Computing with Univalence in Coq using Intensional Presheaves

Loïc Pujet

July 11, 2020

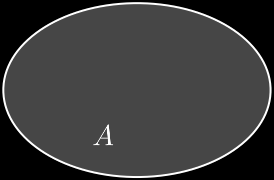
Homotopy Type Theory

Synthetic reasoning about spaces through dependent type theory

Homotopy Type Theory

Synthetic reasoning about spaces through dependent type theory

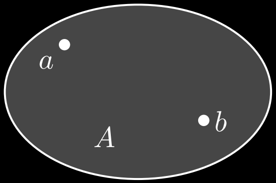
- $A : \mathcal{U}$



Types are spaces

Synthetic reasoning about spaces through dependent type theory

- $A : \mathcal{U}$
- $a, b : A$

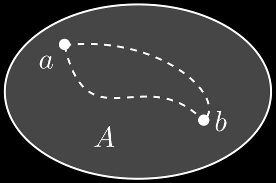


Terms are points

Homotopy Type Theory

Synthetic reasoning about spaces through dependent type theory

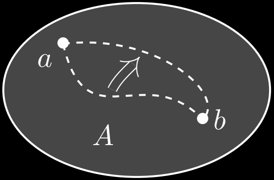
- $A : \mathcal{U}$
- $a, b : A$
- $e, f : a =_A b$



(Propositional) equality proofs are paths

Synthetic reasoning about spaces through dependent type theory

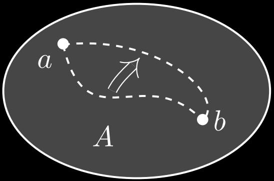
- $A : \mathcal{U}$
- $a, b : A$
- $e, f : a =_A b$
- $h : e =_{a=b} f$



Equalities between equalities are homotopies

Synthetic reasoning about spaces through dependent type theory

- $A : \mathcal{U}$
- $a, b : A$
- $e, f : a =_A b$
- $h : e =_{a=b} f$



And so on...

HoTT is based on intensional MLTT plus

HoTT is based on intensional MLTT plus

Univalence Axiom

$$(A =_{\mathcal{U}} B) \simeq (A \simeq B)$$

"Equivalences are equalities between types"

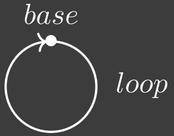
HoTT is based on intensional MLTT plus

Univalence Axiom

$$(A =_{\mathcal{U}} B) \simeq (A \simeq B)$$

"Equivalences are equalities between types"

Higher Inductive Types



$\texttt{Inductive}\ S^1 : \mathcal{U}\ \texttt{:=}$
$|\ base\ :\ S^1$
$|\ loop\ :\ base =_{S^1} base$

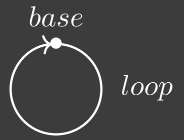
HoTT is based on intensional MLTT plus

Univalence Axiom

$$(A =_{\mathcal{U}} B) \simeq (A \simeq B)$$

"Equivalences are equalities between types"

Higher Inductive Types



$\texttt{Inductive}\ S^1 : \mathcal{U}\ \texttt{:=}$
$\mid base : S^1$
$\mid loop : base =_{S^1} base$

Which do not have computational content as is.

The Cubical Model

The Cubical Model

The Cubical Model[1] is an interpretation of HoTT in constructive set theory. Types are interpreted as **cubical presheaves**.

[1] Coquand et al., 2013

The Cubical Model[1] is an interpretation of HoTT in constructive set theory. Types are interpreted as **cubical presheaves**.

A cubical presheaf is a functor

$$F : \square \to \mathrm{Set}$$

from the category of cubes to Set.

Basically what you get by gluing sets of points, lines, squares, cubes, ... along their faces.

