Homotopy Type Theory for doing Category Theory

David Jaz Myers

Johns Hopkins University

March 26, 2020

Part 1: Homotopy Type Theory

How do you identify one thing with another?

It depends what type of things they are.

How do you identify one thing with another?

It depends what type of things they are.

- To identify the affine plane with $\mathbb{R}^2$, we need to choose a point to serve as the origin.

How do you identify one thing with another?

It depends what type of things they are.

- To identify the affine plane with $\mathbb{R}^2$, we need to choose a point to serve as the origin.
- To identify the tangent space of $\mathbb{S}^2$ at the point $(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$ with $\mathbb{R}^2$, we need to give a basis $\{\partial_1, \partial_2\}$ of it. Then we can identify any tangent vector

$$\mathrm{v} = \mathrm{v}^1\partial_1 + \mathrm{v}^2\partial_2 \quad \text{with} \quad \begin{bmatrix} v^1 \\ v^2 \end{bmatrix}$$

How do you identify one thing with another?

It depends what type of things they are.

- To identify the affine plane with $\mathbb{R}^2$, we need to choose a point to serve as the origin.
- To identify the tangent space of $\mathbb{S}^2$ at the point $(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$ with $\mathbb{R}^2$, we need to give a basis $\{\partial_1, \partial_2\}$ of it. Then we can identify any tangent vector
$$\mathsf{v} = \mathsf{v}^1\partial_1 + \mathsf{v}^2\partial_2 \quad \text{with} \quad \begin{bmatrix} v^1 \\ v^2 \end{bmatrix}$$
- To identify $\mathsf{H}^{\mathsf{n}}(\mathbb{S}^{\mathsf{n}}; \mathbb{Z})$ with $\mathbb{Z}$, we must choose an orientation for the n-sphere $\mathbb{S}^{\mathsf{n}}$.

How do you identify one thing with another?

It depends what type of things they are.

- To identify the affine plane with $\mathbb{R}^2$, we need to choose a point to serve as the origin.
- To identify the tangent space of $\mathbb{S}^2$ at the point $(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$ with $\mathbb{R}^2$, we need to give a basis $\{\partial_1, \partial_2\}$ of it. Then we can identify any tangent vector
$$\mathsf{v} = \mathsf{v}^1\partial_1 + \mathsf{v}^2\partial_2 \quad \text{with} \quad \begin{bmatrix} v^1 \\ v^2 \end{bmatrix}$$
- To identify $\mathsf{H}^n(\mathbb{S}^n; \mathbb{Z})$ with $\mathbb{Z}$, we must choose an orientation for the n-sphere $\mathbb{S}^n$.
- To identify the natural number n such that $\pi_4(\mathbb{S}^3)$ is isomorphic to $\mathbb{Z}_{/n}$ with the number 2, we need to prove that n equals 2.

Type Theory

A *type* is a type of mathematical thing.

Type theory gives *rules* for making new types and new *terms* of them.
It is a full foundation of mathematics, from scratch.

$$\text{a term} : \text{its type}$$

$$a : A$$

$$3 : \mathbb{N}$$

$$\mathbb{N} : \mathbf{Set}$$

$$T_pM : \mathbf{Vect}_{\mathbb{R}}$$

$$\mathbf{Vect}_{\mathbb{R}} : \mathbf{Type}$$

Judgements

$$a : A$$

is *not* a "proposition" – it is not up for debate.

Judgements

$$a : A$$

is *not* a "proposition" – it is not up for debate.

- Saying $3 : \mathbb{N}$ is a **judgement**: the fact that 3 is a *number* is just part of what we mean by 3.

Judgements

$$a : A$$

is *not* a "proposition" – it is not up for debate.

- