The univalence axiom, a brief & incomplete tour

School and Workshop on Univalent Mathematics

tslil clingman, 2019

Univalence & function extensionality

Univalence, the structure-identity principle, and you

Open questions the formulation of univalence

Equivalent reformulations

Notation

By way of warming up, let us fix notation.

Notation

By way of warming up, let us fix notation. We'll write term : type to help distinguish things.

Notation

By way of warming up, let us fix notation. We'll write term : type to help distinguish things.

For the type of *identifications* between types, $A = B$, we'll use

Notation

By way of warming up, let us fix notation. We'll write $\textsf{term} : \textsf{type}$ to help distinguish things.

For the type of *identifications* between types, $A = B$, we'll use

- $A \simeq B :\equiv \sum(f : A \to B), \mathsf{isEquiv}(f)$

Notation

By way of warming up, let us fix notation. We'll write term : type to help distinguish things.

For the type of *identifications* between types, $A = B$, we'll use

- $A \simeq B :\equiv \sum(f : A \to B), \mathsf{isEquiv}(f)$
- $\mathsf{idToEquiv} : (A = B) \to (A \simeq B)$

Notation

By way of warming up, let us fix notation. We'll write term : type to help distinguish things.

For the type of *identifications* between types, $A = B$, we'll use

- $A \simeq B :\equiv \sum(f : A \to B), \mathsf{isEquiv}(f)$
- $\mathsf{idToEquiv} : (A = B) \to (A \simeq B)$
- $\mathsf{coe} : (A = B) \to (A \to B)$

Notation

By way of warming up, let us fix notation. We'll write term : type to help distinguish things.

For the type of *identifications* between types, $A = B$, we'll use

- $A \simeq B :\equiv \sum(f : A \to B), \mathsf{isEquiv}(f)$
- $\mathsf{idToEquiv} : (A = B) \to (A \simeq B)$
- $\mathsf{coe} : (A = B) \to (A \to B)$

For the type of *identifications* between functions, $f = g$, we'll use

Notation

By way of warming up, let us fix notation. We'll write term : type to help distinguish things.

For the type of *identifications* between types, $A = B$, we'll use

- $A \simeq B :\equiv \sum(f : A \to B), \mathsf{isEquiv}(f)$
- $\mathsf{idToEquiv} : (A = B) \to (A \simeq B)$
- $\mathsf{coe} : (A = B) \to (A \to B)$

For the type of *identifications* between functions, $f = g$, we'll use

- $\mathsf{id}_f : f \sim f$

Notation

By way of warming up, let us fix notation. We'll write term : type to help distinguish things.

For the type of *identifications* between types, $A = B$, we'll use

- $A \simeq B :\equiv \sum(f : A \to B), \mathsf{isEquiv}(f)$
- $\mathsf{idToEquiv} : (A = B) \to (A \simeq B)$
- $\mathsf{coe} : (A = B) \to (A \to B)$

For the type of *identifications* between functions, $f = g$, we'll use

- $\mathsf{id}_f : f \sim f$
- $\mathsf{happly} : (f = g) \to (f \sim g)$

Univalence & function extensionality

An unfair and incorrect characterisation

"Univalence sets the type of identifications between types to be equivalences."

An unfair and incorrect characterisation

"Univalence sets the type of identifications between types to be equivalences."

"Function extensionality sets the type of identifications between functions to be homotopies."

An unfair and incorrect characterisation

"Univalence sets the type of identifications between types to be equivalences."

"Function extensionality sets the type of identifications between functions to be homotopies."

Functions are not obviously types.

An unfair and incorrect characterisation

"Univalence sets the type of identifications between types to be equivalences."

"Function extensionality sets the type of identifications between functions to be homotopies."

Functions are not obviously types. The proof we'll talk about today arose from a desire to think of $f \leftrightarrow \mathsf{graph}(f)$.

An unfair and incorrect characterisation

"Univalence sets the type of identifications between types to be equivalences."

"Function extensionality sets the type of identifications between functions to be homotopies."

Functions are not obviously types. The proof we'll talk about today arose from a desire to think of $f \leftrightarrow \mathsf{graph}(\mathsf{f})$.

That didn't work exactly.

An unfair and incorrect characterisation

"Univalence sets the type of identifications between types to be equivalences."

"Function extensionality sets the type of identifications between functions to be homotopies."

Functions are not obviously types. The proof we'll talk about today arose from a desire to think of $f \leftrightarrow \mathsf{graph}(\mathsf{f})$.

That didn't work exactly. Perhaps there is a proof like this?

Types at issue

Let's remind ourselves of the formal statements of the axioms.

Types at issue

Let's remind ourselves of the formal statements of the axioms.

$$\mathsf{ua} : \prod_{(A,B : \mathsf{U_i})} \mathsf{isEquiv}(\mathsf{idToEquiv}_{A,B})$$

Types at issue

Let's remind ourselves of the formal statements of the axioms.

$$\mathsf{ua} : \prod_{(A, B : \mathsf{U_i})} \mathsf{isEquiv}(\mathsf{idToEquiv}_{A,B})$$

$$\mathsf{funext} : \prod_{(A : \mathsf{U_i})} \prod_{(B : A \to \mathsf{U_j})} \prod_{(f, g : \prod (a : A), B)} \mathsf{isEquiv}(\mathsf{happly}_{A,B,f,g})$$

Types at issue

Let's remind ourselves of the formal statements of the axioms.

$$\mathsf{ua} : \prod_{(A,B : \mathsf{U_i})} \mathsf{isEquiv}(\mathsf{idToEquiv}_{A,B})$$

$$\mathsf{funext} : \prod_{(A : \mathsf{U_i})} \prod_{(B : A \to \mathsf{U_j})} \prod_{(f,g : \prod(a:A),B)} \mathsf{isEquiv}(\mathsf{happly}_{A,B,f,g})$$

We will prove that from a term ua we may deduce funext.

Proof idea

Let us fix $A : \mathsf{U_i}$, $B : A \to \mathsf{U_j}$, and $f : \prod(a : A), B$.

$$\sum_{(\,g\,:\,\prod(a\,:\,A),B\,)} f \sim g$$

Proof idea

Let us fix $A : \mathsf{U_i}$, $B : A \to \mathsf{U_j}$, and $f : \prod(a : A), B$.

$$(f, \mathsf{id}_f), (g, h)$$

$$\sum_{(g : \prod(a:A), B)} f \sim g$$

Proof idea

Let us fix $A : \mathsf{U_i}$, $B : A \to \mathsf{U_j}$, and $f : \prod(a : A), B$.

$$(f, \mathsf{id}_f), (g, h)$$

$$\sum_{(g : \prod(a:A), B)} f \sim g$$

Note: $\mathsf{tr}_{f\sim}(\mathsf{id}_f, p : f = g) = \mathsf{happly}(p)$

Proof idea

Let us fix $A : \mathsf{U_i}$, $B : A \to \mathsf{U_j}$, and $f : \prod(a : A), B$.

$$(f, \mathsf{id}_f), (g, h)$$

$$\sum_{(g : \prod(a : A), B)} f \sim g$$

Note: $\mathsf{tr}_{f\sim}(\mathsf{id}_f, p : f = g) = \mathsf{happly}(p)$

So $(f, \mathsf{id}_f) = (g, h) \simeq (\sum(p : f = g), \mathsf{happly}(p) = h) \equiv \mathsf{fib}_{\mathsf{happly}}(h)$

Proof idea

Let us fix $A : \mathsf{U_i}$, $B : A \to \mathsf{U_j}$, and $f : \prod(a : A), B$.

