$ is stuck, and all stuck terms are in the logical relation ☺
- If $P_a = P_b$: transp $P\ e\ p$ reduces to p,
  IH gives $\Gamma \Vdash p : P[a/x]$, yet we need $\Gamma \Vdash p : P[b/x]\ldots$ ☹

Decidability of conversion: the challenge

Usual proof strategy Define logical relation predicate $\Gamma \Vdash t : A$, ensuring t has normal form

Then prove the logical relation "correctly models" every typing rule (fundamental theorem)

Main case The typing rule for transp:

$$\frac{\Gamma, x : A \vdash P : \mathsf{U}_\ell \qquad \Gamma \vdash e : a =_A b \qquad \Gamma \vdash p : P[a/x]}{\Gamma \vdash \mathsf{transp}\ P\ e\ p : P[b/x]}$$

Therefore, the fundamental theorem follows from:

$$\text{(LR Conv)} \quad \Gamma \Vdash A, B \text{ and } \Gamma \vdash A \equiv B : \mathsf{U}_\ell \text{ and } \Gamma \Vdash t : A \implies \Gamma \Vdash t : B$$

Decidability of conversion: the challenge

Usual proof strategy Define logical relation predicate $\Gamma \Vdash t : A$, ensuring t has normal form

Then prove the logical relation "correctly models" every typing rule (fundamental theorem)

Main case The typing rule for transp:

$$\frac{\Gamma, x : A \vdash P : \mathsf{U}_\ell \qquad \Gamma \vdash e : a =_A b \qquad \Gamma \vdash p : P[a/x]}{\Gamma \vdash \mathsf{transp}\ P\ e\ p : P[b/x]}$$

Therefore, the fundamental theorem follows from:

$$\text{(LR Conv)} \quad \Gamma \Vdash A, B \text{ and } \Gamma \vdash A \equiv B : \mathsf{U}_\ell \text{ and } \Gamma \Vdash t : A \implies \Gamma \Vdash t : B$$

Problem Ruling out head mismatch cases (e.g., $\mathsf{Nat} \equiv \mathsf{Bool}$) needs *definitional inversion*

Decidability of conversion: the challenge

Usual proof strategy Define logical relation predicate $\Gamma \Vdash t : A$, ensuring t has normal form

Then prove the logical relation "correctly models" every typing rule (fundamental theorem)

Main case The typing rule for transp:

$$\frac{\Gamma, x : A \vdash P : \mathsf{U}_\ell \qquad \Gamma \vdash e : a =_A b \qquad \Gamma \vdash p : P[a/x]}{\Gamma \vdash \mathsf{transp}\ P\ e\ p : P[b/x]}$$

Therefore, the fundamental theorem follows from:

$$\text{(LR Conv)} \quad \Gamma \Vdash A, B \text{ and } \Gamma \vdash A \equiv B : \mathsf{U}_\ell \text{ and } \Gamma \Vdash t : A \implies \Gamma \Vdash t : B$$

Problem Ruling out head mismatch cases (e.g., $\mathsf{Nat} \equiv \mathsf{Bool}$) needs *definitional inversion*

Fund Thm

Decidability of conversion: the challenge

Usual proof strategy Define logical relation predicate $\Gamma \Vdash t : A$, ensuring t has normal form

Then prove the logical relation "correctly models" every typing rule (fundamental theorem)

Main case The typing rule for transp:

$$\frac{\Gamma, x : A \vdash P : \mathsf{U}_\ell \qquad \Gamma \vdash e : a =_A b \qquad \Gamma \vdash p : P[a/x]}{\Gamma \vdash \mathsf{transp}\ P\ e\ p : P[b/x]}$$

Therefore, the fundamental theorem follows from:

$$\text{(LR Conv)} \quad \Gamma \Vdash A, B \text{ and } \Gamma \vdash A \equiv B : \mathsf{U}_\ell \text{ and } \Gamma \Vdash t : A \implies \Gamma \Vdash t : B$$

Problem Ruling out head mismatch cases (e.g., $\mathsf{Nat} \equiv \mathsf{Bool}$) needs *definitional inversion*



Decidability of conversion: the challenge

Usual proof strategy Define logical relation predicate $\Gamma \Vdash t : A$, ensuring t has normal form

Then prove the logical relation "correctly models" every typing rule (fundamental theorem)

Main case The typing rule for transp:

$$\frac{\Gamma, x : A \vdash P : \mathsf{U}_\ell \qquad \Gamma \vdash e : a =_A b \qquad \Gamma \vdash p : P[a/x]}{\Gamma \vdash \mathsf{transp}\ P\ e\ p : P[b/x]}$$

Therefore, the fundamental theorem follows from:

$$\text{(LR Conv)} \quad \Gamma \Vdash A, B \text{ and } \Gamma \vdash A \equiv B : \mathsf{U}_\ell \text{ and } \Gamma \Vdash t : A \implies \Gamma \Vdash t : B$$

Problem Ruling out head mismatch cases (e.g., $\mathsf{Nat} \equiv \mathsf{Bool}$) needs *definitional inversion*



