A Characterization of Univalent Fibrations

Dan Christensen
University of Western Ontario

CT2015, Aveiro, June 18, 2015

Outline:

- Background on type theory
- Equivalence and univalence
- A characterization of univalent fibrations

Background on Type Theory

Type theory is a logical system in which the basic objects are called types.

Initially, types were thought of as sets, but we will think of them as being like spaces.

Background on Type Theory

Type theory is a logical system in which the basic objects are called types.

Initially, types were thought of as sets, but we will think of them as being like spaces.

We write $x : X$ to indicate that x is a term of type X, which is analogous to the set-theoretic statement $x \in X$.

We assume given a universe type `Type`, and therefore can write $X : \texttt{Type}$ to indicate that X is a type.

Each term has a unique type, so we can't directly talk about intersections, unions, etc. Instead, type theory comes with type constructors that correspond to common constructions in mathematics.

Type Constructors: Function types

For any two types A and B, there is a function type denoted $A \to B$, which should be thought of as an internal hom B^A.

Type Constructors: Function types

For any two types A and B, there is a function type denoted $A \to B$, which should be thought of as an internal hom B^A.

If $f(a)$ is an expression of type B whenever a is of type A, then $\lambda a.f(a)$ denotes the function $A \to B$ sending a to $f(a)$.

Type Constructors: Function types

For any two types A and B, there is a function type denoted $A \to B$, which should be thought of as an internal hom B^A.

If $f(a)$ is an expression of type B whenever a is of type A, then $\lambda a.f(a)$ denotes the function $A \to B$ sending a to $f(a)$.

Examples:

- The identity function $\mathtt{id}_A$ is defined to be $\lambda a.a$.
- The constant function sending everything in A to $b : B$ is $\lambda a.b$.

Type Constructors: Function types

For any two types A and B, there is a function type denoted $A \to B$, which should be thought of as an internal hom B^A.

If $f(a)$ is an expression of type B whenever a is of type A, then $\lambda a.f(a)$ denotes the function $A \to B$ sending a to $f(a)$.

Examples:

- The identity function $\mathtt{id}_A$ is defined to be $\lambda a.a$.
- The constant function sending everything in A to $b : B$ is $\lambda a.b$.
- Given functions $f : A \to B$ and $g : B \to C$, their composite gf is $\lambda a.g(f(a))$.

Type Constructors: Coproduct

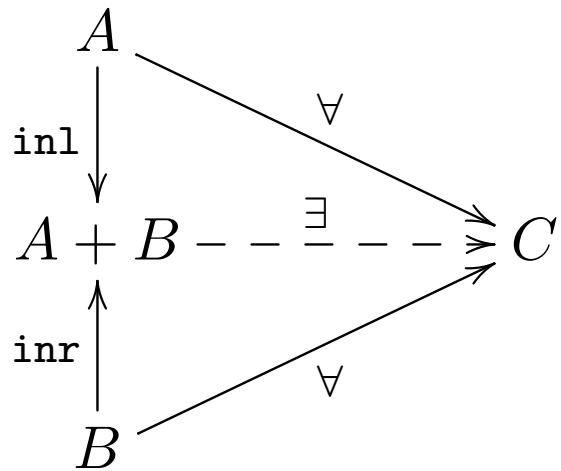
Most constructions in type theory are defined inductively.

Type Constructors: Coproduct

Most constructions in type theory are defined inductively.

For example, given types A and B, there is another type $A + B$ which is generated by terms of the form `inl` a and `inr` b.

"Generated" means that it satisfies a weak universal property:

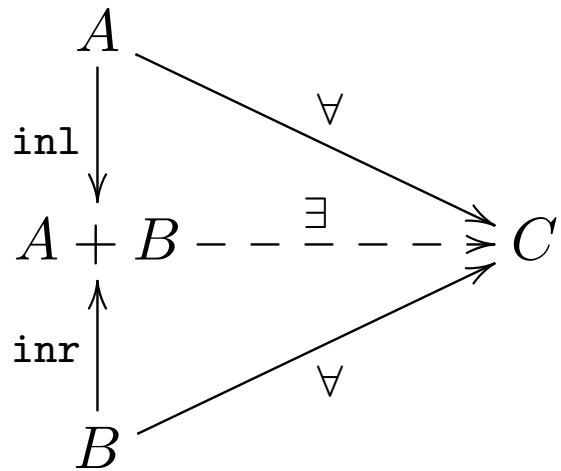


Type Constructors: Coproduct

Most constructions in type theory are defined inductively.

For example, given types A and B, there is another type $A + B$ which is generated by terms of the form `inl` a and `inr` b.

"Generated" means that it satisfies a weak universal property:



(Using dependent types and identity types, one can prove uniqueness.)

Type Constructors: $\varnothing$, `unit`, $\times$, $\mathbb{N}$

There are similar definitions of:

- the empty type $\varnothing$ as a weakly initial object ("free on no generators"),
- the one point type `unit` to be "free on one generator",
- the product $A \times B$ of two types, which is generated by all pairs (a, b), and
- the natural numbers $\mathbb{N}$, generated by $0 : \mathbb{N}$ and $\texttt{succ} : \mathbb{N} \to \mathbb{N}$.

Note the preference for constructions defined by mapping out.

Dependent Types

The above structure is enough to construct types that depend on terms of other types.

These dependent types are one of the key ideas in Martin-Löf type theory, and will play a central role in this talk.

Dependent Types

The above structure is enough to construct types that depend on terms of other types.

These dependent types are one of the key ideas in Martin-Löf type theory, and will play a central role in this talk.

Examples:

$$\lambda a.B : A \longrightarrow \texttt{Type} \quad \text{(a constant type family)}$$
$$\lambda n.A^n : \mathbb{N} \longrightarrow \texttt{Type} \quad \text{(defined inductively)}$$
$$\lambda(A, B).\, A + B : \texttt{Type} \times \texttt{Type} \longrightarrow \texttt{Type}$$

Dependent Sums

Dependent sums are like the Grothendieck construction.

Given a type family $B : A \to \texttt{Type}$, the dependent sum $\sum_{a:A} B(a)$ is freely generated by pairs (a, b) with $b : B(a)$.

Dependent Sums

Dependent sums are like the Grothendieck construction.

Given a type family $B : A \to \mathtt{Type}$, the dependent sum $\sum_{a:A} B(a)$ is freely generated by pairs (a, b) with $b : B(a)$.

The dependent sum has a projection map

$$\mathtt{pr}_1 : \sum_{a:A} B(a) \longrightarrow A$$

sending (a, b) to a, and this can be regarded as a fibration.

Dependent Sums

Dependent sums are like the Grothendieck construction.

Given a type family $B : A \to \mathtt{Type}$, the dependent sum $\sum_{a:A} B(a)$ is freely generated by pairs (a, b) with $b : B(a)$.

The dependent sum has a projection map

$$\mathtt{pr}_1 : \sum_{a:A} B(a) \longrightarrow A$$

sending (a, b) to a, and this can be regarded as a fibration.

There is also a dependent product $\prod_{x:A} B(x)$. A term is a function f sending each $x : A$ to an $f(x) : B(x)$.

$\prod_{a:A} B(a)$ can also be thought of as the space of sections of $\mathtt{pr}_1$.

Propositions as Types: Curry-Howard

A type can be thought of as a proposition, which is true when inhabited:

Types	$\longleftrightarrow$	Propositions
$\varnothing$	$\longleftrightarrow$	false
`unit`	$\longleftrightarrow$	true
$P \times Q$	$\longleftrightarrow$	P and Q
$P + Q$	$\longleftrightarrow$	P or Q
$P \to Q$	$\longleftrightarrow$	P implies Q
$\prod_{x:A} P(x)$	$\longleftrightarrow$	$\forall x P(x)$
$\sum_{x:A} P(x)$	$\longleftrightarrow$	$\exists x P(x)$

Propositions as Types: Curry-Howard

A type can be thought of as a proposition, which is true when inhabited:

$$
\begin{array}{rcl}
\text{Types} & \longleftrightarrow & \text{Propositions} \\
\\
\varnothing & \longleftrightarrow & \text{false} \\
\texttt{unit} & \longleftrightarrow & \text{true} \\
P \times Q & \longleftrightarrow & P \text{ and } Q \\
P + Q & \longleftrightarrow & P \text{ or } Q \\
P \to Q & \longleftrightarrow & P \text{ implies } Q \\
\displaystyle\prod_{x:A} P(x) & \longleftrightarrow & \forall x P(x) \\
\displaystyle\sum_{x:A} P(x) & \longleftrightarrow & \exists x P(x)
\end{array}
$$

But what about the proposition $a = b$?

Identity Types

Given a type A, the identity type of A is a type family $A \times A \to \texttt{Type}$ whose values are written $a = b$ for $a, b : A$.

This type family is generated by "reflexivity" terms of the form $\texttt{refl}_a : a = a$ for each $a : A$.

Identity Types

Given a type A, the identity type of A is a type family $A \times A \to \mathtt{Type}$ whose values are written $a = b$ for $a, b : A$.

This type family is generated by "reflexivity" terms of the form $\mathtt{refl}_a : a = a$ for each $a : A$.

A term of type $a = b$ was historically thought of as the assertion that a equals b, but in our interpretation should be thought of as a path from a to b in A, with $\mathtt{refl}_a$ being the constant path at a.

Identity Types

Given a type A, the identity type of A is a type family $A \times A \to \mathtt{Type}$ whose values are written $a = b$ for $a, b : A$.

This type family is generated by "reflexivity" terms of the form $\mathtt{refl}_a : a = a$ for each $a : A$.

A term of type $a = b$ was historically thought of as the assertion that a equals b, but in our interpretation should be thought of as a path from a to b in A, with $\mathtt{refl}_a$ being the constant path at a.

The associated map $\sum_{a,b:A} (a = b) \longrightarrow A \times A$ was historically thought of as the diagonal map $A \to A \times A$, but for our purposes it better regarded as the path fibration $A^I \to A \times A$, which is obtained by replacing the diagonal map by a fibration.

Identity Types

Given a type A, the identity type of A is a type family $A \times A \to \mathtt{Type}$ whose values are written $a = b$ for $a, b : A$.

This type family is generated by "reflexivity" terms of the form $\mathtt{refl}_a : a = a$ for each $a : A$.

A term of type $a = b$ was historically thought of as the assertion that a equals b, but in our interpretation should be thought of as a path from a to b in A, with $\mathtt{refl}_a$ being the constant path at a.

The associated map $\sum_{a,b:A} (a = b) \longrightarrow A \times A$ was historically thought of as the diagonal map $A \to A \times A$, but for our purposes it better regarded as the path fibration $A^I \to A \times A$, which is obtained by replacing the diagonal map by a fibration.

Since $a = b$ is a type, *it* has an associated path type $p = q$ for $p, q : a = b$. It is not always the case that $p = q$.

Equivalences

We say that $f : A \to B$ is an equivalence if it has left and right inverses. That is,

$$\texttt{IsEquiv}f :\equiv \left(\sum_{g:B\to A} (gf = \texttt{id}_A) \right) \times \left(\sum_{h:B\to A} (fh = \texttt{id}_B) \right).$$

Equivalences

We say that $f : A \to B$ is an equivalence if it has left and right inverses. That is,

$$\texttt{IsEquiv}f \;:\equiv\; \left(\sum_{g:B\to A} (gf = \texttt{id}_A)\right) \times \left(\sum_{h:B\to A} (fh = \texttt{id}_B)\right).$$

The type of equivalences from A to B is

$$A \simeq B \;:\equiv\; \sum_{f:A\to B} \texttt{IsEquiv}f.$$

Univalence Axiom

For types A and B, we define functions $\omega : (A = B) \to (A \simeq B)$ by sending $\mathtt{refl}_A$ to $\mathtt{id}_A$.

Univalence Axiom

For types A and B, we define functions $\omega : (A = B) \to (A \simeq B)$ by sending $\mathtt{refl}_A$ to $\mathtt{id}_A$.

The Univalence Axiom says that ω is an equivalence for all types A and B.

Univalence Axiom

For types A and B, we define functions $\omega : (A = B) \to (A \simeq B)$ by sending $\mathtt{refl}_A$ to $\mathtt{id}_A$.

The Univalence Axiom says that ω is an equivalence for all types A and B.

If ω is an equivalence, then there is an inverse map

$$(A \simeq B) \longrightarrow (A = B)$$

which implies that equivalent types are equal.

Univalence Axiom

For types A and B, we define functions $\omega : (A = B) \to (A \simeq B)$ by sending $\mathtt{refl}_A$ to $\mathtt{id}_A$.

The Univalence Axiom says that ω is an equivalence for all types A and B.

If ω is an equivalence, then there is an inverse map

$$(A \simeq B) \longrightarrow (A = B)$$

which implies that equivalent types are equal.

This is an assertion about the universe `Type`, and it does not hold in the standard set-theoretic model.

But it does hold for a model in simplicial sets [Kapulkin-Lumsdaine-Voevodsky] and it allows one to prove many homotopical results.

Univalent type families

We say that a type family $B : A \to \mathtt{Type}$ is univalent if the composite map

$$(a = a') \xrightarrow{\mathtt{ap}\ B} (B(a) = B(a')) \xrightarrow{\omega} (B(a) \simeq B(a'))$$

is an equivalence, where $\mathtt{ap}\ B$ sends $\mathtt{refl}_a$ to $\mathtt{refl}_{B(a)}$.

Univalent type families

We say that a type family $B : A \to \mathtt{Type}$ is univalent if the composite map

$$(a = a') \xrightarrow{\mathtt{ap}\ B} (B(a) = B(a')) \xrightarrow{\omega} (B(a) \simeq B(a'))$$

is an equivalence, where $\mathtt{ap}\ B$ sends $\mathtt{refl}_a$ to $\mathtt{refl}_{B(a)}$.

The Univalence Axiom is the special case where the type family is $\mathtt{id}_{\mathtt{Type}} : \mathtt{Type} \to \mathtt{Type}$.

Univalent type families

We say that a type family $B : A \to \mathtt{Type}$ is univalent if the composite map

$$(a = a') \xrightarrow{\mathtt{ap}\ B} (B(a) = B(a')) \xrightarrow{\omega} (B(a) \simeq B(a'))$$

is an equivalence, where $\mathtt{ap}\ B$ sends $\mathtt{refl}_a$ to $\mathtt{refl}_{B(a)}$.

The Univalence Axiom is the special case where the type family is $\mathtt{id}_{\mathtt{Type}} : \mathtt{Type} \to \mathtt{Type}$.

Our goal is to characterize the univalent type families, as a way to better understand the Univalence Axiom.

The Case of Simplicial Sets

In the simplicial model, the notions of homotopy and equivalence match the standard notions, and a type family corresponds to a Kan fibration, obtained using the dependent sum construction.

The Case of Simplicial Sets

In the simplicial model, the notions of homotopy and equivalence match the standard notions, and a type family corresponds to a Kan fibration, obtained using the dependent sum construction.

Associated to a Kan fibration $E \to B$ is a fibration $Eq(E) \to B \times B$ whose fibre over (b, b') is the simplicial set of equivalences from the fibre E_b to the fibre $E_{b'}$.

The Case of Simplicial Sets

In the simplicial model, the notions of homotopy and equivalence match the standard notions, and a type family corresponds to a Kan fibration, obtained using the dependent sum construction.

Associated to a Kan fibration $E \to B$ is a fibration $Eq(E) \to B \times B$ whose fibre over (b, b') is the simplicial set of equivalences from the fibre E_b to the fibre $E_{b'}$.

The original fibration is univalent iff a certain natural map from the path space B^I to $Eq(E)$ is an equivalence.

Examples:

- The identity map $X \to X$ is univalent iff X is empty or contractible.
- The double-cover $S^\infty \to \mathbb{R}P^\infty$ is univalent.

$\mathtt{BAut}(F)$

We next construct a univalent type family associated to any type F. Define

$$\mathtt{BAut}(F) :\equiv \sum_{Z:\mathtt{Type}} |F \simeq Z|,$$

where $|P|$ denotes the propositional truncation of the type P. This is a type which is either empty or contractible, and has the same truth value as P.

Note that $\mathtt{BAut}(F)$ has a natural basepoint $(F, \mathtt{id}_F)$.

$\mathtt{BAut}(F)$

We next construct a univalent type family associated to any type F. Define

$$\mathtt{BAut}(F) :\equiv \sum_{Z:\mathtt{Type}} |F \simeq Z|,$$

where $|P|$ denotes the propositional truncation of the type P. This is a type which is either empty or contractible, and has the same truth value as P.

Note that $\mathtt{BAut}(F)$ has a natural basepoint $(F, \mathtt{id}_F)$.

Define

$$\mathtt{Aut}(F) :\equiv (F \simeq F),$$

the monoid of self-equivalences of F.

$\mathtt{BAut}(F)$

We next construct a univalent type family associated to any type F. Define

$$\mathtt{BAut}(F) :\equiv \sum_{Z:\mathtt{Type}} |F \simeq Z|,$$

where $|P|$ denotes the propositional truncation of the type P. This is a type which is either empty or contractible, and has the same truth value as P.

Note that $\mathtt{BAut}(F)$ has a natural basepoint $(F, \mathtt{id}_F)$.

Define

$$\mathtt{Aut}(F) :\equiv (F \simeq F),$$

the monoid of self-equivalences of F.

Lemma (Easy)**.** $\Omega\mathtt{BAut}(F) \simeq \mathtt{Aut}(F)$, where $\Omega X :\equiv (x_0 = x_0)$.

$\mathtt{BAut}(F)$

We next construct a univalent type family associated to any type F. Define

$$\mathtt{BAut}(F) :\equiv \sum_{Z:\mathtt{Type}} |F \simeq Z|,$$

where $|P|$ denotes the propositional truncation of the type P. This is a type which is either empty or contractible, and has the same truth value as P.

Note that $\mathtt{BAut}(F)$ has a natural basepoint $(F, \mathtt{id}_F)$.

Define

$$\mathtt{Aut}(F) :\equiv (F \simeq F),$$

the monoid of self-equivalences of F.

Lemma (Easy)**.** $\Omega\mathtt{BAut}(F) \simeq \mathtt{Aut}(F)$, where $\Omega X :\equiv (x_0 = x_0)$.

I don't know how to define BG in general, but the above implies that this gives the correct result in a model.

The universal fibration with fibre F

Since $\mathtt{BAut}(F) :\equiv \sum_{Z:\mathtt{Type}} |F \simeq Z|$, we have a canonical map

$$\alpha :\equiv \mathtt{pr}_1 : \mathtt{BAut}(F) \longrightarrow \mathtt{Type}.$$

The universal fibration with fibre F

Since $\mathtt{BAut}(F) :\equiv \sum_{Z:\mathtt{Type}} |F \simeq Z|$, we have a canonical map

$$\alpha :\equiv \mathtt{pr}_1 : \mathtt{BAut}(F) \longrightarrow \mathtt{Type}.$$

Proposition. The type family α is univalent for any type F.

A somewhat technical argument was sketched by Moerdijk for simplicial sets in a talk. It is proved for ∞-topoi by Gepner and J. Kock. In fact, the proposition is easy to prove in type theory (see the HoTT library), and implies the result in simplicial sets and other models.

The universal fibration with fibre F

Since $\mathtt{BAut}(F) :\equiv \sum_{Z:\mathtt{Type}} |F \simeq Z|$, we have a canonical map

$$\alpha :\equiv \mathtt{pr}_1 : \mathtt{BAut}(F) \longrightarrow \mathtt{Type}.$$

Proposition. The type family α is univalent for any type F.

A somewhat technical argument was sketched by Moerdijk for simplicial sets in a talk. It is proved for ∞-topoi by Gepner and J. Kock. In fact, the proposition is easy to prove in type theory (see the HoTT library), and implies the result in simplicial sets and other models.

Consider the associated fibration $\sum_{(Z,e):\mathtt{BAut}(F)} Z \to \mathtt{BAut}(F)$.

The universal fibration with fibre F

Since $\mathtt{BAut}(F) :\equiv \sum_{Z:\mathtt{Type}} |F \simeq Z|$, we have a canonical map

$$\alpha :\equiv \mathtt{pr}_1 : \mathtt{BAut}(F) \longrightarrow \mathtt{Type}.$$

Proposition. The type family α is univalent for any type F.

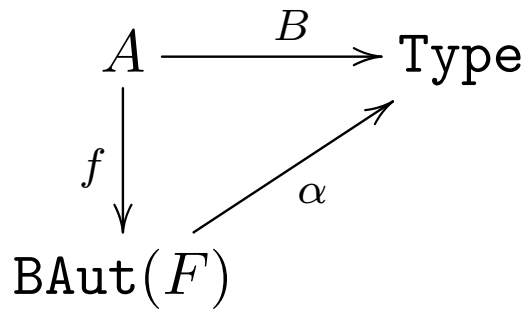
A somewhat technical argument was sketched by Moerdijk for simplicial sets in a talk. It is proved for ∞-topoi by Gepner and J. Kock. In fact, the proposition is easy to prove in type theory (see the HoTT library), and implies the result in simplicial sets and other models.

Consider the associated fibration $\sum_{(Z,e):\mathtt{BAut}(F)} Z \to \mathtt{BAut}(F)$.

Taking $F = \mathtt{unit}$ gives the identity map on a contractible space, and taking $F = \mathtt{unit} + \mathtt{unit}$ gives $S^\infty \to \mathbb{R}P^\infty$.

A Characterization of Univalent Fibrations

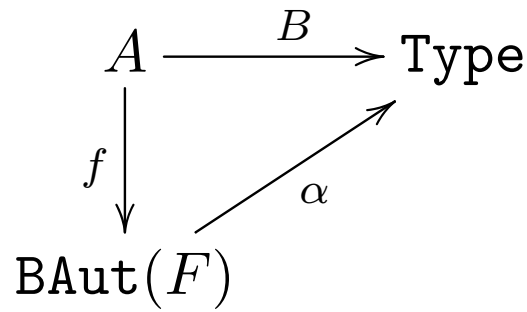
Theorem (C)**.** If $B : A \to \mathtt{Type}$ is a univalent type family and A is connected, then there is a type F and an equivalence $f : A \simeq \mathtt{BAut}(F)$ such that



commutes up to homotopy.

A Characterization of Univalent Fibrations

Theorem (C)**.** If $B : A \to \mathtt{Type}$ is a univalent type family and A is connected, then there is a type F and an equivalence $f : A \simeq \mathtt{BAut}(F)$ such that



commutes up to homotopy.

More generally, if A is not connected, then it is a coproduct of a set-indexed family of $\mathtt{BAut}(F)$'s, with pairwise non-equivalent F's, with a similar commuting diagram.

The case of the empty coproduct gives $\varnothing \to \varnothing$, our other example.

Consequences

- Univalent fibrations are exactly the classifying bundles for fibrations with fibre F (or an appropriate coproduct of such). This is a notion from the 1950's that is extremely well-studied in topology.

Consequences

- Univalent fibrations are exactly the classifying bundles for fibrations with fibre F (or an appropriate coproduct of such). This is a notion from the 1950's that is extremely well-studied in topology.
- If the Univalence Axiom holds, then $\mathtt{id}_{\mathtt{Type}} : \mathtt{Type} \to \mathtt{Type}$ is a univalent type family, so $\mathtt{Type}$ is equivalent to $\sum \mathtt{BAut}(F)$, where the sum is over equivalence classes of $F : \mathtt{Type}$.
  We can also describe the total space of the universal fibration over $\mathtt{Type}$.

Consequences

- Univalent fibrations are exactly the classifying bundles for fibrations with fibre F (or an appropriate coproduct of such). This is a notion from the 1950's that is extremely well-studied in topology.
- If the Univalence Axiom holds, then $\mathtt{id}_{\mathtt{Type}} : \mathtt{Type} \to \mathtt{Type}$ is a univalent type family, so $\mathtt{Type}$ is equivalent to $\sum \mathtt{BAut}(F)$, where the sum is over equivalence classes of $F : \mathtt{Type}$.
  We can also describe the total space of the universal fibration over $\mathtt{Type}$.
- A given type A is rarely the base of a univalent fibration, and when it is, it is usually the base of only one univalent fibration up to equivalence. However, coincidences can occur. E.g. taking F to be $\mathtt{unit}$ gives $\mathtt{unit} \to \mathtt{unit}$, but taking F to be $\varnothing$ gives $\varnothing \to \mathtt{unit}$, so both are univalent.

Consequences

- Univalent fibrations are exactly the classifying bundles for fibrations with fibre F (or an appropriate coproduct of such). This is a notion from the 1950's that is extremely well-studied in topology.
- If the Univalence Axiom holds, then $\mathtt{id}_{\mathtt{Type}} : \mathtt{Type} \to \mathtt{Type}$ is a univalent type family, so $\mathtt{Type}$ is equivalent to $\sum \mathtt{BAut}(F)$, where the sum is over equivalence classes of $F : \mathtt{Type}$.
  We can also describe the total space of the universal fibration over $\mathtt{Type}$.
- A given type A is rarely the base of a univalent fibration, and when it is, it is usually the base of only one univalent fibration up to equivalence. However, coincidences can occur. E.g. taking F to be $\mathtt{unit}$ gives $\mathtt{unit} \to \mathtt{unit}$, but taking F to be $\varnothing$ gives $\varnothing \to \mathtt{unit}$, so both are univalent.

  I know of no other coincidences!

About the Proof

It's almost tautologous that a univalent family $B : A \to \texttt{Type}$ factors through $\texttt{BAut}(F)$ when B is connected. But there is some work in showing that the map $A \to \texttt{BAut}(F)$ is an equivalence.

About the Proof

It's almost tautologous that a univalent family $B : A \to \texttt{Type}$ factors through $\texttt{BAut}(F)$ when B is connected. But there is some work in showing that the map $A \to \texttt{BAut}(F)$ is an equivalence.

The proof has been formalized in `Coq`.

About the Proof

It's almost tautologous that a univalent family $B : A \to \texttt{Type}$ factors through $\texttt{BAut}(F)$ when B is connected. But there is some work in showing that the map $A \to \texttt{BAut}(F)$ is an equivalence.

The proof has been formalized in `Coq`.

I also have a more complicated proof for simplicial sets, that doesn't use a universe or univalence. This could possibly give another way to prove that simplicial sets has a univalent universe, but there are non-trivial issues of strictness.

About the Proof

It's almost tautologous that a univalent family $B : A \to \texttt{Type}$ factors through $\texttt{BAut}(F)$ when B is connected. But there is some work in showing that the map $A \to \texttt{BAut}(F)$ is an equivalence.

The proof has been formalized in `Coq`.

I also have a more complicated proof for simplicial sets, that doesn't use a universe or univalence. This could possibly give another way to prove that simplicial sets has a univalent universe, but there are non-trivial issues of strictness.

To learn more about homotopy type theory: These slides and a longer intro to type theory are on my web site.

Mike Shulman's slides from two series of lectures are great.

Thanks for listening!

About the Proof

It's almost tautologous that a univalent family $B : A \to \texttt{Type}$ factors through $\texttt{BAut}(F)$ when B is connected. But there is some work in showing that the map $A \to \texttt{BAut}(F)$ is an equivalence.

The proof has been formalized in `Coq`.

I also have a more complicated proof for simplicial sets, that doesn't use a universe or univalence. This could possibly give another way to prove that simplicial sets has a univalent universe, but there are non-trivial issues of strictness.

To learn more about homotopy type theory: These slides and a longer intro to type theory are on my web site.

Mike Shulman's slides from two series of lectures are great.

And sorry for going over time!