$(f, \mathsf{id}_f), (g, h)$

$$\sum_{(g : \prod(a:A), B)} f \sim g$$

this is prop $\to$ funext

Note: $\mathsf{tr}_{f\sim}(\mathsf{id}_f, p : f = g) = \mathsf{happly}(p)$
So $(f, \mathsf{id}_f) = (g, h) \simeq (\sum(p : f = g), \mathsf{happly}(p) = h) \equiv \mathsf{fib}_{\mathsf{happly}}(h)$

Proof idea

Let us fix $A : \mathsf{U_i}$, $B : A \to \mathsf{U_j}$, and $f : \prod(a : A), B$.

$(f, \mathsf{id}_f), (g, h)$

$$\sum_{(g : \prod(a:A), B)} f \sim g \qquad\qquad A \to \Big(\sum_{(a:A)} \sum_{(b:Ba)} fa = b\Big)$$

this is prop $\to$ funext

Note: $\mathsf{tr}_{f\sim}(\mathsf{id}_f, p : f = g) = \mathsf{happly}(p)$
So $(f, \mathsf{id}_f) = (g, h) \simeq (\sum(p : f = g), \mathsf{happly}(p) = h) \equiv \mathsf{fib}_{\mathsf{happly}}(h)$

Proof idea

Let us fix $A : \mathsf{U_i}$, $B : A \to \mathsf{U_j}$, and $f : \prod(a : A), B$.

$(f, \mathsf{id}_f), (g, h)$

$$\sum_{(g : \prod(a:A),B)} f \sim g \qquad\qquad A \to \Big(\sum_{(a:A)} \underbrace{\sum_{(b:Ba)} fa = b}_{\mathsf{himg}_f(a)}\Big)$$

this is prop $\to$ funext

Note: $\mathsf{tr}_{f\sim}(\mathsf{id}_f, p : f = g) = \mathsf{happly}(p)$

So $(f, \mathsf{id}_f) = (g, h) \simeq (\sum(p : f = g), \mathsf{happly}(p) = h) \equiv \mathsf{fib}_{\mathsf{happly}}(h)$

Proof idea

Let us fix $A : \mathsf{U_i}$, $B : A \to \mathsf{U_j}$, and $f : \prod(a : A), B$.

$(f, \mathsf{id}_f), (g, h)$

$$\sum_{(g : \prod(a:A), B)} f \sim g \qquad\qquad A \to \Big(\underbrace{\sum_{(a : A)} \underbrace{\sum_{(b : Ba)} fa = b}_{\mathsf{himg}_f(a)}}_{\mathsf{hgraph}(f)}\Big)$$

this is prop $\to$ funext

Note: $\mathsf{tr}_{f\sim}(\mathsf{id}_f, p : f = g) = \mathsf{happly}(p)$

So $(f, \mathsf{id}_f) = (g, h) \simeq (\sum(p : f = g), \mathsf{happly}(p) = h) \equiv \mathsf{fib}_{\mathsf{happly}}(h)$

Proof idea

Let us fix $A : \mathsf{U_i}$, $B : A \to \mathsf{U_j}$, and $f : \prod(a : A), B$.

$(f, \mathsf{id}_f), (g, h)$

$$\sum_{(g : \prod(a:A), B)} f \sim g$$

this is prop $\to$ funext

$\overline{g} \quad a \mapsto (a', b, p)$

$$A \to \underbrace{\Big(\sum_{(a:A)} \underbrace{\sum_{(b : Ba)} fa = b}_{\mathsf{himg}_f(a)} \Big)}_{\mathsf{hgraph}(f)}$$

Note: $\mathsf{tr}_{f\sim}(\mathsf{id}_f, p : f = g) = \mathsf{happly}(p)$

So $(f, \mathsf{id}_f) = (g, h) \simeq (\sum(p : f = g), \mathsf{happly}(p) = h) \equiv \mathsf{fib}_{\mathsf{happly}}(h)$

Proof idea

Let us fix $A : \mathsf{U_i}$, $B : A \to \mathsf{U_j}$, and $f : \prod(a : A), B$.

$$(f, \mathsf{id}_f), (g, h)$$

$$\sum_{(g : \prod(a:A), B)} f \sim g$$

this is prop $\to$ funext

$$\overline{g} \qquad a \mapsto (a', b, p)$$

$$A \to (\underbrace{\sum_{(a:A)} \underbrace{\sum_{(b:Ba)} fa = b}_{\mathsf{himg}_f(a)}}_{\mathsf{hgraph}(f)})$$

with $\mathsf{pr}_A \circ \overline{g} \sim \mathsf{id}_A$

Note: $\mathsf{tr}_{f\sim}(\mathsf{id}_f, p : f = g) = \mathsf{happly}(p)$
So $(f, \mathsf{id}_f) = (g, h) \simeq (\sum(p : f = g), \mathsf{happly}(p) = h) \equiv \mathsf{fib}_{\mathsf{happly}}(h)$

Proof idea

Let us fix $A : \mathsf{U_i}$, $B : A \to \mathsf{U_j}$, and $f : \prod(a : A), B$.

$(f, \mathsf{id}_f), (g, h)$ $\qquad\qquad$ $\overline{g} \quad a \mapsto (a', b, p)$

$$\sum_{(g : \prod(a : A), B)} f \sim g \quad \overset{\checkmark}{\simeq} \quad A \to \Big(\underbrace{\sum_{(a : A)} \underbrace{\sum_{(b : Ba)} fa = b}_{\mathsf{himg}_f(a)}}_{\mathsf{hgraph}(f)} \Big)$$

this is prop $\to$ funext

with $\mathsf{pr}_A \circ \overline{g} \sim \mathsf{id}_A$

Note: $\mathsf{tr}_{f\sim}(\mathsf{id}_f, p : f = g) = \mathsf{happly}(p)$

So $(f, \mathsf{id}_f) = (g, h) \simeq (\sum(p : f = g), \mathsf{happly}(p) = h) \equiv \mathsf{fib}_{\mathsf{happly}}(h)$

But $\sum(\overline{g} : A \to \mathsf{hgraph}(f)), \mathsf{pr_A} \circ \overline{g} \sim \mathsf{id}_A$ is not obviously a prop.

But $\sum(\overline{g} : A \to \mathsf{hgraph}(f)), \mathsf{pr_A} \circ \overline{g} \sim \mathsf{id}_A$ is not obviously a prop.

Let's enlarge slightly,

But $\sum(\overline{g} : A \to \mathsf{hgraph}(f)), \mathsf{pr_A} \circ \overline{g} \sim \mathsf{id}_A$ is not obviously a prop.

Let's enlarge slightly,

$$\sum_{(\overline{g} : A \to \mathsf{hgraph}(f))} \mathsf{pr_A} \circ \overline{g} = \mathsf{id}_A$$

But $\sum(\overline{g} : A \to \mathsf{hgraph}(f)), \mathsf{pr_A} \circ \overline{g} \sim \mathsf{id}_A$ is not obviously a prop.

