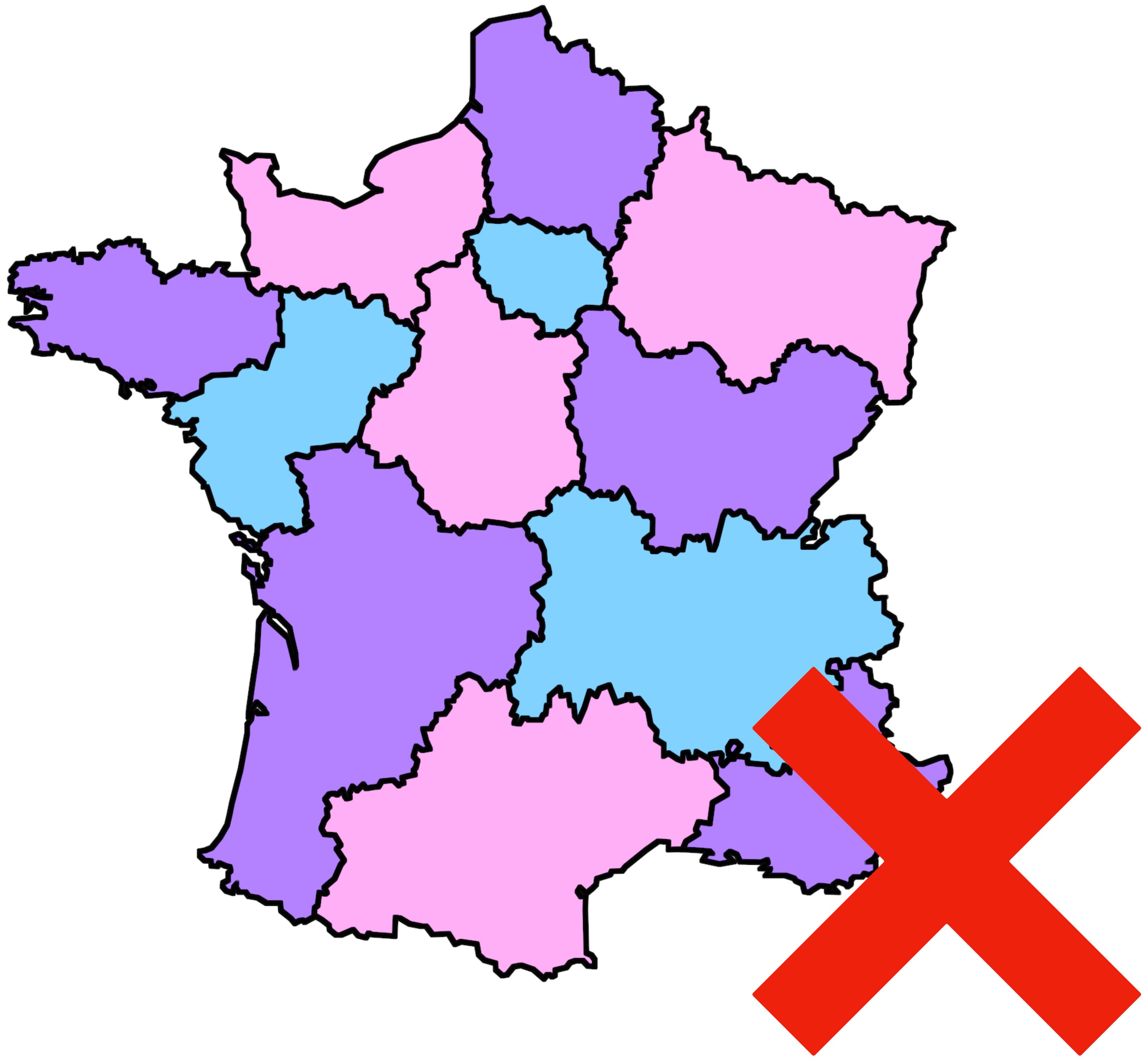
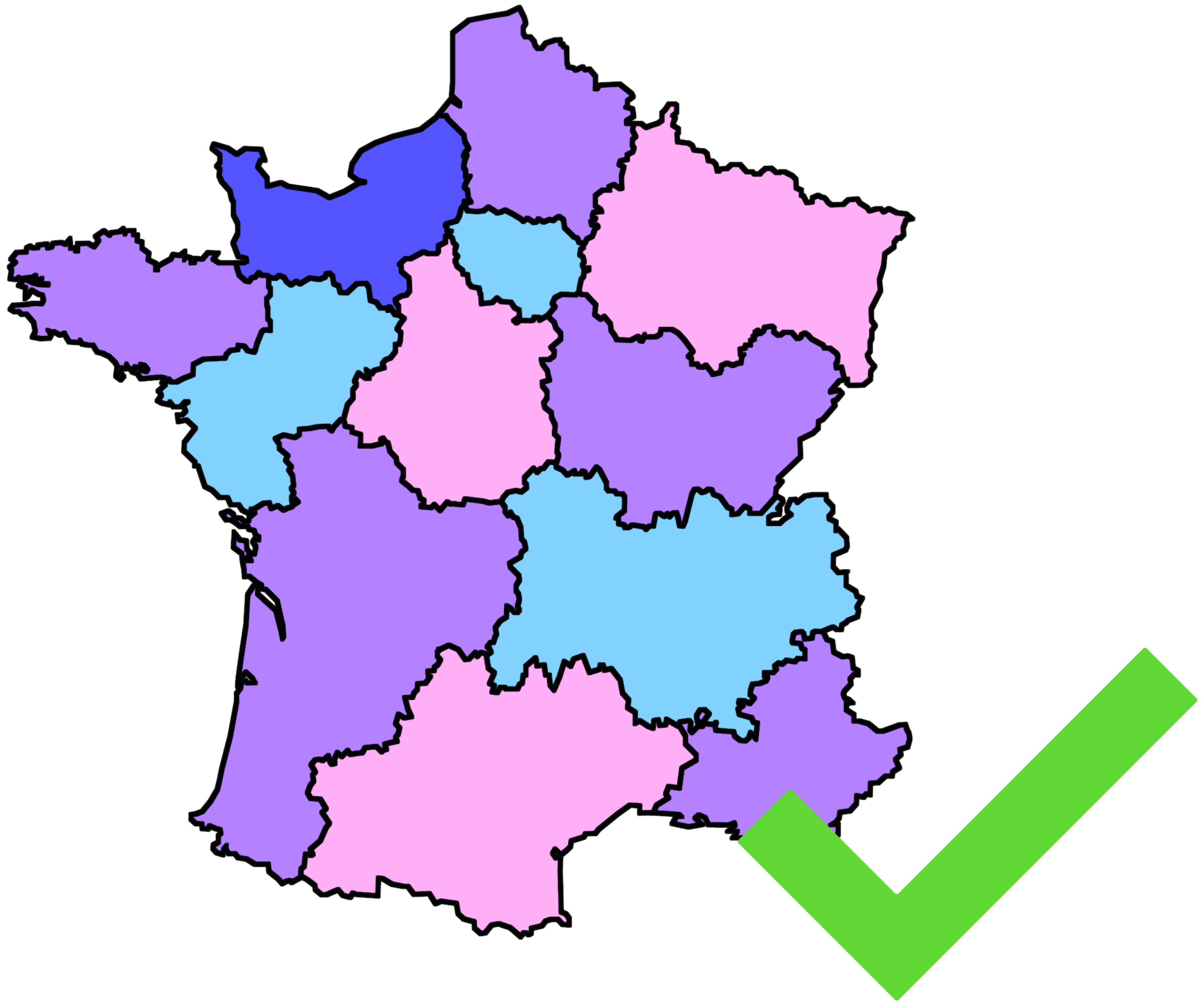
Computing with Extensionality Principles in Type Theory

13 December 2022

What is a proof?

« Given any map, you can always find a way to color all the regions so that two adjacent regions never have the same color, using at most four colors. »





What is a proof?

« Given any map, you can always find a way to color all the regions so that two adjacent regions never have the same color, using at most four colors. »

What is a proof?

« Given any map, you can always find a way to color all the regions so that two adjacent regions never have the same color, using at most four colors. »

What is a proof?

1. Every map has a finite number of regions

2. Every region has a finite number of adjacent regions

3. …

??. Given any map, you can always find a way to color all the regions so that two adjacent regions never have the same color, using at most four colors.

What is a proof?

1. Every map has a finite number of regions

2. Every region has a finite number of adjacent regions

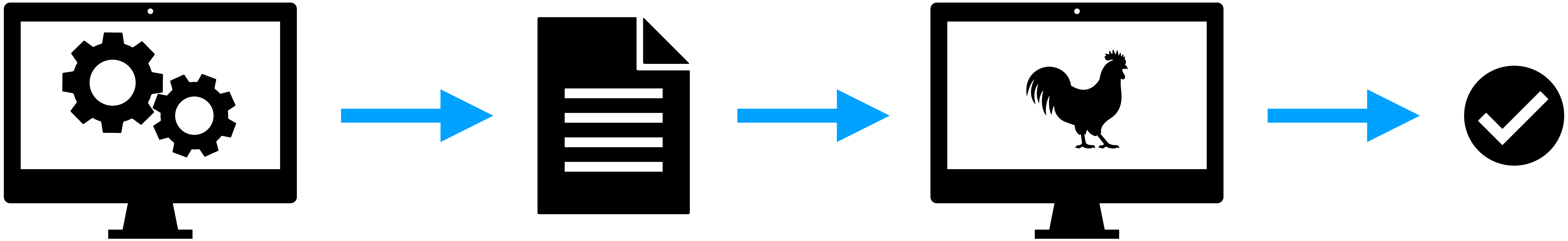
3. …

8937383. Given any map, you can always find a way to color all the regions so that two adjacent regions never have the same color, using at most four colors.

Appel and Haken 1976, *Every planar map is four colorable*

What is a proof assistant?

Four color theorem verified with the Coq proof assistant by Gonthier, 2008



We have to teach the computer how to read and check mathematics.
→ for this, we need a mathematical theory of mathematics!

Coq, Lean, Agda… speak the language of dependent type theory.

Gonthier 2008, *Formal Proof — The Four-Color Theorem*

Martin-Löf Type Theory

MLTT is a sweet spot in the Curry-Howard correspondence.

Powerful…

Expressive enough to do a lot of mathematics

Sufficient to define most computable functions

You do not need a lemma to prove that 3+9 is 12.

…but tractable

Decidable

Your computer can always tell whether your proof is correct

Normalization and canonicity

Closed terms always compute!

Martin-Löf Type Theory

MLTT is not without flaws: the inductive equality type encodes equality of programs, not equality of behaviors.

```
Inductive eq (A : Type) (a : A) : A -> Type :=
| eq_refl : eq A a a
```

No function extensionality

You can prove $\forall$`n. n+1` = `1+n`, but you cannot prove `λn.n+1` = `λn.1+n`

No quotient types

Given a relation R on a type A, you cannot form the type A/R

Martin-Löf Type Theory

MLTT is not without flaws: the inductive equality type encodes equality of programs, not equality of behaviors.

```
Inductive eq (A : Type) (a : A) : A -> Type :=
| eq_refl : eq A a a
```

In fact, equality is decidable in the empty context:

Canonicity → every closed proof of equality computes to `eq_refl`, which means the two sides have to be convertible

Decidable typing → the type-checker can decide if `eq_refl` applies

Martin-Löf Type Theory

Workarounds?