[1] Coquand et al., 2013

The Cubical Model

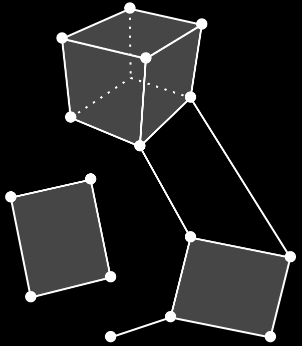
The Cubical Model[1] is an interpretation of HoTT in constructive set theory. Types are interpreted as **cubical presheaves**.

A cubical presheaf is a functor

$$F : \square \to \mathrm{Set}$$

from the category of cubes to Set.

Basically what you get by gluing sets of points, lines, squares, cubes, ... along their faces.



[1] Coquand et al., 2013

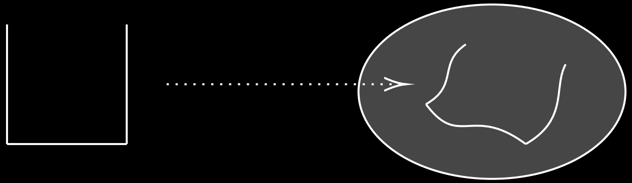
The Cubical Model

But not every cubical presheaf can be a type! Types are **fibrant** cubical presheaves.

The Cubical Model

But not every cubical presheaf can be a type! Types are **fibrant** cubical presheaves.

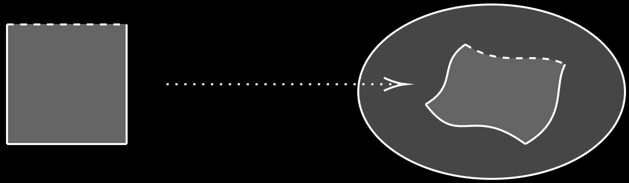
Fibrancy structure (roughly) : maps any open n-dimensional box to a n-dimensional cube that extends it



The Cubical Model

But not every cubical presheaf can be a type! Types are **fibrant** cubical presheaves.

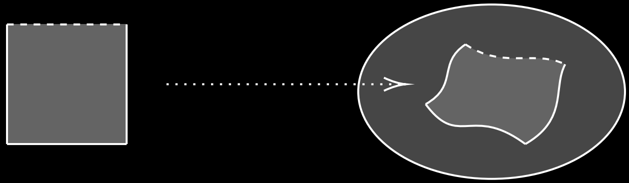
Fibrancy structure (roughly) : maps any open n-dimensional box to a n-dimensional cube that extends it



The Cubical Model

But not every cubical presheaf can be a type! Types are **fibrant** cubical presheaves.

Fibrancy structure (roughly) : maps any open n-dimensional box to a n-dimensional cube that extends it



The box filling operation implements the **J eliminator** for the equalities (higher dimensional faces) of the cubical structure.

The Cubical Model

Fibrant cubical presheaves are stable under all of the constructs of MLTT, so one can build a model with them.

The Cubical Model

Fibrant cubical presheaves are stable under all of the constructs of MLTT, so one can build a model with them.

This models also happens to realize function extensionality, univalence, and HITs $\Rightarrow$ full model of HoTT.

The Cubical Model

Fibrant cubical presheaves are stable under all of the constructs of MLTT, so one can build a model with them.

This models also happens to realize function extensionality, univalence, and HITs $\Rightarrow$ full model of HoTT.

It has been reified into **Cubical Type Theory**, which satisfies canonicity and univalence. However, normalization and decidability of type-checking are still open.

Models as Syntactic Translations

Models as Syntactic Translations

Translations are a class of models for dependent type theory.
Judgements in the source theory $\mathcal{S}$ are mapped to the target theory $\mathcal{T}$:

Models as Syntactic Translations

Translations are a class of models for dependent type theory.
Judgements in the source theory $\mathcal{S}$ are mapped to the target theory $\mathcal{T}$:

$$\Gamma \vdash_{\mathcal{S}} t : A$$

Models as Syntactic Translations

Translations are a class of models for dependent type theory.
Judgements in the source theory $\mathcal{S}$ are mapped to the target theory $\mathcal{T}$:

$$\Gamma \vdash_{\mathcal{S}} t : A \qquad \rightsquigarrow \qquad [\Gamma] \vdash_{\mathcal{T}} [t] : [A]$$