Let's enlarge slightly,

$$\sum_{(\overline{g} : A \to \mathsf{hgraph}(f))} \mathsf{pr_A} \circ \overline{g} = \mathsf{id}_A$$

How does this compare to what we want?

But $\sum(\overline{g} : A \to \mathsf{hgraph}(f)), \mathsf{pr_A} \circ \overline{g} \sim \mathsf{id}_A$ is not obviously a prop.

Let's enlarge slightly,

$$\sum_{(\,\overline{g} : A \to \mathsf{hgraph}(f)\,)} \mathsf{pr_A} \circ \overline{g} = \mathsf{id}_A \qquad\qquad \sum_{(\,g : \prod(a:A),B\,)} f \sim g$$

How does this compare to what we want?

But $\sum(\overline{g} : A \to \mathsf{hgraph}(f)), \mathsf{pr_A} \circ \overline{g} \sim \mathsf{id}_A$ is not obviously a prop.

Let's enlarge slightly,

$$\sum_{(\overline{g} : A \to \mathsf{hgraph}(f))} \mathsf{pr_A} \circ \overline{g} = \mathsf{id}_A \qquad\qquad \sum_{(g : \prod(a:A), B)} f \sim g$$

How does this compare to what we want?

Recall $\mathsf{hgraph}(f) \equiv (\sum(a : A), \underbrace{\sum(b : Ba), fa = b)}_{\mathsf{himg}_f(a)}$

But $\sum(\overline{g} : A \to \mathsf{hgraph}(f)), \mathsf{pr_A} \circ \overline{g} \sim \mathsf{id}_A$ is not obviously a prop.

Let's enlarge slightly,

$$\begin{tikzcd}[column sep=large] \displaystyle\sum_{(\overline{g} : A \to \mathsf{hgraph}(f))} \mathsf{pr_A} \circ \overline{g} = \mathsf{id}_A & & \displaystyle\sum_{(g : \prod(a:A), B)} f \sim g \arrow[ll, bend left, "{s(g,h) :\equiv (\lambda a.(a,(ga,ha)), \mathsf{refl})}"] \end{tikzcd}$$

How does this compare to what we want?

Recall $\mathsf{hgraph}(f) \equiv (\sum(a:A), \underbrace{\sum(b:Ba), fa = b)}_{\mathsf{himg}_f(a)}$

But $\sum(\overline{g} : A \to \mathsf{hgraph}(f)), \mathsf{pr_A} \circ \overline{g} \sim \mathsf{id}_A$ is not obviously a prop.

Let's enlarge slightly,

$$\begin{tikzcd} \sum\limits_{(\overline{g} : A \to \mathsf{hgraph}(f))} \mathsf{pr_A} \circ \overline{g} = \mathsf{id}_A \arrow[rr, bend left, "r(\overline{g}, p) :\equiv \text{transport yoga in } \mathsf{himg}_f"] & & \sum\limits_{(g : \prod(a : A), B)} f \sim g \arrow[ll, bend left, "s(g, h) :\equiv (\lambda a.(a, (ga, ha)), \mathsf{refl})"] \end{tikzcd}$$

How does this compare to what we want?

Recall $\mathsf{hgraph}(f) \equiv (\sum(a : A), \underbrace{\sum(b : Ba), fa = b}_{\mathsf{himg}_f(a)})$

But $\sum(\overline{g} : A \to \mathsf{hgraph}(f)), \mathsf{pr_A} \circ \overline{g} \sim \mathsf{id}_A$ is not obviously a prop.

Let's enlarge slightly,

$$\begin{tikzcd} \sum\limits_{(\overline{g} : A \to \mathsf{hgraph}(f))} \mathsf{pr_A} \circ \overline{g} = \mathsf{id}_A \arrow[rr, bend left, "r(\overline{g}, p) :\equiv \text{transport yoga in } \mathsf{himg}_f"] \arrow[rr, phantom, "\boxed{\text{retract}}"] & & \sum\limits_{(g : \prod(a:A), B)} f \sim g \arrow[ll, bend left, "s(g, h) :\equiv (\lambda a.(a, (ga, ha)), \mathsf{refl})"] \end{tikzcd}$$

How does this compare to what we want?

Recall $\mathsf{hgraph}(f) \equiv (\sum(a : A), \underbrace{\sum(b : Ba), fa = b)}_{\mathsf{himg}_f(a)}$

What we have,

$$\left(\sum_{(\,\overline{g}\,:\,A\to\mathsf{hgraph}(f)\,)} \mathsf{pr}_A \circ \overline{g} = \mathsf{id}_A\right) \underset{s}{\overset{r}{\rightleftarrows}} \left(\sum_{(\,g\,:\,\prod(a\,:\,A),B\,)} f \sim g\right)$$

What we have,

$$\left(\sum_{(\,\overline{g}\,:\,A\to\mathsf{hgraph}(f)\,)} \mathsf{pr}_A \circ \overline{g} = \mathsf{id}_A\right) \underset{s}{\overset{r}{\rightleftarrows}} \left(\sum_{(\,g\,:\,\prod(a\,:\,A),B\,)} f \sim g\right)$$

is good enough™.

What we have,

$$\left(\sum_{(\,\overline{g}\,:\,A \to \mathsf{hgraph}(f)\,)} \mathsf{pr}_A \circ \overline{g} = \mathsf{id}_A\right) \underset{s}{\overset{r}{\rightleftarrows}} \left(\sum_{(\,g\,:\,\prod(a\,:\,A),B\,)} f \sim g\right)$$

is good enough™. Being a prop is contagious across retracts.

What we have,

$$\left(\sum_{(\,\overline{g}\,:\,A \to \mathsf{hgraph}(f)\,)} \mathsf{pr}_A \circ \overline{g} = \mathsf{id}_A \right) \underset{s}{\overset{r}{\rightleftarrows}} \left(\sum_{(\,g\,:\,\prod(a\,:\,A),B\,)} f \sim g \right)$$

is good enough™. Being a prop is contagious across retracts.

$$⚠ \qquad \mathsf{fib}_{\mathsf{pr}_A \circ (-)}(\mathsf{id}_A) \equiv \sum_{(\,\overline{g}\,:\,A \to \mathsf{hgraph}(f)\,)} \mathsf{pr}_\mathsf{A} \circ \overline{g} = \mathsf{id}_A \qquad ⚠$$

What we have,

$$\left(\sum_{(\overline{g}\,:\,A\to\mathsf{hgraph}(f))} \mathsf{pr}_A \circ \overline{g} = \mathsf{id}_A\right) \underset{s}{\overset{r}{\rightleftarrows}} \left(\sum_{(g\,:\,\prod(a\,:\,A),B)} f \sim g\right)$$

is good enough™. Being a prop is contagious across retracts.

$$⚠ \qquad \mathsf{fib}_{\mathsf{pr}_A\circ(-)}(\mathsf{id}_A) \equiv \sum_{(\overline{g}\,:\,A\to\mathsf{hgraph}(f))} \mathsf{pr}_\mathsf{A} \circ \overline{g} = \mathsf{id}_A \qquad ⚠$$

Theorem (Voevodsky)

Assuming only ua, if $q : C \to D$ is an equivalence then so too is $q \circ (-) : (E \to C) \to (E \to D)$.