Use axioms: simply postulate function extensionality, etc

Breaks canonicity

Use setoids: equip every type with an equivalence relation, and ensure that functions preserve them

Does not scale to large proofs

Add the reflection rule to merge conversion and the inductive equality (ETT)

Breaks decidability

Use cubical type theory

Relies on complex rules from homotopy theory, computation is not always efficient

Martin-Löf Type Theory

Workarounds?

Use setoids: equip every type with an equivalence relation, and ensure that functions preserve them

Does not scale to large proofs

Add the reflection rule to merge conversion and the inductive equality (ETT)

Breaks decidability

Use cubical type theory

Relies on complex rules from homotopy theory, computation is not always efficient

Use observational type theory

Less well-understood meta-theoretic properties (is it compatible with impredicativity?)

Abel et al. 2020, Failure of normalization in impredicative [...]

Contributions

A meta-theory for (impredicative) observational type theory

- A formal calculus CC^{obs}
- Normalization, decidability, consistency, canonicity, proof-theoretic strength
- (Almost) everything is formalized in Agda

Toward a translation of univalent type theory in CC^{obs}

- Using the cubical model of Cohen *et al*.
- Homotopy canonicity, decidability, proof-theoretic results

Cubical synthetic homotopy theory

- Short and clean proofs that rely on computation in Cubical Agda

Observational Type Theory and CC^{obs}

Instead of being an inductive datatype, the equality is defined by recursion on the types.

$$\texttt{S (S 0)} \sim_{\mathbb{N}} \texttt{S (S 0)} \longrightarrow \texttt{S 0} \sim_{\mathbb{N}} \texttt{S 0} \longrightarrow \texttt{0} \sim_{\mathbb{N}} \texttt{0} \longrightarrow \top$$

$$\texttt{f} \sim_{A \to B} \texttt{g} \longrightarrow \prod (\texttt{x} : \texttt{A}) .\ \texttt{f x} \sim_{B} \texttt{g x}$$

Observational equality types are *definitionally proof-irrelevant*.

Altenkirch *et al.* 2017, Observational Equality Now!
Altenkirch 1999, Extensional equality in intensional type theory

Observational Type Theory and CC^{obs}

Observational equality types are *definitionally proof-irrelevant*.

Predicative hierarchy of types	Impredicative universe of propositions
$\mathtt{Type}_0 < \mathtt{Type}_1 < \ldots$	`Prop`
Datatypes	Logical constraints
Computable functions	Observational equality
Constructive proofs	Computationally irrelevant proofs

Altenkirch *et al.* 2017, Observational Equality Now!

Altenkirch 1999, Extensional equality in intensional type theory

Observational Type Theory and CC^{obs}

The observational equality lives in `Prop`. How do we eliminate it?

$$\frac{\mathtt{P} : \mathbb{N} \to \mathtt{Type} \qquad \mathtt{n}, \mathtt{m} : \mathbb{N} \qquad \mathtt{e} : \mathtt{m} \sim_{\mathbb{N}} \mathtt{n} \qquad \mathtt{t} : \mathtt{P}\ \mathtt{m}}{\mathtt{??} : \mathtt{P}\ \mathtt{n}}$$

Observational Type Theory and CC^{obs}

The observational equality lives in $\texttt{Prop}$. How do we eliminate it?

We use a type-casting operator:

$$\frac{A, B : \texttt{Type} \qquad e : A \sim_{\texttt{Type}} B \qquad x : A}{\texttt{cast}(A, B, e, x) : B}$$

$$\frac{P : \mathbb{N} \to \texttt{Type} \qquad n, m : \mathbb{N} \qquad e : m \sim_{\mathbb{N}} n \qquad t : P\ m}{??\ : P\ n}$$

Observational Type Theory and CC^{obs}

The observational equality lives in `Prop`. How do we eliminate it?