Decidability of conversion: the challenge

Usual proof strategy Define logical relation predicate $\Gamma \Vdash t : A$, ensuring t has normal form

Then prove the logical relation "correctly models" every typing rule (fundamental theorem)

Main case The typing rule for transp:

$$\frac{\Gamma, x : A \vdash P : \mathsf{U}_\ell \qquad \Gamma \vdash e : a =_A b \qquad \Gamma \vdash p : P[a/x]}{\Gamma \vdash \mathsf{transp}\ P\ e\ p : P[b/x]}$$

Therefore, the fundamental theorem follows from:

$$\text{(LR Conv)} \quad \Gamma \Vdash A, B \text{ and } \Gamma \vdash A \equiv B : \mathsf{U}_\ell \text{ and } \Gamma \Vdash t : A \implies \Gamma \Vdash t : B$$

Problem Ruling out head mismatch cases (e.g., $\mathsf{Nat} \equiv \mathsf{Bool}$) needs *definitional inversion*



Decidability of conversion: breaking the circularity

We extend the theory with a trivial conversion operator:

$$\frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t : B} \qquad \frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t \equiv t : A}$$

Decidability of conversion: breaking the circularity

We extend the theory with a trivial conversion operator:

$$\frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t : B} \qquad \frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t \equiv t : A}$$

Observation For judgements not mentioning conv, extension does not change derivability

Decidability of conversion: breaking the circularity

We extend the theory with a trivial conversion operator:

$$\frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t : B} \qquad \frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t \equiv t : A}$$

But computationally, treat conv like cast of *observational equality*:

$$\mathsf{conv}_{\Pi(x:A_1).B_1,\Pi(x:A_2).B_2}\ f \longrightarrow \lambda(x_2 : A_2).\ \mathsf{let}\ x_1 := \mathsf{conv}_{A_2,A_1}\ x_2\ \mathsf{in}\ \mathsf{conv}_{B_1[x_1],B_2[x_2]}\ (f\ x_1)$$

$$\mathsf{conv}_{\mathsf{Nat},\mathsf{Bool}}\ t \not\longrightarrow$$

Decidability of conversion: breaking the circularity

We extend the theory with a trivial conversion operator:

$$\frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t : B} \qquad \frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t \equiv t : A}$$

But computationally, treat conv like cast of *observational equality*:

$$\mathsf{conv}_{\Pi(x:A_1).B_1,\Pi(x:A_2).B_2}\ f \quad \longrightarrow \quad \lambda(x_2 : A_2).\ \mathsf{let}\ x_1 := \mathsf{conv}_{A_2,A_1}\ x_2\ \mathsf{in}\ \mathsf{conv}_{B_1[x_1],B_2[x_2]}\ (f\ x_1)$$

$$\mathsf{conv}_{\mathsf{Nat},\mathsf{Bool}}\ t \quad \not\longrightarrow$$

Main insight Thanks to the reduction of conv, we can prove a variant of LR Conv:

(LR Conv') $\Gamma \Vdash A, B$ and $\Gamma \vdash A \equiv B : \mathsf{U}_\ell$ and $\Gamma \Vdash t : A \implies \Gamma \Vdash \mathsf{conv}_{A,B}\ t : B$

Decidability of conversion: breaking the circularity

We extend the theory with a trivial conversion operator:

$$\frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t : B} \qquad\qquad \frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t \equiv t : A}$$

But computationally, treat conv like cast of *observational equality*:

$$\mathsf{conv}_{\Pi(x:A_1).B_1,\Pi(x:A_2).B_2}\ f \longrightarrow \lambda(x_2 : A_2).\ \mathsf{let}\ x_1 := \mathsf{conv}_{A_2,A_1}\ x_2\ \mathsf{in}\ \mathsf{conv}_{B_1[x_1],B_2[x_2]}\ (f\ x_1)$$

$$\mathsf{conv}_{\mathsf{Nat},\mathsf{Bool}}\ t \not\longrightarrow$$

Main insight Thanks to the reduction of conv, we can prove a variant of LR Conv:

$$\text{(LR Conv')} \quad \Gamma \Vdash A, B \text{ and } \Gamma \vdash A \equiv B : \mathsf{U}_\ell \text{ and } \Gamma \Vdash t : A \implies \Gamma \Vdash \mathsf{conv}_{A,B}\ t : B$$

Indeed, head mismatch cases don't need definitional inversion:
if $\mathsf{Nat} \equiv \mathsf{Bool}$ and $t : \mathsf{Nat}$ then $\mathsf{conv}_{\mathsf{Nat},\mathsf{Bool}}\ t : \mathsf{Bool}$ is stuck, hence in the logical relation!

Decidability of conversion: breaking the circularity

We extend the theory with a trivial conversion operator:

$$\frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t : B} \qquad \frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t \equiv t : A}$$

Main insight Thanks to the reduction of conv, we can prove a variant of LR Conv:

(LR Conv') $\quad \Gamma \Vdash A, B$ and $\Gamma \vdash A \equiv B : \mathsf{U}_\ell$ and $\Gamma \Vdash t : A \implies \Gamma \Vdash \mathsf{conv}_{A,B}\ t : B$

Fundamental Thm follows from LR Conv', by making transp reduce to conv:

$\mathsf{transp}\ P\ e\ p \longrightarrow \mathsf{conv}_{P[a/x],P[b/x]}\ p$ whenever $\mathsf{nf}(P[a/x]) = \mathsf{nf}(P[b/x])$

Decidability of conversion: breaking the circularity

We extend the theory with a trivial conversion operator:

$$\frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t : B} \qquad \frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t \equiv t : A}$$

Main insight Thanks to the reduction of conv, we can prove a variant of LR Conv:

(LR Conv') $\Gamma \Vdash A, B$ and $\Gamma \vdash A \equiv B : \mathsf{U}_\ell$ and $\Gamma \Vdash t : A \implies \Gamma \Vdash \mathsf{conv}_{A,B}\ t : B$

Fundamental Thm follows from LR Conv', by making transp reduce to conv:

$\mathsf{transp}\ P\ e\ p \longrightarrow \mathsf{conv}_{P[a/x],P[b/x]}\ p$ whenever $\mathsf{nf}(P[a/x]) = \mathsf{nf}(P[b/x])$

In summary No more circularities!

Decidability of conversion: breaking the circularity

We extend the theory with a trivial conversion operator:

$$\frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t : B} \qquad \frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t \equiv t : A}$$

Main insight Thanks to the reduction of conv, we can prove a variant of LR Conv:

(LR Conv') $\quad \Gamma \Vdash A, B$ and $\Gamma \vdash A \equiv B : \mathsf{U}_\ell$ and $\Gamma \Vdash t : A \implies \Gamma \Vdash \mathsf{conv}_{A,B}\ t : B$

Fundamental Thm follows from LR Conv', by making transp reduce to conv:

$\mathsf{transp}\ P\ e\ p \longrightarrow \mathsf{conv}_{P[a/x],P[b/x]}\ p$ whenever $\mathsf{nf}(P[a/x]) = \mathsf{nf}(P[b/x])$

In summary No more circularities!

LR Conv'

Decidability of conversion: breaking the circularity

We extend the theory with a trivial conversion operator:

$$\frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t : B} \qquad\qquad \frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t \equiv t : A}$$

Main insight Thanks to the reduction of conv, we can prove a variant of LR Conv:

(LR Conv') $\quad \Gamma \Vdash A, B$ and $\Gamma \vdash A \equiv B : \mathsf{U}_\ell$ and $\Gamma \Vdash t : A \implies \Gamma \Vdash \mathsf{conv}_{A,B}\ t : B$

Fundamental Thm follows from LR Conv', by making transp reduce to conv:

$$\mathsf{transp}\ P\ e\ p \longrightarrow \mathsf{conv}_{P[a/x],P[b/x]}\ p \qquad \text{whenever } \mathsf{nf}(P[a/x]) = \mathsf{nf}(P[b/x])$$

In summary No more circularities!

LR Conv' $\Longrightarrow$ Fund Thm

Decidability of conversion: breaking the circularity

We extend the theory with a trivial conversion operator:

$$\frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t : B} \qquad \frac{\Gamma \vdash A \equiv B : \mathsf{U}_\ell \qquad \Gamma \vdash t : A}{\Gamma \vdash \mathsf{conv}_{A,B}\ t \equiv t : A}$$

Main insight Thanks to the reduction of conv, we can prove a variant of LR Conv:

(LR Conv') $\quad \Gamma \Vdash A, B$ and $\Gamma \vdash A \equiv B : \mathsf{U}_\ell$ and $\Gamma \Vdash t : A \implies \Gamma \Vdash \mathsf{conv}_{A,B}\ t : B$

Fundamental Thm follows from LR Conv', by making transp reduce to conv:

$\mathsf{transp}\ P\ e\ p \longrightarrow \mathsf{conv}_{P[a/x],P[b/x]}\ p$ whenever $\mathsf{nf}(P[a/x]) = \mathsf{nf}(P[b/x])$

In summary No more circularities!



Propositional canonicity: the proof strategy

Canonicity If $\vdash t : \mathsf{Nat}$ then $\vdash t \equiv \mathsf{S}^k(\mathsf{0}) : \mathsf{Nat}$ for some k

Propositional canonicity: the proof strategy

Canonicity If $\vdash t : \mathsf{Nat}$ then $\vdash t \equiv \mathsf{S}^k(\mathsf{0}) : \mathsf{Nat}$ for some k

Too strong in practice: fails with axioms...