What we have,

$$\left(\sum_{(\overline{g}\,:\,A \to \mathsf{hgraph}(f))} \mathsf{pr}_A \circ \overline{g} = \mathsf{id}_A\right) \underset{s}{\overset{r}{\rightleftarrows}} \left(\sum_{(g\,:\,\prod(a\,:\,A),B)} f \sim g\right)$$

is good enough™. Being a prop is contagious across retracts.

$$⚠ \qquad \mathsf{fib}_{\mathsf{pr}_A \circ (-)}(\mathsf{id}_A) \equiv \sum_{(\overline{g}\,:\,A \to \mathsf{hgraph}(f))} \mathsf{pr}_\mathsf{A} \circ \overline{g} = \mathsf{id}_A \qquad ⚠$$

Theorem (Voevodsky)

Assuming only ua, if $q : C \to D$ is an equivalence then so too is $q \circ (-) : (E \to C) \to (E \to D)$.

And for our silly type $\mathsf{pr}_A : \mathsf{hgraph}(f) \to A$ *is an equivalence!*

Putting it together

ua

Putting it together

ua

- Define hgraph$(f) :\equiv \sum(a : A), \sum(b : Ba), fa = b$

Putting it together

ua

- Define $\mathsf{hgraph}(f) :\equiv \sum(a : A), \sum(b : Ba), fa = b$
- Deduce $\mathsf{isEquiv}(\mathsf{pr}_A)$

Putting it together

ua

- Define $\mathsf{hgraph}(f) :\equiv \sum(a : A), \sum(b : Ba), fa = b$
- Deduce $\mathsf{isEquiv}(\mathsf{pr}_A)$
- Apply thm. of V.V. $\rightsquigarrow$ $\mathsf{ua} \mapsto p : \mathsf{isEquiv}(\mathsf{pr}_A \circ (-))$

Putting it together

$\mathsf{ua} \to \mathsf{isContr}(\mathsf{fib}_{\mathsf{pr}_A \circ (-)}(\mathsf{id}_A))$

- Define $\mathsf{hgraph}(f) :\equiv \sum(a : A), \sum(b : Ba), fa = b$
- Deduce $\mathsf{isEquiv}(\mathsf{pr}_A)$
- Apply thm. of V.V. $\rightsquigarrow$ $\mathsf{ua} \mapsto p : \mathsf{isEquiv}(\mathsf{pr}_A \circ (-))$

Putting it together

$\mathsf{ua} \to \mathsf{isContr}(\mathsf{fib}_{\mathsf{pr}_A \circ (-)}(\mathsf{id}_A))$

- Define $\mathsf{hgraph}(f) :\equiv \sum(a : A), \sum(b : Ba), fa = b$
- Deduce $\mathsf{isEquiv}(\mathsf{pr}_A)$
- Apply thm. of V.V. $\rightsquigarrow$ $\mathsf{ua} \mapsto p : \mathsf{isEquiv}(\mathsf{pr}_A \circ (-))$
- Retract $\mathsf{fib}_{\mathsf{pr}_A \circ (-)}(\mathsf{id}_A)$ onto $\sum(g : \prod(a : A), B), f \sim g$

Putting it together

$$\mathsf{ua} \to \mathsf{isContr}(\mathsf{fib}_{\mathsf{pr}_A \circ (-)}(\mathsf{id}_A)) \to \mathsf{isContr}\left(\sum_{(g : \prod(a:A), B)} f \sim g\right)$$

- Define $\mathsf{hgraph}(f) :\equiv \sum(a : A), \sum(b : Ba), fa = b$
- Deduce $\mathsf{isEquiv}(\mathsf{pr}_A)$
- Apply thm. of V.V. $\rightsquigarrow$ $\mathsf{ua} \mapsto p : \mathsf{isEquiv}(\mathsf{pr}_A \circ (-))$
- Retract $\mathsf{fib}_{\mathsf{pr}_A \circ (-)}(\mathsf{id}_A)$ onto $\sum(g : \prod(a : A), B), f \sim g$

Putting it together

$$\mathsf{ua} \to \mathsf{isContr}(\mathsf{fib}_{\mathsf{pr}_A \circ (-)}(\mathsf{id}_A)) \to \mathsf{isContr}\left(\sum_{(g : \prod(a:A), B)} f \sim g\right)$$

- Define $\mathsf{hgraph}(f) :\equiv \sum(a : A), \sum(b : Ba), fa = b$
- Deduce $\mathsf{isEquiv}(\mathsf{pr}_A)$
- Apply thm. of V.V. $\rightsquigarrow$ $\mathsf{ua} \mapsto p : \mathsf{isEquiv}(\mathsf{pr}_A \circ (-))$
- Retract $\mathsf{fib}_{\mathsf{pr}_A \circ (-)}(\mathsf{id}_A)$ onto $\sum(g : \prod(a : A), B), f \sim g$
- Do some equality induction

Putting it together

$$\mathsf{ua} \to \mathsf{isContr}(\mathsf{fib}_{\mathsf{pr}_A \circ (-)}(\mathsf{id}_A)) \to \mathsf{isContr}\left(\sum_{(g : \prod(a:A), B)} f \sim g\right)$$
$$\to \mathsf{funext}$$

- Define $\mathsf{hgraph}(f) :\equiv \sum(a : A), \sum(b : Ba), fa = b$
- Deduce $\mathsf{isEquiv}(\mathsf{pr}_A)$
- Apply thm. of V.V. $\rightsquigarrow$ $\mathsf{ua} \mapsto p : \mathsf{isEquiv}(\mathsf{pr}_A \circ (-))$
- Retract $\mathsf{fib}_{\mathsf{pr}_A \circ (-)}(\mathsf{id}_A)$ onto $\sum(g : \prod(a : A), B), f \sim g$
- Do some equality induction

Putting it together

$$\mathsf{ua} \to \mathsf{isContr}(\mathsf{fib}_{\mathsf{pr}_A \circ (-)}(\mathsf{id}_A)) \to \mathsf{isContr}\left(\sum_{(g : \prod(a:A), B)} f \sim g\right)$$
$$\to \mathsf{funext}$$

- Define $\mathsf{hgraph}(f) :\equiv \sum(a : A), \sum(b : Ba), fa = b$
- Deduce $\mathsf{isEquiv}(\mathsf{pr}_A)$
- Apply thm. of V.V. $\rightsquigarrow$ $\mathsf{ua} \mapsto p : \mathsf{isEquiv}(\mathsf{pr}_A \circ (-))$
- Retract $\mathsf{fib}_{\mathsf{pr}_A \circ (-)}(\mathsf{id}_A)$ onto $\sum(g : \prod(a : A), B), f \sim g$
- Do some equality induction

$\blacksquare$

This proof is not intricate,

This proof is not intricate, most complicated part is some 2-path algebra.

Is the aim of type theory is to make that which is formal, formally formal?

This proof is not intricate, most complicated part is some 2-path algebra.

I have formalised it…

This proof is not intricate, most complicated part is some 2-path algebra.

I have formalised it… ahem in that other language.

Is the aim of type theory is to make that which is formal, formally formal?

This proof is not intricate, most complicated part is some 2-path algebra.

I have formalised it… ahem in that other language.

$\sim$ 200 lines on the HoTT library,

Is the aim of type theory is to make that which is formal, formally formal?

This proof is not intricate, most complicated part is some 2-path algebra.