We use a type-casting operator:

$$\frac{A, B : \mathtt{Type} \qquad e : A \sim_{\mathtt{Type}} B \qquad x : A}{\mathtt{cast}(A, B, e, x) : B}$$

$$\frac{P : \mathbb{N} \to \mathtt{Type} \qquad n, m : \mathbb{N} \qquad e : m \sim_{\mathbb{N}} n \qquad t : P\ m}{\mathtt{cast}(P\ m, P\ n, \mathtt{ap}\ f\ e, t) : P\ n}$$

Observational Type Theory and CC^{obs}

$$\mathtt{cast}(\boxed{\mathbb{N}, \mathbb{N}}, e, S\ 0) \longrightarrow S\,(\mathtt{cast}(\mathbb{N}, \mathbb{N}, e, 0)) \longrightarrow S\ 0$$

$$\mathtt{cast}(\boxed{A \to B, A' \to B'}, e, f)$$

$$\longrightarrow \lambda(x : A').\ \mathtt{cast}(B, B', \mathtt{snd}\ e, f\ \mathtt{cast}(A', A, \mathtt{fst}\ e, x))$$

$$e : (A \to B) \sim_{\mathtt{Type}} (A' \to B')$$

Observational Type Theory and CC^{obs}

$\texttt{cast}(\mathbb{N}, \mathbb{N}, e, \texttt{S}\ \texttt{0}) \longrightarrow \texttt{S}\ (\texttt{cast}(\mathbb{N}, \mathbb{N}, e, \texttt{0})) \longrightarrow \texttt{S}\ \texttt{0}$

$\texttt{cast}(A \to B, A' \to B', e, f)$

$\longrightarrow \lambda(x : A').\ \texttt{cast}(B, B', \texttt{snd}\ e, f\ \texttt{cast}(A', A, \texttt{fst}\ e, x))$

$e : (A' \sim_{\texttt{Type}} A) \times (B \sim_{\texttt{Type}} B')$

Observational Type Theory and CC^{obs}

CC^{obs} supports basic inductive datatypes.

Dependent sums, integers, lists, W-types, etc. work as in MLTT.

Indexed inductive types are more interesting!

```
Inductive eq (A : Type) (a : A) : A -> Type :=
| eq_refl : eq A a a
```

With cast, we can show that $a \sim_A b \longleftrightarrow \mathtt{eq}\ A\ a\ b$

Thus we can show function extensionality for the inductive equality

Observational Type Theory and CC^{obs}

```
Inductive eq (A : Type) (a : A) : A -> Type :=
| eq_refl : eq A a a
```

We add a new way to inhabit eq

$$\texttt{eq\_cast : a} \sim_{\texttt{A}} \texttt{b -> eq A a b}$$

The observational equality equates `eq_refl` and `eq_cast`

$$\texttt{eq\_refl} \sim_{\texttt{eq A a a}} \texttt{eq\_cast e} \longrightarrow \texttt{T}$$

Observational Type Theory and CC^{obs}

```
Inductive eq (A : Type) (a : A) : A -> Type :=
| eq_refl : eq A a a
```

We add a new way to inhabit eq

eq_cast : a $\sim_A$ b -> eq A a a

The J eliminator computes on eq_cast

J(A, t, B, b, t', eq_cast e)
  ⟶ cast(B t eq_refl, B t' (eq_cast e), e', b)

OTT analogue of Swan's identity type

Andrew Swan 2014, An algebraic weak factorisation system [...]

Observational Type Theory and CC^{obs}

Alternative solution: have the observational equality compute on reflexivity

We add a new conversion rule

new

$$\frac{\texttt{A} \equiv \texttt{B}}{\texttt{cast(A, B, e, t)} \equiv \texttt{t}}$$

This rule is not very gracefully handled by reduction, but we can implement it in the comparison algorithm for neutral values, as suggested by Allais *et al*.

Indexed inductive types become simpler! We can now use the usual trick of adding an equality proof in constructors.