Propositional canonicity: the proof strategy

Prop. canonicity If $\vdash t : \mathsf{Nat}$ then $\vdash e : t = \mathsf{S}^k(\mathsf{0})$ for some e, k

Propositional canonicity: the proof strategy

Prop. canonicity If $\vdash t : \mathsf{Nat}$ then $\vdash e : t = \mathsf{S}^k(0)$ for some e, k

Strategy *Propositional-canonicity-by-conservativity* [Felicissimo et al. 2026]:
if $\mathbb{T}$ has a conservative extension $\mathbb{T}'$, then canonicity of $\mathbb{T}'$ implies prop. canonicity of $\mathbb{T}$

Propositional canonicity: the proof strategy

Prop. canonicity If $\vdash t : \mathsf{Nat}$ then $\vdash e : t = \mathsf{S}^k(0)$ for some e, k

Strategy *Propositional-canonicity-by-conservativity* [Felicissimo et al. 2026]:
if $\mathbb{T}$ has a conservative extension $\mathbb{T}'$, then canonicity of $\mathbb{T}'$ implies prop. canonicity of $\mathbb{T}$

$$\mathbb{T} \qquad\qquad \subset \qquad\qquad \mathbb{T}'$$

$$\vdash t : \mathsf{Nat}$$

Propositional canonicity: the proof strategy

Prop. canonicity If $\vdash t : \mathsf{Nat}$ then $\vdash e : t = \mathsf{S}^k(0)$ for some e, k

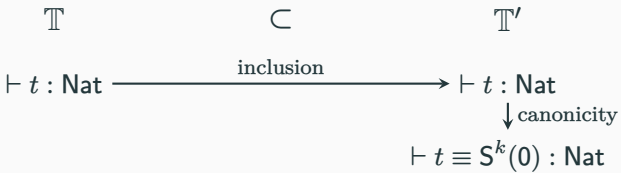
Strategy *Propositional-canonicity-by-conservativity* [Felicissimo et al. 2026]:
if $\mathbb{T}$ has a conservative extension $\mathbb{T}'$, then canonicity of $\mathbb{T}'$ implies prop. canonicity of $\mathbb{T}$

$$\begin{array}{ccc} \mathbb{T} & \subset & \mathbb{T}' \\ \vdash t : \mathsf{Nat} & \xrightarrow{\text{inclusion}} & \vdash t : \mathsf{Nat} \end{array}$$

Propositional canonicity: the proof strategy

Prop. canonicity If $\vdash t : \mathsf{Nat}$ then $\vdash e : t = \mathsf{S}^k(0)$ for some e, k

Strategy *Propositional-canonicity-by-conservativity* [Felicissimo et al. 2026]:
if $\mathbb{T}$ has a conservative extension $\mathbb{T}'$, then canonicity of $\mathbb{T}'$ implies prop. canonicity of $\mathbb{T}$



Propositional canonicity: the proof strategy

Prop. canonicity If $\vdash t : \mathsf{Nat}$ then $\vdash e : t = \mathsf{S}^k(0)$ for some e, k

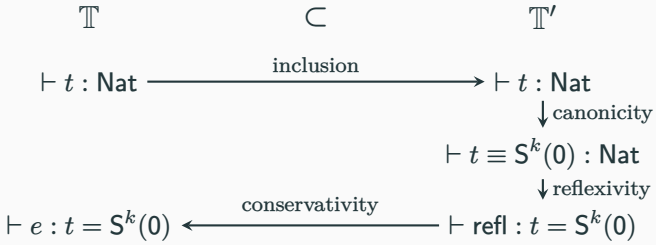
Strategy *Propositional-canonicity-by-conservativity* [Felicissimo et al. 2026]:
if $\mathbb{T}$ has a conservative extension $\mathbb{T}'$, then canonicity of $\mathbb{T}'$ implies prop. canonicity of $\mathbb{T}$

$$
\begin{array}{ccc}
\mathbb{T} & \subset & \mathbb{T}' \\
\vdash t : \mathsf{Nat} & \xrightarrow{\text{inclusion}} & \vdash t : \mathsf{Nat} \\
 & & \downarrow \text{canonicity} \\
 & & \vdash t \equiv \mathsf{S}^k(0) : \mathsf{Nat} \\
 & & \downarrow \text{reflexivity} \\
 & & \vdash \mathsf{refl} : t = \mathsf{S}^k(0)
\end{array}
$$

Propositional canonicity: the proof strategy

Prop. canonicity If $\vdash t : \mathsf{Nat}$ then $\vdash e : t = \mathsf{S}^k(0)$ for some e, k

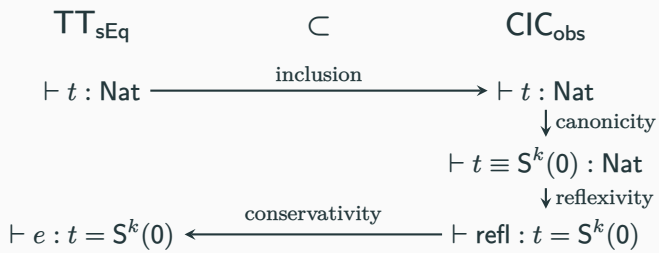
Strategy *Propositional-canonicity-by-conservativity* [Felicissimo et al. 2026]:
if $\mathbb{T}$ has a conservative extension $\mathbb{T}'$, then canonicity of $\mathbb{T}'$ implies prop. canonicity of $\mathbb{T}$



Propositional canonicity: the proof strategy

Prop. canonicity If $\vdash t : \mathsf{Nat}$ then $\vdash e : t = \mathsf{S}^k(0)$ for some e, k