Models as Syntactic Translations

Translations are a class of models for dependent type theory.
Judgements in the source theory $\mathcal{S}$ are mapped to the target theory $\mathcal{T}$:

$$\Gamma \vdash_{\mathcal{S}} t : A \qquad \rightsquigarrow \qquad [\Gamma] \vdash_{\mathcal{T}} [t] : [A]$$

Translations do this inductively by translating every typing rule:

Models as Syntactic Translations

Translations are a class of models for dependent type theory.
Judgements in the source theory $\mathcal{S}$ are mapped to the target theory $\mathcal{T}$:

$$\Gamma \vdash_{\mathcal{S}} t \,:\, A \qquad \rightsquigarrow \qquad [\Gamma] \vdash_{\mathcal{T}} [t] \,:\, [A]$$

Translations do this inductively by translating every typing rule:

$$\frac{\Gamma, x : A \vdash_{\mathcal{S}} t : B}{\Gamma \vdash_{\mathcal{S}} \lambda x \,.\, t \,:\, \Pi\,(x : A)\,.\, B}$$

Models as Syntactic Translations

Translations are a class of models for dependent type theory.
Judgements in the source theory $\mathcal{S}$ are mapped to the target theory $\mathcal{T}$:

$$\Gamma \vdash_{\mathcal{S}} t \,:\, A \qquad \rightsquigarrow \qquad [\Gamma] \vdash_{\mathcal{T}} [t] \,:\, [A]$$

Translations do this inductively by translating every typing rule:

$$\frac{\Gamma, x : A \vdash_{\mathcal{S}} t : B}{\Gamma \vdash_{\mathcal{S}} \lambda x \,.\, t \,:\, \Pi\,(x : A) \,.\, B} \qquad \rightsquigarrow \qquad \frac{[\Gamma], x : [A] \vdash_{\mathcal{T}} [t] : [B]}{[\Gamma] \vdash_{\mathcal{T}} [\lambda x \,.\, t] \,:\, \Pi\,(x : [A]) \,.\, [B]}$$

Models as Syntactic Translations

Translations are a class of models for dependent type theory.
Judgements in the source theory $\mathcal{S}$ are mapped to the target theory $\mathcal{T}$:

$$\Gamma \vdash_{\mathcal{S}} t \,:\, A \qquad \rightsquigarrow \qquad [\Gamma] \vdash_{\mathcal{T}} [t] \,:\, [A]$$

Translations do this inductively by translating every typing rule:

$$\frac{\Gamma, x : A \vdash_{\mathcal{S}} t : B}{\Gamma \vdash_{\mathcal{S}} \lambda x \,.\, t \,:\, \Pi\,(x : A)\,.\,B} \qquad \rightsquigarrow \qquad \frac{[\Gamma], x : [A] \vdash_{\mathcal{T}} [t] : [B]}{[\Gamma] \vdash_{\mathcal{T}} [\lambda x \,.\, t] \,:\, \Pi\,(x : [A])\,.\,[B]}$$

And they have to preserve definitional equality between terms:

Translations are a class of models for dependent type theory.
Judgements in the source theory $\mathcal{S}$ are mapped to the target theory $\mathcal{T}$:

$$\Gamma \vdash_{\mathcal{S}} t \,:\, A \qquad \rightsquigarrow \qquad [\Gamma] \vdash_{\mathcal{T}} [t] \,:\, [A]$$

Translations do this inductively by translating every typing rule:

$$\frac{\Gamma, x : A \vdash_{\mathcal{S}} t : B}{\Gamma \vdash_{\mathcal{S}} \lambda x \,.\, t \,:\, \Pi\,(x : A) \,.\, B} \qquad \rightsquigarrow \qquad \frac{[\Gamma], x : [A] \vdash_{\mathcal{T}} [t] : [B]}{[\Gamma] \vdash_{\mathcal{T}} [\lambda x \,.\, t] \,:\, \Pi\,(x : [A]) \,.\, [B]}$$