Allais *et al*. 2013, New Equations for Neutral Terms

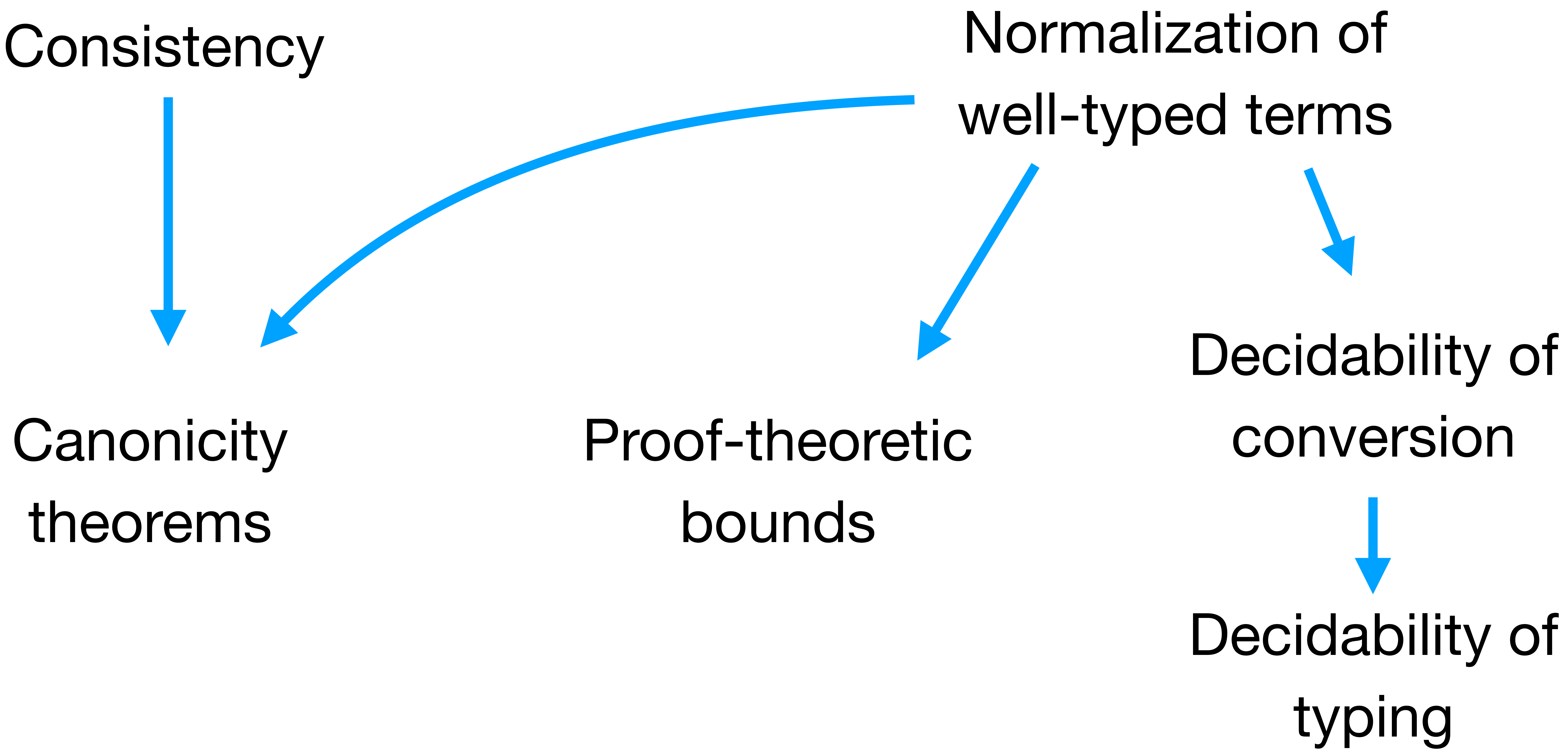
Observational Type Theory and CC^{obs}

With these ingredients, CC^{obs} supports all typing rules and computations rules of MLTT, and adds extensionality principles (funext, propext).

On top of this, we can add new types:

- Quotients of a type by a *proof-irrelevant* equivalence relation
- Irrelevant squash types
- Subset types

Meta-theory of CC^{obs}



Meta-theory of CC^{obs}

Consistency is proved by constructing a model in ZF set theory with Grothendieck universes.