Strategy *Propositional-canonicity-by-conservativity* [Felicissimo et al. 2026]:
if $\mathbb{T}$ has a conservative extension $\mathbb{T}'$, then canonicity of $\mathbb{T}'$ implies prop. canonicity of $\mathbb{T}$



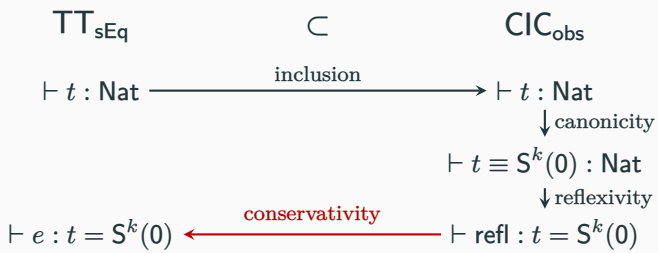
Thm [Pujet et al. 2025] $\mathsf{CIC}_{\mathsf{obs}}$ has canonicity, with any Prop axioms valid in its set model

Supports commonly used axioms: FunExt, $\mathsf{PropExt}$, LEM, ...

Propositional canonicity: the proof strategy

Prop. canonicity If $\vdash t : \mathsf{Nat}$ then $\vdash e : t = \mathsf{S}^k(0)$ for some e, k

Strategy *Propositional-canonicity-by-conservativity* [Felicissimo et al. 2026]:
if $\mathbb{T}$ has a conservative extension $\mathbb{T}'$, then canonicity of $\mathbb{T}'$ implies prop. canonicity of $\mathbb{T}$



Thm [Pujet et al. 2025] $\mathsf{CIC}_{\mathsf{obs}}$ has canonicity, with any Prop axioms valid in its set model

Supports commonly used axioms: FunExt, $\mathsf{PropExt}$, LEM, ...

Conservativity of $\mathsf{CIC}_{\mathsf{obs}}$ over $\mathsf{TT}_{\mathsf{sEq}}$

In $\mathsf{CIC}_{\mathsf{obs}}$, equality is eliminated with cast, which computes by case analysis on types:

$$\mathsf{cast}\ (\Pi(x : A_1).B_1)\ (\Pi(x : A_2).B_2)\ e\ f\ \equiv\ \lambda(x_2 : A_2).\ \mathsf{let}\ x_1 := \mathsf{cast}\ A_2\ A_1\ (\mathsf{inj}_1^{\Pi}\ e)\ x_2\ \mathsf{in}$$
$$\mathsf{cast}\ B_1[x_1]\ B_2[x_2]\ (\mathsf{inj}_2^{\Pi}\ e\ x_2)\ (f\ x_1)$$

Conservativity of $\mathsf{CIC}_{\mathsf{obs}}$ over $\mathsf{TT}_{\mathsf{sEq}}$

In $\mathsf{CIC}_{\mathsf{obs}}$, equality is eliminated with cast, which computes by case analysis on types:

$$\mathsf{cast}\ (\Pi(x : A_1).B_1)\ (\Pi(x : A_2).B_2)\ e\ f\ \equiv\ \lambda(x_2 : A_2).\ \mathsf{let}\ x_1 := \mathsf{cast}\ A_2\ A_1\ (\mathsf{inj}_1^{\Pi}\ e)\ x_2\ \mathsf{in}$$
$$\mathsf{cast}\ B_1[x_1]\ B_2[x_2]\ (\mathsf{inj}_2^{\Pi}\ e\ x_2)\ (f\ x_1)$$

where inj_1^{Π}, inj_2^{Π} are *injectivity axioms*, needed for cast to compute:

$$\mathsf{inj}_1^{\Pi} : (e : \Pi(x : A_1).B_1 = \Pi(x : A_2).B_2) \to A_2 = A_1$$
$$\mathsf{inj}_2^{\Pi} : (e : \Pi(x : A_1).B_1 = \Pi(x : A_2).B_2) \to (x_2 : A_2) \to$$
$$B_1[\mathsf{cast}\ A_2\ A_1\ (\mathsf{inj}_1^{\Pi}\ e)\ x_2] = B_2[x_2]$$

Conservativity of $\mathsf{CIC}_{\mathsf{obs}}$ over $\mathsf{TT}_{\mathsf{sEq}}$

In $\mathsf{CIC}_{\mathsf{obs}}$, equality is eliminated with cast, which computes by case analysis on types:

$$\mathsf{cast}\ (\Pi(x : A_1).B_1)\ (\Pi(x : A_2).B_2)\ e\ f\ \equiv\ \lambda(x_2 : A_2).\ \mathsf{let}\ x_1 := \mathsf{cast}\ A_2\ A_1\ (\mathsf{inj}_1^{\Pi}\ e)\ x_2\ \mathsf{in}\ \mathsf{cast}\ B_1[x_1]\ B_2[x_2]\ (\mathsf{inj}_2^{\Pi}\ e\ x_2)\ (f\ x_1)$$

Key insight Computation rules hold *propositionally* in $\mathsf{TT}_{\mathsf{sEq}}$, supposing FunExt, inj_1^{Π}, inj_2^{Π}