And they have to preserve definitional equality between terms:

$$t \equiv_{\mathcal{S}} u$$

Models as Syntactic Translations

Translations are a class of models for dependent type theory.
Judgements in the source theory $\mathcal{S}$ are mapped to the target theory $\mathcal{T}$:

$$\Gamma \vdash_{\mathcal{S}} t \,:\, A \qquad \rightsquigarrow \qquad [\Gamma] \vdash_{\mathcal{T}} [t] \,:\, [A]$$

Translations do this inductively by translating every typing rule:

$$\frac{\Gamma, x : A \vdash_{\mathcal{S}} t : B}{\Gamma \vdash_{\mathcal{S}} \lambda x \,.\, t \,:\, \Pi\,(x : A)\,.\,B} \qquad \rightsquigarrow \qquad \frac{[\Gamma], x : [A] \vdash_{\mathcal{T}} [t] : [B]}{[\Gamma] \vdash_{\mathcal{T}} [\lambda x \,.\, t] \,:\, \Pi\,(x : [A])\,.\,[B]}$$

And they have to preserve definitional equality between terms:

$$t \equiv_{\mathcal{S}} u \qquad \rightsquigarrow \qquad [t] \equiv_{\mathcal{T}} [u]$$

Models as Syntactic Translations

A translation induces a new definitional equality on $\mathcal{S}$ that extends $\equiv_{\mathcal{S}}$

$$t \cong u \qquad \Leftrightarrow \qquad [t] \equiv_{\mathcal{T}} [u]$$

Models as Syntactic Translations

A translation induces a new definitional equality on $\mathcal{S}$ that extends $\equiv_{\mathcal{S}}$

$$t \cong u \qquad \Leftrightarrow \qquad [t] \equiv_{\mathcal{T}} [u]$$

The new theory $(\mathcal{S}, \cong)$ can inherit **good properties** from $\mathcal{T}$:

Models as Syntactic Translations

A translation induces a new definitional equality on $\mathcal{S}$ that extends $\equiv_{\mathcal{S}}$

$$t \cong u \qquad \Leftrightarrow \qquad [t] \equiv_{\mathcal{T}} [u]$$

The new theory $(\mathcal{S}, \cong)$ can inherit **good properties** from $\mathcal{T}$:

$\mathcal{T} \nvdash [\bot] \qquad \Rightarrow \qquad \mathcal{S}$ and $(\mathcal{S}, \cong)$ are consistent

Models as Syntactic Translations

A translation induces a new definitional equality on $\mathcal{S}$ that extends $\equiv_{\mathcal{S}}$

$$t \cong u \qquad \Leftrightarrow \qquad [t] \equiv_{\mathcal{T}} [u]$$

The new theory $(\mathcal{S}, \cong)$ can inherit **good properties** from $\mathcal{T}$:

$\mathcal{T} \nvdash [\bot]$ $\quad\Rightarrow\quad$ $\mathcal{S}$ and $(\mathcal{S}, \cong)$ are consistent

$\mathcal{T}$ satisfies canonicity for $[\mathbb{N}]$ $\quad\Rightarrow\quad$ $(\mathcal{S}, \cong)$ satisfies canonicity

Models as Syntactic Translations

A translation induces a new definitional equality on $\mathcal{S}$ that extends $\equiv_{\mathcal{S}}$

$$t \cong u \qquad \Leftrightarrow \qquad [t] \equiv_{\mathcal{T}} [u]$$

The new theory $(\mathcal{S}, \cong)$ can inherit **good properties** from $\mathcal{T}$:

$\mathcal{T} \nvdash [\bot]$	$\Rightarrow$	$\mathcal{S}$ and $(\mathcal{S}, \cong)$ are consistent
$\mathcal{T}$ satisfies canonicity for $[\mathbb{N}]$	$\Rightarrow$	$(\mathcal{S}, \cong)$ satisfies canonicity
$\mathcal{T}$ has decidable type-checking	$\Rightarrow$	$(\mathcal{S}, \cong)$ has decidable type-checking