I have formalised it… ahem in that other language.

$\sim$ 200 lines on the HoTT library,

$\sim$ 50 of which is a single, uninteresting, technical fact about equivalences

Is the aim of type theory is to make that which is formal, formally formal?

This proof is not intricate, most complicated part is some 2-path algebra.

I have formalised it... ahem in that other language.

$\sim$ 200 lines on the HoTT library,

$\sim$ 50 of which is a single, uninteresting, technical fact about equivalences $\rightsquigarrow$ good exercise?

Univalence

Facts

- Implies function extensionality

Univalence

Facts

- Implies function extensionality
- Along with some mild assumptions, U is not a set

Univalence

Facts

- Implies function extensionality
- Along with some mild assumptions, $\mathcal{U}$ is not a set
- Consistent to assume (Voevodsky's model in sSET)

Univalence

Facts

- Implies function extensionality
- Along with some mild assumptions, U is not a set
- Consistent to assume (Voevodsky's model in sSet)
- Consequently, for any proposition $P : \mathsf{U} \to \mathsf{U}$ and witness $A \simeq B$, one cannot show $P(A)$ but not $P(B)$.

Univalence

Facts

- Implies function extensionality
- Along with some mild assumptions, U is not a set
- Consistent to assume (Voevodsky's model in sSET)
- Consequently, for any proposition $P : \mathsf{U} \to \mathsf{U}$ and witness $A \simeq B$, one cannot show $P(A)$ but not $P(B)$.

Univalence

Facts

- Implies function extensionality
- Along with some mild assumptions, U is not a set
- Consistent to assume (Voevodsky's model in sSET)
- Consequently, for any proposition $P : \mathsf{U} \to \mathsf{U}$ and witness $A \simeq B$, one cannot show $P(A)$ but not $P(B)$.

- By the *structure identity principle*, ua means that many algebraic things are univalent too.

Univalence, the structure-identity principle, and you

Set Theory™ does not understand equality of *structures*:

Set Theory™ does not understand equality of *structures*:

- given groups $G \cong H$

Set Theory™ does not understand equality of *structures*:

- given groups $G \cong H$
- “everything that’s true of G is true of H”

Set Theory™ does not understand equality of *structures*:

- given groups $G \cong H$
- "everything that's true of G is true of H"
- vs $\emptyset \in G$

Set Theory™ does not understand equality of *structures*:

- given groups $G \cong H$
- "everything that's true of G is true of H"
- vs $\emptyset \in G$

Isolate "group theoretic statements"

Set Theory™ does not understand equality of *structures*:

- given groups $G \cong H$
- "everything that's true of G is true of H"
- vs $\emptyset \in G$

Isolate "group theoretic statements"

…and everything related to groups

Set Theory™ does not understand equality of *structures*:

- given groups $G \cong H$
- "everything that's true of G is true of H"
- vs $\emptyset \in G$

Isolate "group theoretic statements"

…and everything related to groups

OR

Change foundations

For a flavour of the *structure-identity* principle, let's pick our favourite toy algebraic structure,

For a flavour of the *structure-identity* principle, let's pick our favourite toy algebraic structure, ~~affine schemes~~ magmas

For a flavour of the *structure-identity* principle, let's pick our favourite toy algebraic structure, ~~affine schemes~~ magmas

$$\mathsf{Magma} :\equiv \sum (A : \mathsf{Set}), A \to (A \to A)$$

For a flavour of the *structure-identity* principle, let's pick our favourite toy algebraic structure, ~~affine schemes~~ magmas

$$\mathsf{Magma} :\equiv \sum(A : \mathsf{Set}), A \to (A \to A)$$

Audience Participation

For a flavour of the *structure-identity* principle, let's pick our favourite toy algebraic structure, ~~affine schemes~~ magmas

$$\mathsf{Magma} :\equiv \sum(A : \mathsf{Set}), A \to (A \to A)$$

Audience Participation

$$\left(f : A \simeq B,\ t : \prod(a, b : A), f(a \star b) =_B (fa) \oplus (fb)\right)$$

For a flavour of the *structure-identity* principle, let's pick our favourite toy algebraic structure, ~~affine schemes~~ magmas

$$\mathsf{Magma} :\equiv \sum(A : \mathsf{Set}), A \to (A \to A)$$

Audience Participation

$$(f : A \simeq B,\ t : \prod(a, b : A), f(a \star b) =_B (fa) \oplus (fb))$$

$$\xleftrightarrow[\text{funext}]{\text{happly}}$$

For a flavour of the *structure-identity* principle, let's pick our favourite toy algebraic structure, ~~affine schemes~~ magmas

$$\mathsf{Magma} :\equiv \textstyle\sum(A : \mathsf{Set}), A \to (A \to A)$$

Audience Participation

$$(f : A \simeq B,\ t : \textstyle\prod(a, b : A), f(a \star b) =_B (fa) \oplus (fb))$$

$$\xleftrightarrow[\text{funext}]{\text{happly}} \quad (f : A \simeq B,\ s : (f \star =_{B \to (B \to B)} \oplus (f \times f)))$$

For a flavour of the *structure-identity* principle, let's pick our favourite toy algebraic structure, ~~affine schemes~~ magmas

$$\mathsf{Magma} :\equiv \sum(A : \mathsf{Set}), A \to (A \to A)$$

Audience Participation

$$(f : A \simeq B,\ t : \prod(a, b : A), f(a \star b) =_B (fa) \oplus (fb))$$

$$\underset{\mathsf{funext}}{\overset{\mathsf{happly}}{\longleftrightarrow}} \qquad (f : A \simeq B,\ s : (f \star =_{B \to (B \to B)} \oplus (f \times f)))$$

$$\underset{\mathsf{ua}}{\overset{\mathsf{idtoeqv}}{\longleftrightarrow}}$$

For a flavour of the *structure-identity* principle, let's pick our favourite toy algebraic structure, ~~affine schemes~~ magmas

$$\mathsf{Magma} :\equiv \sum(A : \mathsf{Set}), A \to (A \to A)$$

Audience Participation

$$(f : A \simeq B,\ t : \prod(a, b : A), f(a \star b) =_B (fa) \oplus (fb))$$

$$\underset{\mathsf{funext}}{\overset{\mathsf{happly}}{\longleftrightarrow}} \quad (f : A \simeq B,\ s : (f \star =_{B \to (B \to B)} \oplus (f \times f)))$$

$$\underset{\mathsf{ua}}{\overset{\mathsf{idtoeqv}}{\longleftrightarrow}} \quad (q : A =_{\mathsf{U}} B,\ r : \mathsf{transport}(q, \star) =_{B \to (B \to B)} \oplus)$$