Hence, $\mathsf{CIC}_{\mathsf{obs}}$ only differs from $\mathsf{TT}_{\mathsf{sEq}}$ by making more equalities definitional

Conservativity of $\mathsf{CIC}_{\mathsf{obs}}$ over $\mathsf{TT}_{\mathsf{sEq}}$

In $\mathsf{CIC}_{\mathsf{obs}}$, equality is eliminated with cast, which computes by case analysis on types:

$$\mathsf{cast}\ (\Pi(x : A_1).B_1)\ (\Pi(x : A_2).B_2)\ e\ f\ \equiv\ \lambda(x_2 : A_2).\ \mathsf{let}\ x_1 := \mathsf{cast}\ A_2\ A_1\ (\mathsf{inj}_1^{\Pi}\ e)\ x_2\ \mathsf{in}\ \mathsf{cast}\ B_1[x_1]\ B_2[x_2]\ (\mathsf{inj}_2^{\Pi}\ e\ x_2)\ (f\ x_1)$$

Key insight Computation rules hold *propositionally* in $\mathsf{TT}_{\mathsf{sEq}}$, supposing FunExt, inj_1^{Π}, inj_2^{Π}

Hence, $\mathsf{CIC}_{\mathsf{obs}}$ only differs from $\mathsf{TT}_{\mathsf{sEq}}$ by making more equalities definitional

Same situation as conservativity of ETT over ITT + UIP + FunExt [Hofmann 1995]!

Conservativity of $\mathsf{CIC_{obs}}$ over $\mathsf{TT_{sEq}}$

In $\mathsf{CIC_{obs}}$, equality is eliminated with cast, which computes by case analysis on types:

$$\mathsf{cast}\ (\Pi(x : A_1).B_1)\ (\Pi(x : A_2).B_2)\ e\ f\ \equiv\ \lambda(x_2 : A_2).\ \mathsf{let}\ x_1 := \mathsf{cast}\ A_2\ A_1\ (\mathsf{inj}_1^\Pi\ e)\ x_2\ \mathsf{in}\ \mathsf{cast}\ B_1[x_1]\ B_2[x_2]\ (\mathsf{inj}_2^\Pi\ e\ x_2)\ (f\ x_1)$$

Key insight Computation rules hold *propositionally* in $\mathsf{TT_{sEq}}$, supposing FunExt, inj_1^Π, inj_2^Π

Hence, $\mathsf{CIC_{obs}}$ only differs from $\mathsf{TT_{sEq}}$ by making more equalities definitional

Same situation as conservativity of ETT over ITT + UIP + FunExt [Hofmann 1995]!

Adapting the technique of Winterhalter et al. [2019], we prove:

Thm $\mathsf{CIC_{obs}}$ is conservative over $\mathsf{TT_{sEq}}$, assuming axioms $\mathsf{FunExt}, \mathsf{inj}_1^\Pi, \mathsf{inj}_2^\Pi$ are declared

Conservativity of $\mathsf{CIC}_{\mathsf{obs}}$ over $\mathsf{TT}_{\mathsf{sEq}}$

In $\mathsf{CIC}_{\mathsf{obs}}$, equality is eliminated with cast, which computes by case analysis on types:

$$\mathsf{cast}\ (\Pi(x : A_1).B_1)\ (\Pi(x : A_2).B_2)\ e\ f\ \equiv\ \lambda(x_2 : A_2).\ \mathsf{let}\ x_1 := \mathsf{cast}\ A_2\ A_1\ (\mathsf{inj}_1^{\Pi}\ e)\ x_2\ \mathsf{in}\ \mathsf{cast}\ B_1[x_1]\ B_2[x_2]\ (\mathsf{inj}_2^{\Pi}\ e\ x_2)\ (f\ x_1)$$

Key insight Computation rules hold *propositionally* in $\mathsf{TT}_{\mathsf{sEq}}$, supposing FunExt, inj_1^{Π}, inj_2^{Π}

Hence, $\mathsf{CIC}_{\mathsf{obs}}$ only differs from $\mathsf{TT}_{\mathsf{sEq}}$ by making more equalities definitional

Same situation as conservativity of ETT over ITT + UIP + FunExt [Hofmann 1995]!

Adapting the technique of Winterhalter et al. [2019], we prove:

Thm $\mathsf{CIC}_{\mathsf{obs}}$ is conservative over $\mathsf{TT}_{\mathsf{sEq}}$, assuming axioms $\mathsf{FunExt}, \mathsf{inj}_1^{\Pi}, \mathsf{inj}_2^{\Pi}$ are declared

Cor $\mathsf{TT}_{\mathsf{sEq}}$ has propositional canonicity, under inherited technical conditions on the axioms

Evaluation-via-erasure

Using prop. canonicity, we justify the transport erasure performed by `native_decide`

Evaluation-via-erasure

Using prop. canonicity, we justify the transport erasure performed by `native_decide`