Models as Syntactic Translations

A translation induces a new definitional equality on $\mathcal{S}$ that extends $\equiv_{\mathcal{S}}$

$$t \cong u \qquad \Leftrightarrow \qquad [t] \equiv_{\mathcal{T}} [u]$$

The new theory $(\mathcal{S}, \cong)$ can inherit **good properties** from $\mathcal{T}$:

$\mathcal{T} \nvdash [\bot]$	$\Rightarrow$	$\mathcal{S}$ and $(\mathcal{S}, \cong)$ are consistent
$\mathcal{T}$ satisfies canonicity for $[\mathbb{N}]$	$\Rightarrow$	$(\mathcal{S}, \cong)$ satisfies canonicity
$\mathcal{T}$ has decidable type-checking	$\Rightarrow$	$(\mathcal{S}, \cong)$ has decidable type-checking

"$(\mathcal{S}, \cong)$ computes if $\mathcal{T}$ computes"

Models as Syntactic Translations

A translation induces a new definitional equality on $\mathcal{S}$ that extends $\equiv_{\mathcal{S}}$

$$t \cong u \qquad \Leftrightarrow \qquad [t] \equiv_{\mathcal{T}} [u]$$

The new theory $(\mathcal{S}, \cong)$ can inherit **good properties** from $\mathcal{T}$:

$$\begin{array}{lcl}
\mathcal{T} \nvdash [\bot] & \Rightarrow & \mathcal{S} \text{ and } (\mathcal{S}, \cong) \text{ are consistent} \\
\mathcal{T} \text{ satisfies canonicity for } [\mathbb{N}] & \Rightarrow & (\mathcal{S}, \cong) \text{ satisfies canonicity} \\
\mathcal{T} \text{ has decidable type-checking} & \Rightarrow & (\mathcal{S}, \cong) \text{ has decidable type-checking}
\end{array}$$

"$(\mathcal{S}, \cong)$ computes if $\mathcal{T}$ computes"

Note that $(\mathcal{S}, \cong)$ does not have a directed reduction, though.

Models as Syntactic Translations

This is enough motivation for writing the cubical model as a translation from HoTT to MLTT.

Models as Syntactic Translations

This is enough motivation for writing the cubical model as a translation from HoTT to MLTT.

The naive approach to write such a translation **fails**, however.

Models as Syntactic Translations

This is enough motivation for writing the cubical model as a translation from HoTT to MLTT.

The naive approach to write such a translation **fails**, however.
Translation must preserve definitional equality, in particular reduction:

$$[A\{x \leftarrow t\}] \equiv [A]\{x \leftarrow [t]\}$$

But crucial properties such as functoriality of presheaves are stated with propositional equality, which does not imply conversion.

Models as Syntactic Translations

This is enough motivation for writing the cubical model as a translation from HoTT to MLTT.

The naive approach to write such a translation **fails**, however.
Translation must preserve definitional equality, in particular reduction:

$$[A\{x \leftarrow t\}] \equiv [A]\{x \leftarrow [t]\}$$

But crucial properties such as functoriality of presheaves are stated with propositional equality, which does not imply conversion.
(unless we are dealing with an extensional target, but then computation breaks)

Definitional Presheaves

Definitional Presheaves

This failure happens when we try to write the usual definition of a presheaf in intensional type theory.

Claim: in ITT, this is **not the right definition** of a presheaf.

Definitional Presheaves

This failure happens when we try to write the usual definition of a presheaf in intensional type theory.

Claim: in ITT, this is **not the right definition** of a presheaf.
Assuming a definitional category $\mathbb{C}$, one can define

Definitional Presheaves

This failure happens when we try to write the usual definition of a presheaf in intensional type theory.