For a flavour of the *structure-identity* principle, let's pick our favourite toy algebraic structure, ~~affine schemes~~ magmas

$$\mathsf{Magma} :\equiv \sum(A : \mathsf{Set}), A \to (A \to A)$$

Audience Participation

$$(f : A \simeq B,\ t : \prod(a, b : A), f(a \star b) =_B (fa) \oplus (fb))$$

$$\xleftrightarrow[\text{funext}]{\text{happly}} \quad (f : A \simeq B,\ s : (f \star =_{B \to (B \to B)} \oplus (f \times f)))$$

$$\xleftrightarrow[\text{ua}]{\text{idtoeqv}} \quad (q : A =_{\mathsf{U}} B,\ r : \mathsf{transport}(q, \star) =_{B \to (B \to B)} \oplus)$$

$$\xleftrightarrow[\text{lemmas}]{\text{generic}}$$

For a flavour of the *structure-identity* principle, let's pick our favourite toy algebraic structure, ~~affine schemes~~ magmas

$$\mathsf{Magma} :\equiv \sum(A : \mathsf{Set}), A \to (A \to A)$$

Audience Participation

$$(f : A \simeq B,\ t : \prod(a, b : A), f(a \star b) =_B (fa) \oplus (fb))$$

$$\xleftrightarrow[\text{funext}]{\text{happly}} \quad (f : A \simeq B,\ s : (f \star =_{B \to (B \to B)} \oplus (f \times f)))$$

$$\xleftrightarrow[\text{ua}]{\text{idtoeqv}} \quad (q : A =_{\mathsf{U}} B,\ r : \mathsf{transport}(q, \star) =_{B \to (B \to B)} \oplus)$$

$$\xleftrightarrow[\text{lemmas}]{\text{generic}} \quad p : (A, \star) =_{\mathsf{Magma}} (B, \oplus)$$

Structure-identity principle

For which structures does univalent type theory automatically understand equality?

Structure-identity principle

For which structures does univalent type theory automatically understand equality? A possible answer using category theory:

Structure-identity principle

For which structures does univalent type theory automatically understand equality? A possible answer using category theory:

Definition

A category C is univalent when $\mathsf{idtoiso} : a = b \to \mathsf{iso}_C(a, b)$ is an equivalence.

Structure-identity principle

For which structures does univalent type theory automatically understand equality? A possible answer using category theory:

Definition

A category C is univalent when $\mathsf{idtoiso} : a = b \to \mathsf{iso}_C(a, b)$ is an equivalence.

Theorem

If C is a univalent category and T is a monad on C then $\mathsf{Alg}(T)$ is a univalent category.

Structure-identity principle

For which structures does univalent type theory automatically understand equality? A possible answer using category theory:

Definition

A category C is univalent when $\mathsf{idtoiso} : a = b \to \mathsf{iso}_C(a, b)$ is an equivalence.

Theorem

If C is a univalent category and T is a monad on C then $\mathsf{Alg}(T)$ is a univalent category.

“isomorphism of algebraic structures is equivalent to equality”

For which structures does univalent type theory automatically understand equality? A possible answer using category theory:

Definition

A category C is univalent when $\mathsf{idtoiso} : a = b \to \mathsf{iso}_C(a,b)$ is an equivalence.

Theorem

If C is a univalent category and T is a monad on C then $\mathsf{Alg}(T)$ is a univalent category.

“isomorphism of algebraic structures is equivalent to equality”

...where does the S.I.P. fail?

Open questions the formulation of univalence

Putting the *fun* in function extensionality

Definition (Naïve function extensionality)

$$\mathsf{nfunext} : \prod_{(A : \mathsf{U_i})} \prod_{(B : \mathsf{U_j})} \prod_{(f, g : \prod(a : A), B)} f \sim g \to f = g$$

Putting the *fun* in function extensionality

Definition (Naïve function extensionality)

$$\mathsf{nfunext} : \prod_{(A : \mathsf{U_i})} \prod_{(B : \mathsf{U_j})} \prod_{(f, g : \prod (a : A), B)} f \sim g \to f = g$$

- A priori unrelated to happly

Putting the *fun* in function extensionality

Definition (Naïve function extensionality)

$$\mathsf{nfunext} : \prod_{(A : \mathsf{U_i})} \prod_{(B : \mathsf{U_j})} \prod_{(f, g : \prod (a : A), B)} f \sim g \to f = g$$

- A priori unrelated to happly
- We can easily upgrade to nfunext′ which satisfies $\mathsf{nfunext}'(\mathsf{happly}(p)) = p$ by 'subtracting-off' $\mathsf{nfunext}(\mathsf{id}_f)$

Putting the *fun* in function extensionality

Definition (Naïve function extensionality)

$$\mathsf{nfunext} : \prod_{(A:\mathsf{U_i})(B:\mathsf{U_j})(f,g:\prod(a:A),B)} f \sim g \to f = g$$

- A priori unrelated to happly
- We can easily upgrade to nfunext′ which satisfies $\mathsf{nfunext}'(\mathsf{happly}(p)) = p$ by 'subtracting-off' $\mathsf{nfunext}(\mathsf{id}_f)$
- This fixed nfunext′ is compatible with the first, can inhabit $\mathsf{happly} \circ \mathsf{nfunext}' = \mathsf{happly} \circ \mathsf{nfunext}$

Definition (Naïve function extensionality)

$$\mathsf{nfunext} : \prod_{(A : \mathsf{U_i})} \prod_{(B : \mathsf{U_j})} \prod_{(f, g : \prod(a : A), B)} f \sim g \to f = g$$

- A priori unrelated to happly
- We can easily upgrade to nfunext′ which satisfies $\mathsf{nfunext}'(\mathsf{happly}(p)) = p$ by 'subtracting-off' $\mathsf{nfunext}(\mathsf{id}_f)$
- This fixed nfunext′ is compatible with the first, can inhabit $\mathsf{happly} \circ \mathsf{nfunext}' = \mathsf{happly} \circ \mathsf{nfunext}$

Can we do better?

The setup

Theorem

From nfunext we may derive funext.

The setup

Theorem

From nfunext we may derive funext.

What about ua?

Theorem

From nfunext we may derive funext.

What about ua?

Definition (Naïve univalence)

$$\mathsf{nua} : \prod(A, B : \mathsf{U_i}), A \simeq B \to A = B$$

The setup

Theorem

From nfunext we may derive funext.

What about ua?

Definition (Naïve univalence)

$$\mathsf{nua} : \prod(A, B : \mathsf{U_i}), A \simeq B \to A = B$$

$$\mathsf{nua}\beta : \prod(A, B : \mathsf{U_i}), \prod((f, e) : A \simeq B), \mathsf{coe}(\mathsf{nua}(f, e)) = f$$

The setup

Theorem

From nfunext we may derive funext.

What about ua?

Definition (Naïve univalence)

$$\mathsf{nua} : \prod(A, B : \mathsf{U_i}), A \simeq B \to A = B$$

$$\mathsf{nua}\beta : \prod(A, B : \mathsf{U_i}), \prod((f, e) : A \simeq B), \mathsf{coe}(\mathsf{nua}(f, e)) = f$$

It is known that nua+nuaβ $\leftrightarrow$ ua

The questions

Definition (Naïve univalence)

$$\mathsf{nua} : \prod(A, B : \mathsf{U_i}), A \simeq B \to A = B$$

$$\mathsf{nua}\beta : \prod(A, B : \mathsf{U_i}), \prod((f, e) : A \simeq B), \mathsf{coe}(\mathsf{nua}(f, e)) = f$$

The questions

Definition (Naïve univalence)

$$\mathsf{nua} : \prod(A, B : \mathsf{U_i}), A \simeq B \to A = B$$

$$\mathsf{nua}\beta : \prod(A, B : \mathsf{U_i}), \prod((f, e) : A \simeq B), \mathsf{coe}(\mathsf{nua}(f, e)) = f$$

Question

From nua alone can we derive ua?