Erasure $|-|$ into an untyped λ-calculus, removing proofs, types and transports:

$$e, e' ::= x \mid \lambda x.\, e \mid e_1\ e_2 \mid 0 \mid \mathsf{S}(e) \mid \mathsf{rec}_{\mathsf{Nat}}(e_1, x\, y.\, e_2, e_3) \mid \square$$

$$|p| := \square \ \text{(if } p : P : \mathsf{Prop}) \qquad |A| := \square \ \text{(if } A \text{ is a type)} \qquad |\mathsf{transp}\ P\ e\ p| := |p|$$

with call-by-name reduction $\mapsto$, and $e \Downarrow k$ if e evaluates to $\mathsf{S}^k(0)$

Evaluation-via-erasure

Using prop. canonicity, we justify the transport erasure performed by `native_decide`

Erasure $|-|$ into an untyped λ-calculus, removing proofs, types and transports:

$$e, e' ::= x \mid \lambda x.\, e \mid e_1\ e_2 \mid 0 \mid \mathsf{S}(e) \mid \mathsf{rec}_{\mathsf{Nat}}(e_1, x\, y.\, e_2, e_3) \mid \square$$

$$|p| := \square \ \text{(if } p : P : \mathsf{Prop}) \qquad |A| := \square \ \text{(if } A \text{ is a type)} \qquad |\mathsf{transp}\ P\ e\ p| := |p|$$

with call-by-name reduction $\mapsto$, and $e \Downarrow k$ if e evaluates to $\mathsf{S}^k(0)$

Thm (evaluation-via-erasure) $\vdash t : \mathsf{Nat}$ implies $|t| \Downarrow k$ and $\vdash e : t = \mathsf{S}^k(0)$
for some e and (unique) k, under inherited technical conditions on the axioms

Evaluation-via-erasure

Using prop. canonicity, we justify the transport erasure performed by `native_decide`

Erasure $|-|$ into an untyped λ-calculus, removing proofs, types and transports:

$$e, e' ::= x \mid \lambda x.\, e \mid e_1\ e_2 \mid 0 \mid \mathsf{S}(e) \mid \mathsf{rec}_{\mathsf{Nat}}(e_1, x\, y.\, e_2, e_3) \mid \square$$

$$|p| := \square \ \text{(if } p : P : \mathsf{Prop}) \qquad |A| := \square \ \text{(if } A \text{ is a type)} \qquad |\mathsf{transp}\ P\ e\ p| := |p|$$

with call-by-name reduction $\mapsto$, and $e \Downarrow k$ if e evaluates to $\mathsf{S}^k(0)$

Thm (evaluation-via-erasure) $\vdash t : \mathsf{Nat}$ implies $|t| \Downarrow k$ and $\vdash e : t = \mathsf{S}^k(0)$
for some e and (unique) k, under inherited technical conditions on the axioms

Proof Refined propositional canonicity statement also gives *evaluation*: $t \Downarrow k$ in $\mathsf{CIC}_{\mathsf{obs}}$

Evaluation-via-erasure

Using prop. canonicity, we justify the transport erasure performed by `native_decide`

Erasure $|-|$ into an untyped λ-calculus, removing proofs, types and transports:

$$e, e' ::= x \mid \lambda x.\, e \mid e_1\ e_2 \mid 0 \mid \mathsf{S}(e) \mid \mathsf{rec}_{\mathsf{Nat}}(e_1, x\, y.\, e_2, e_3) \mid \square$$

$$|p| := \square \ \text{(if } p : P : \mathsf{Prop}) \qquad |A| := \square \ \text{(if } A \text{ is a type)} \qquad |\mathsf{transp}\ P\ e\ p| := |p|$$

with call-by-name reduction $\mapsto$, and $e \Downarrow k$ if e evaluates to $\mathsf{S}^k(0)$

Thm (evaluation-via-erasure) $\vdash t : \mathsf{Nat}$ implies $|t| \Downarrow k$ and $\vdash e : t = \mathsf{S}^k(0)$
for some e and (unique) k, under inherited technical conditions on the axioms

Proof Refined propositional canonicity statement also gives *evaluation*: $t \Downarrow k$ in $\mathsf{CIC}_{\mathsf{obs}}$

Extend $|-|$ with $|\mathsf{cast}\ A\ B\ e\ t| := |t|$, then show $\mathsf{CIC}_{\mathsf{obs}}$ step $t \longrightarrow u$ gives $|t| \mapsto |u|$ or $|t| = |u|$

Evaluation-via-erasure

Using prop. canonicity, we justify the transport erasure performed by `native_decide`

Erasure $|-|$ into an untyped λ-calculus, removing proofs, types and transports:

$$e, e' ::= x \mid \lambda x.\, e \mid e_1\ e_2 \mid 0 \mid \mathsf{S}(e) \mid \mathsf{rec}_{\mathsf{Nat}}(e_1, x\, y.\, e_2, e_3) \mid \square$$

$$|p| := \square \ \text{(if } p : P : \mathsf{Prop}) \qquad |A| := \square \ \text{(if } A \text{ is a type)} \qquad |\mathsf{transp}\ P\ e\ p| := |p|$$

with call-by-name reduction $\mapsto$, and $e \Downarrow k$ if e evaluates to $\mathsf{S}^k(0)$

Thm (evaluation-via-erasure) $\vdash t : \mathsf{Nat}$ implies $|t| \Downarrow k$ and $\vdash e : t = \mathsf{S}^k(0)$
for some e and (unique) k, under inherited technical conditions on the axioms

Proof Refined propositional canonicity statement also gives *evaluation*: $t \Downarrow k$ in $\mathsf{CIC}_{\mathsf{obs}}$

Extend $|-|$ with $|\mathsf{cast}\ A\ B\ e\ t| := |t|$, then show $\mathsf{CIC}_{\mathsf{obs}}$ step $t \longrightarrow u$ gives $|t| \mapsto |u|$ or $|t| = |u|$

By iterating over the reduction steps of $t \Downarrow k$ in $\mathsf{CIC}_{\mathsf{obs}}$, we get $|t| \Downarrow k$ in the λ-calculus

Conclusion

We proved Lean's proof-irrelevant equality has the expected metatheoretic properties:

Conclusion

We proved Lean's proof-irrelevant equality has the expected metatheoretic properties:

- *Decidable conversion*, settling the open problem of Abel & Coquand [2020]

Conclusion

We proved Lean's proof-irrelevant equality has the expected metatheoretic properties:

- *Decidable conversion*, settling the open problem of Abel & Coquand [2020]
- *Propositional canonicity*, even with commonly used axioms: FunExt, PropExt, LEM, ...

Conclusion

We proved Lean's proof-irrelevant equality has the expected metatheoretic properties:

- *Decidable conversion*, settling the open problem of Abel & Coquand [2020]
- *Propositional canonicity*, even with commonly used axioms: FunExt, PropExt, LEM, ...
- *Sound evaluation-via-erasure*, justifying the transport erasure of `native_decide`

Conclusion

We proved Lean's proof-irrelevant equality has the expected metatheoretic properties:

- *Decidable conversion*, settling the open problem of Abel & Coquand [2020]
- *Propositional canonicity*, even with commonly used axioms: FunExt, PropExt, LEM, ...
- *Sound evaluation-via-erasure*, justifying the transport erasure of `native_decide`

In practice TranspMotive in Rocq 9.3, and an evaluation-via-erasure tactic for Rocq

Conclusion

We proved Lean's proof-irrelevant equality has the expected metatheoretic properties:

- *Decidable conversion*, settling the open problem of Abel & Coquand [2020]
- *Propositional canonicity*, even with commonly used axioms: FunExt, PropExt, LEM, ...
- *Sound evaluation-via-erasure*, justifying the transport erasure of `native_decide`

In practice TranspMotive in Rocq 9.3, and an evaluation-via-erasure tactic for Rocq

Key takeaway techniques

- Observational conv breaks the circularity in the logical relation proof of decidability

Conclusion

We proved Lean's proof-irrelevant equality has the expected metatheoretic properties:

- *Decidable conversion*, settling the open problem of Abel & Coquand [2020]
- *Propositional canonicity*, even with commonly used axioms: FunExt, PropExt, LEM, ...
- *Sound evaluation-via-erasure*, justifying the transport erasure of `native_decide`

In practice TranspMotive in Rocq 9.3, and an evaluation-via-erasure tactic for Rocq

Key takeaway techniques

- Observational conv breaks the circularity in the logical relation proof of decidability
- *Propositional-canonicity-by-conservativity*, via $\mathsf{CIC}_{\mathsf{obs}}$

Conclusion

We proved Lean's proof-irrelevant equality has the expected metatheoretic properties:

- *Decidable conversion*, settling the open problem of Abel & Coquand [2020]
- *Propositional canonicity*, even with commonly used axioms: FunExt, PropExt, LEM, ...
- *Sound evaluation-via-erasure*, justifying the transport erasure of `native_decide`

In practice TranspMotive in Rocq 9.3, and an evaluation-via-erasure tactic for Rocq

Key takeaway techniques

- Observational conv breaks the circularity in the logical relation proof of decidability
- *Propositional-canonicity-by-conservativity*, via $\mathsf{CIC}_{\mathsf{obs}}$

Future work Combining with our previous work on accessibility [Felicissimo et al. 2026]: towards a well-behaved foundation for both Rocq and Lean?

Conclusion

We proved Lean's proof-irrelevant equality has the expected metatheoretic properties:

- *Decidable conversion*, settling the open problem of Abel & Coquand [2020]
- *Propositional canonicity*, even with commonly used axioms: FunExt, PropExt, LEM, ...
- *Sound evaluation-via-erasure*, justifying the transport erasure of `native_decide`

In practice TranspMotive in Rocq 9.3, and an evaluation-via-erasure tactic for Rocq

Key takeaway techniques

- Observational conv breaks the circularity in the logical relation proof of decidability
- *Propositional-canonicity-by-conservativity*, via $\mathsf{CIC}_{\mathsf{obs}}$

Future work Combining with our previous work on accessibility [Felicissimo et al. 2026]: towards a well-behaved foundation for both Rocq and Lean?

Thank you for your attention!