Claim: in ITT, this is **not the right definition** of a presheaf.
Assuming a definitional category $\mathbb{C}$, one can define

$$\mathrm{Presheaf} \; : \mathcal{U}$$

Definitional Presheaves

This failure happens when we try to write the usual definition of a presheaf in intensional type theory.

Claim: in ITT, this is **not the right definition** of a presheaf.
Assuming a definitional category C, one can define

$$\mathrm{Presheaf} \;:\; \mathcal{U} \qquad \mathrm{Elem} \;:\; \mathrm{Presheaf} \to \mathrm{C}_0 \to \mathcal{U}$$

Definitional Presheaves

This failure happens when we try to write the usual definition of a presheaf in intensional type theory.

Claim: in ITT, this is **not the right definition** of a presheaf.
Assuming a definitional category C, one can define

$$\mathrm{Presheaf} : \mathcal{U} \qquad \mathrm{Elem} : \mathrm{Presheaf} \to \mathrm{C}_0 \to \mathcal{U}$$

$$\begin{array}{c} e : \mathrm{Elem}\ F\ c \\ \alpha : \mathcal{C}_1(d, c) \end{array} \quad \vdash \quad e|_\alpha : \mathrm{Elem}\ F\ d$$

Definitional Presheaves

This failure happens when we try to write the usual definition of a presheaf in intensional type theory.

Claim: in ITT, this is **not the right definition** of a presheaf.
Assuming a definitional category C, one can define

$$\mathrm{Presheaf} \; : \mathcal{U} \qquad \mathrm{Elem} \; : \; \mathrm{Presheaf} \to \mathrm{C}_0 \to \mathcal{U}$$

$$\begin{array}{c} e : \mathrm{Elem}\; F\; c \\ \alpha : \mathcal{C}_1(d, c) \end{array} \quad \vdash \quad e|_\alpha : \mathrm{Elem}\; F\; d$$

$$e|_\alpha|_\beta \equiv e|_{\alpha\circ\beta}$$

Definitional Presheaves

This failure happens when we try to write the usual definition of a presheaf in intensional type theory.

Claim: in ITT, this is **not the right definition** of a presheaf.
Assuming a definitional category C, one can define

$$\mathrm{Presheaf} \ : \mathcal{U} \qquad \mathrm{Elem} \ : \ \mathrm{Presheaf} \to \mathrm{C}_0 \to \mathcal{U}$$

$$\begin{array}{c} e : \mathrm{Elem}\ F\ c \\ \alpha : \mathcal{C}_1(d, c) \end{array} \quad \vdash \quad e|_\alpha : \mathrm{Elem}\ F\ d$$

$$e|_\alpha|_\beta \equiv e|_{\alpha \circ \beta}$$

$$\mathrm{NatTransfo} \ : \ \mathrm{Presheaf} \to \mathrm{Presheaf} \to \mathcal{U}$$

This failure happens when we try to write the usual definition of a presheaf in intensional type theory.

Claim: in ITT, this is **not the right definition** of a presheaf.
Assuming a definitional category C, one can define

$$\mathrm{Presheaf} : \mathcal{U} \qquad \mathrm{Elem} : \mathrm{Presheaf} \to \mathrm{C}_0 \to \mathcal{U}$$

$$\begin{matrix} e : \mathrm{Elem}\ F\ c \\ \alpha : \mathcal{C}_1(d, c) \end{matrix} \quad \vdash \quad e|_\alpha : \mathrm{Elem}\ F\ d$$

$$e|_\alpha|_\beta \equiv e|_{\alpha \circ \beta}$$

$$\mathrm{NatTransfo} : \mathrm{Presheaf} \to \mathrm{Presheaf} \to \mathcal{U}$$

$$\mathrm{Eval} : \mathrm{NatTransfo}\ F\ G \to \mathrm{Elem}\ F\ c \to \mathrm{Elem}\ G\ c$$

Definitional Presheaves

This failure happens when we try to write the usual definition of a presheaf in intensional type theory.