The questions

Definition (Naïve univalence)

$$\mathsf{nua} : \prod(A, B : \mathsf{U_i}), A \simeq B \to A = B$$

$$\mathsf{nua}\beta : \prod(A, B : \mathsf{U_i}), \prod((f, e) : A \simeq B), \mathsf{coe}(\mathsf{nua}(f, e)) = f$$

Question

From nua alone can we derive ua?

Our proof of ua $\mapsto$ funext made essential use of uaβ

The questions

Definition (Naïve univalence)

$$\mathsf{nua} : \prod(A, B : \mathsf{U_i}), A \simeq B \to A = B$$

$$\mathsf{nua}\beta : \prod(A, B : \mathsf{U_i}), \prod((f, e) : A \simeq B), \mathsf{coe}(\mathsf{nua}(f, e)) = f$$

Question

From nua alone can we derive ua?

Our proof of ua $\mapsto$ funext made essential use of uaβ

Question

From nua alone can we derive funext?

Equivalent reformulations

The Orton-Pitts reformulation

Based on our previous discussion we'll assume ambient funext

The Orton-Pitts reformulation

Based on our previous discussion we'll assume ambient funext

Definition (Orton-Pitts naïve univalence)

1. unit : $A = \sum(a : A), 1$

These imply nua. Given $(f, e) : A \simeq B$,

The Orton-Pitts reformulation

Based on our previous discussion we'll assume ambient funext

Definition (Orton-Pitts naïve univalence)

1. unit : $A = \sum(a : A), 1$
2. flip : $\sum(a : A), \sum(b : B), C = \sum(b : B), \sum(a : A), C$

These imply nua. Given $(f, e) : A \simeq B$,

The Orton-Pitts reformulation

Based on our previous discussion we'll assume ambient funext

Definition (Orton-Pitts naïve univalence)

1. unit : $A = \sum(a : A), 1$
2. flip : $\sum(a : A), \sum(b : B), C = \sum(b : B), \sum(a : A), C$
3. contract : $\mathsf{isContr}(A) \to (A = 1)$

These imply nua. Given $(f, e) : A \simeq B$,

Based on our previous discussion we'll assume ambient funext

Definition (Orton-Pitts naïve univalence)

1. unit : $A = \sum(a : A), 1$
2. flip : $\sum(a : A), \sum(b : B), C = \sum(b : B), \sum(a : A), C$
3. contract : $\mathsf{isContr}(A) \to (A = 1)$

These imply nua. Given $(f, e) : A \simeq B$,

Based on our previous discussion we'll assume ambient funext

Definition (Orton-Pitts naïve univalence)

1. unit : $A = \sum(a : A), 1$
2. flip : $\sum(a : A), \sum(b : B), C = \sum(b : B), \sum(a : A), C$
3. contract : $\mathsf{isContr}(A) \to (A = 1)$

These imply nua. Given $(f, e) : A \simeq B$,

A

Based on our previous discussion we'll assume ambient funext

Definition (Orton-Pitts naïve univalence)

1. unit : $A = \sum(a : A), 1$
2. flip : $\sum(a : A), \sum(b : B), C = \sum(b : B), \sum(a : A), C$
3. contract : $\mathsf{isContr}(A) \to (A = 1)$

These imply nua. Given $(f, e) : A \simeq B$,

$$A \overset{(1.)}{=} (\sum(a : A), 1)$$

Based on our previous discussion we'll assume ambient funext

Definition (Orton-Pitts naïve univalence)

1. unit : $A = \sum(a : A), 1$
2. flip : $\sum(a : A), \sum(b : B), C = \sum(b : B), \sum(a : A), C$
3. contract : isContr$(A) \to (A = 1)$

These imply nua. Given $(f, e) : A \simeq B$,

$$A \overset{(1.)}{=} \left(\sum(a : A), 1\right) \overset{(3.)^*}{=} \left(\sum(a : A), \overbrace{\sum(b : B), fa = b}^{A \to \mathsf{U}}\right)$$

Based on our previous discussion we'll assume ambient funext

Definition (Orton-Pitts naïve univalence)

1. unit : $A = \sum(a : A), 1$
2. flip : $\sum(a : A), \sum(b : B), C = \sum(b : B), \sum(a : A), C$
3. contract : $\mathsf{isContr}(A) \to (A = 1)$

These imply nua. Given $(f, e) : A \simeq B$,

$$A \overset{(1.)}{=} (\textstyle\sum(a : A), 1) \overset{(3.)^*}{=} (\textstyle\sum(a : A), \overbrace{\textstyle\sum(b : B), f\,a = b}^{A \to \mathsf{U}})$$

$$\overset{(2.)}{=} (\textstyle\sum(b : B), \underbrace{\textstyle\sum(a : A), f\,a = b}_{B \to \mathsf{U}})$$

Based on our previous discussion we'll assume ambient funext

Definition (Orton-Pitts naïve univalence)

1. unit : $A = \sum(a : A), 1$
2. flip : $\sum(a : A), \sum(b : B), C = \sum(b : B), \sum(a : A), C$
3. contract : isContr$(A) \to (A = 1)$

These imply nua. Given $(f, e) : A \simeq B$,

$$A \overset{(1.)}{=} \left(\sum(a : A), 1\right) \overset{(3.)^*}{=} \left(\sum(a : A), \overbrace{\sum(b : B), f a = b}^{A \to \mathsf{U}}\right)$$

$$\overset{(2.)}{=} \left(\sum(b : B), \underbrace{\sum(a : A), f a = b}_{B \to \mathsf{U}}\right) \overset{(3.)^*}{=} \left(\sum(b : B), 1\right)$$

Based on our previous discussion we'll assume ambient funext

Definition (Orton-Pitts naïve univalence)

1. unit : $A = \sum(a : A), 1$
2. flip : $\sum(a : A), \sum(b : B), C = \sum(b : B), \sum(a : A), C$
3. contract : $\mathsf{isContr}(A) \to (A = 1)$

These imply nua. Given $(f, e) : A \simeq B$,

$$A \overset{(1.)}{=} (\textstyle\sum(a : A), 1) \overset{(3.)^*}{=} (\textstyle\sum(a : A), \overbrace{\textstyle\sum(b : B), f\,a = b}^{A \to \mathsf{U}})$$

$$\overset{(2.)}{=} (\textstyle\sum(b : B), \underbrace{\textstyle\sum(a : A), f\,a = b}_{B \to \mathsf{U}}) \overset{(3.)^*}{=} (\textstyle\sum(b : B), 1) \overset{(1.)}{=} B$$

The Orton-Pitts reformulation

Classically (1.), (2.), (3.) are implied by nua, so equivalent.

The Orton-Pitts reformulation

Classically (1.), (2.), (3.) are implied by nua, so equivalent.

Definition (Orton-Pitts nuaβ)

4. unitfi : $(\mathsf{coe}(\mathsf{unit}))a = (a, *)$

The Orton-Pitts reformulation

Classically (1.), (2.), (3.) are implied by nua, so equivalent.

Definition (Orton-Pitts nuaβ)

4. unitfi : $(\mathsf{coe}(\mathsf{unit}))a = (a, *)$
5. flipfi : $(\mathsf{coe}(\mathsf{flip}))(a, b, c) = (b, a, c)$

The Orton-Pitts reformulation

Classically (1.), (2.), (3.) are implied by nua, so equivalent.