From there, we obtain that

- There are no inhabitants of $\perp$ in the empty context
- There are no closed proofs of anti-diagonal equalities between types

Meta-theory of CC^{obs}

Normalization, canonicity and decidability of conversion are proved by constructing a reducibility model.

We used the inductive-recursive framework of Abel, Öhman and Vezzosi to formally prove these results in Agda.

Using an inductive construction of the universe is essential:
the observational equality and the cast operator compute on types.
The proof would not work with a open universe *à la* reducibility candidates

We extended the inductive-recursive model to support proof-irrelevant impredicative propositions.

Abel et al. 2018, Decidability of conversion for type theory [...]

Meta-theory of CC^{obs}

Normalization, canonicity and decidability of conversion are proved by constructing a reducibility model.

Since propositions are proof-irrelevant, they do not play any role in computation. We account for this in our model:
all proofs of propositions are reducible as long as they are well typed.

→ The reducibility model alone is not sufficient to derive canonicity.
But we can recover it with the help of the consistency theorem.

Abel et al. 2018, Decidability of conversion for type theory [...]

Meta-theory of CC^{obs}

Normalization, canonicity and decidability of conversion are proved by constructing a reducibility model.

Our reducibility model can be encoded in bare MLTT by replacing small induction recursion with inductive types.

→ Any integer function that can be defined in CC^{obs} can also be defined in MLTT.

→ The power of impredicativity is confined to the irrelevant layer.

Abel et al. 2018, Decidability of conversion for type theory […]

Toward a cubical translation into CC^{obs}

The univalence axiom is the cornerstone of HoTT.
Univalence is implemented as a postulate → it blocks computation.

The cubical model of Cohen et al. gives a constructive interpretation of univalence in fibrant cubical presheaves

The "prefascist" translation of Pédrot builds intensional presheaf models that respect computation:
if $\Gamma \vdash t \equiv u$ then $[\,\Gamma\,] \vdash [\,t\,] \equiv [\,u\,]$

Prefascist translation + fibration structures = cubical translation

Pédrot 2020, Russian constructivism in a prefascist theory
Cohen *et al*. 2016, Cubical type theory

Toward a cubical translation into CC^{obs}

Prefascist translation + fibration structures = cubical translation

Consequences

- A machine-checked computational interpretation of univalence
- Homotopy canonicity
- Univalent type theory cannot define more integer functions than MLTT

Orton *et al*. 2017, Axioms for modelling cubical type theory in a topos

Pédrot 2020, Russian constructivism in a prefascist theory

Cohen *et al*. 2016, Cubical type theory

Toward a cubical translation into CC^{obs}

So far, I have a machine-checked translation of

- the integers
- function types, dependent products and function extensionality
- the equality type
- a non-fibrant universe

Missing piece of the puzzle: the gluing construction

It seems rather involved, but doable

Cubical Synthetic Homotopy

Claim: a computational system allows cleaner and more elegant proofs

In order to support this, I proved some basic result of homotopy theory in Cubical Agda, in collaboration with Mörtberg

	HoTT Agda	Cubical Agda
$\Omega(S^1) = Z$	90 loc	50 loc
$T^2 = S^1 \times S^1$	150 loc	25 loc
3×3 lemma	3000 loc	200 loc
Associativity of join	210 loc	90 loc

Cubical Synthetic Homotopy

Claim: a computational system allows cleaner and more elegant proofs

In order to support this, I proved some basic result of homotopy theory in Cubical Agda, in collaboration with Mörtberg

Note that a lot of the improvement is due to the inherently cubical quality of the results, not only to computation!

Perspectives

Finish the cubical translation (Gluing)

Implement CC^{obs} in Coq?

Many people in the community want a good proof assistant for the internal language of a 1-topos.

Add HITs, and translate my proof of the Hopf fibration to obtain a CC^{obs} definition of the Brunerie number. Does it compute?

Thank you!