Claim: in ITT, this is **not the right definition** of a presheaf.
Assuming a definitional category C, one can define

$$\mathrm{Presheaf} \,:\, \mathcal{U} \qquad \mathrm{Elem} \,:\, \mathrm{Presheaf} \to \mathrm{C}_0 \to \mathcal{U}$$

$$\begin{matrix} e : \mathrm{Elem}\ F\ c \\ \alpha : \mathcal{C}_1(d,c) \end{matrix} \quad \vdash \quad e|_\alpha : \mathrm{Elem}\ F\ d$$

$$e|_\alpha|_\beta \equiv e|_{\alpha\circ\beta}$$

$$\mathrm{NatTransfo} \,:\, \mathrm{Presheaf} \to \mathrm{Presheaf} \to \mathcal{U}$$

$$\mathrm{Eval} \,:\, \mathrm{NatTransfo}\ F\ G \to \mathrm{Elem}\ F\ c \to \mathrm{Elem}\ G\ c$$

$$(\mathrm{Eval}\ \theta\ e)|_\alpha \equiv \mathrm{Eval}\ \theta\ e|_\alpha$$

Definitional Presheaves

Using this construction, Pédrot managed to write presheaf models of MLTT as translations to **MLTT+SProp+UIP** (all types are definitional hsets).

Definitional Presheaves

Using this construction, Pédrot managed to write presheaf models of MLTT as translations to **MLTT+SProp+UIP** (all types are definitional hsets).

This theory is not as well understood as MLTT, but it does have a definite computational behaviour.

Definitional Presheaves

Using this construction, Pédrot managed to write presheaf models of MLTT as translations to **MLTT+SProp+UIP** (all types are definitional hsets).

This theory is not as well understood as MLTT, but it does have a definite computational behaviour.

Our goal is to do the same with *fibrant* cubical presheaves, to get a translation from HoTT to MLTT+SProp+UIP.

Definitional Presheaves

What has been done, what remains to be done

Dependent Products	✓
Dependent Sums	✓
Booleans	✓
Cubical equality	✓
Function extensionality	✓
J eliminator	✓
Weak univalence	✓
Full univalence	**?**
Fibrancy of the universe	**?**
Higher Inductive Types	**?**

Meta-theoretical Consequences

What we (would) get:

- a formal, *computational* proof of consistency for HoTT
-
-

Meta-theoretical Consequences

What we (would) get:

- a formal, *computational* proof of consistency for HoTT
- a new way to compute with univalence
-

Meta-theoretical Consequences

What we (would) get:

- a formal, *computational* proof of consistency for HoTT
- a new way to compute with univalence
- $\mathrm{MLTT} + \mathrm{UIP} + \mathrm{SProp}$ has good properties $\Rightarrow$ $(\mathrm{HoTT}, \cong)$ has good properties.

Meta-theoretical Consequences

What we (would) get:

- a formal, *computational* proof of consistency for HoTT
- a new way to compute with univalence
- $\mathrm{MLTT} + \mathrm{UIP} + \mathrm{SProp}$ has good properties $\Rightarrow$ $(\mathrm{HoTT}, \cong)$ has good properties.

This is nice, but we already knew that "Cubical type theory has good properties $\Rightarrow$ $(\mathrm{HoTT}, \cong')$ has good properties."

Meta-theoretical Consequences

What we (would) get:

- a formal, *computational* proof of consistency for HoTT
- a new way to compute with univalence
- $\mathrm{MLTT} + \mathrm{UIP} + \mathrm{SProp}$ has good properties $\Rightarrow (\mathrm{HoTT}, \cong)$ has good properties.

This is nice, but we already knew that "Cubical type theory has good properties $\Rightarrow (\mathrm{HoTT}, \cong')$ has good properties."

This method really shines when it comes to extensions of HoTT:

- Good properties for 2-level type theory with little extra work
- You want computational simplicial types? Just define them in the model using strict equality, prove their fibrancy, and voilà!
- etc

Thank you!