Definition (Orton-Pitts nuaβ)

4. unitfi : $(\mathsf{coe}(\mathsf{unit}))a = (a, *)$
5. flipfi : $(\mathsf{coe}(\mathsf{flip}))(a, b, c) = (b, a, c)$

(1.)-(5.) together imply nuaβ. Given $(f, e) : A \simeq B$,

The Orton-Pitts reformulation

Classically (1.), (2.), (3.) are implied by nua, so equivalent.

Definition (Orton-Pitts nuaβ)

4. unitfi : $(\mathsf{coe}(\mathsf{unit}))a = (a, *)$
5. flipfi : $(\mathsf{coe}(\mathsf{flip}))(a, b, c) = (b, a, c)$

(1.)-(5.) together imply nuaβ. Given $(f, e) : A \simeq B$,

$a : A$

The Orton-Pitts reformulation

Classically (1.), (2.), (3.) are implied by nua, so equivalent.

Definition (Orton-Pitts nuaβ)

4. unitfi : $(\mathsf{coe}(\mathsf{unit}))a = (a, *)$
5. flipfi : $(\mathsf{coe}(\mathsf{flip}))(a, b, c) = (b, a, c)$

(1.)-(5.) together imply nuaβ. Given $(f, e) : A \simeq B$,

$$a : A \overset{(1.),(4.)}{\mapsto} (a, *) : \sum_{(a : A)} 1$$

The Orton-Pitts reformulation

Classically (1.), (2.), (3.) are implied by nua, so equivalent.

Definition (Orton-Pitts nuaβ)

4. unitfi : $(\mathsf{coe}(\mathsf{unit}))a = (a, *)$
5. flipfi : $(\mathsf{coe}(\mathsf{flip}))(a, b, c) = (b, a, c)$

(1.)-(5.) together imply nuaβ. Given $(f, e) : A \simeq B$,

$$a : A \overset{(1.),(4.)}{\mapsto} (a, *) : \sum_{(a:A)} 1 \overset{(3.)^*}{\mapsto} (a, fa, \mathsf{refl}) : \sum_{(a:A)} \sum_{(b:B)} fa = b$$

The Orton-Pitts reformulation

Classically (1.), (2.), (3.) are implied by nua, so equivalent.

Definition (Orton-Pitts nuaβ)

4. unitfi : $(\mathsf{coe}(\mathsf{unit}))a = (a, *)$
5. flipfi : $(\mathsf{coe}(\mathsf{flip}))(a, b, c) = (b, a, c)$

(1.)-(5.) together imply nuaβ. Given $(f, e) : A \simeq B$,

$$a : A \overset{(1.),(4.)}{\mapsto} (a, *) : \sum_{(a:A)} 1 \overset{(3.)^*}{\mapsto} (a, fa, \mathsf{refl}) : \sum_{(a:A)} \sum_{(b:B)} fa = b$$

$$\overset{(2.),(5.)}{\mapsto} (fa, a, \mathsf{refl}) : \sum_{(b:B)} \sum_{(a:A)} fa = b$$

Classically (1.), (2.), (3.) are implied by nua, so equivalent.

Definition (Orton-Pitts nuaβ)

4. unitfi : $(\mathsf{coe}(\mathsf{unit}))a = (a, *)$
5. flipfi : $(\mathsf{coe}(\mathsf{flip}))(a, b, c) = (b, a, c)$

(1.)-(5.) together imply nuaβ. Given $(f, e) : A \simeq B$,

$$a : A \overset{(1.),(4.)}{\mapsto} (a, *) : \sum_{(a:A)} 1 \overset{(3.)^*}{\mapsto} (a, fa, \mathsf{refl}) : \sum_{(a:A)} \sum_{(b:B)} fa = b$$

$$\overset{(2.),(5.)}{\mapsto} (fa, a, \mathsf{refl}) : \sum_{(b:B)} \sum_{(a:A)} fa = b \overset{(3.)^*}{\mapsto} (fa, *) : \sum_{(b:B)} 1$$

Classically (1.), (2.), (3.) are implied by nua, so equivalent.

Definition (Orton-Pitts nuaβ)

4. unitfi : $(\mathsf{coe}(\mathsf{unit}))a = (a, *)$
5. flipfi : $(\mathsf{coe}(\mathsf{flip}))(a, b, c) = (b, a, c)$

(1.)-(5.) together imply nuaβ. Given $(f, e) : A \simeq B$,

$$a : A \overset{(1.),(4.)}{\mapsto} (a, *) : \sum_{(a:A)} 1 \overset{(3.)^*}{\mapsto} (a, fa, \mathsf{refl}) : \sum_{(a:A)} \sum_{(b:B)} fa = b$$

$$\overset{(2.),(5.)}{\mapsto} (fa, a, \mathsf{refl}) : \sum_{(b:B)} \sum_{(a:A)} fa = b \overset{(3.)^*}{\mapsto} (fa, *) : \sum_{(b:B)} 1$$

$$\overset{(1.),(4.)}{\mapsto} fa : B$$

The Orton-Pitts reformulation

Definition (Orton-Pitts univalence)

1. unit : $A = \sum(a : A), 1$
2. flip : $\sum(a : A), \sum(b : B), C = \sum(b : B), \sum(a : A), C$
3. contract : $\mathsf{isContr}(A) \to (A = 1)$
4. unitfi : $(\mathsf{coe}(\mathsf{unit}))a = (a, *)$
5. flipfi : $(\mathsf{coe}(\mathsf{flip}))(a, b, c) = (b, a, c)$

The Orton-Pitts reformulation

Definition (Orton-Pitts univalence)

1. unit : $A = \sum(a : A), 1$
2. flip : $\sum(a : A), \sum(b : B), C = \sum(b : B), \sum(a : A), C$
3. contract : $\mathsf{isContr}(A) \to (A = 1)$
4. unitfi : $(\mathsf{coe}(\mathsf{unit}))a = (a, *)$
5. flipfi : $(\mathsf{coe}(\mathsf{flip}))(a, b, c) = (b, a, c)$

and this is logically equivalent to ua.

The Orton-Pitts reformulation

Definition (Orton-Pitts univalence)

1. unit : $A = \sum(a : A), 1$
2. flip : $\sum(a : A), \sum(b : B), C = \sum(b : B), \sum(a : A), C$
3. contract : isContr$(A) \to (A = 1)$
4. unitfi : (coe(unit))$a = (a, *)$
5. flipfi : (coe(flip))$(a, b, c) = (b, a, c)$

and this is logically equivalent to ua. So we can rephrase,

The Orton-Pitts reformulation

Definition (Orton-Pitts univalence)

1. unit : $A = \sum(a : A), 1$
2. flip : $\sum(a : A), \sum(b : B), C = \sum(b : B), \sum(a : A), C$
3. contract : $\mathsf{isContr}(A) \to (A = 1)$
4. unitfi : $(\mathsf{coe}(\mathsf{unit}))a = (a, *)$
5. flipfi : $(\mathsf{coe}(\mathsf{flip}))(a, b, c) = (b, a, c)$

and this is logically equivalent to ua. So we can rephrase,

Question

In the presence of funext, do (1.)-(3.) above imply (4.) and (5.), for possibly 'modified' unit and flip.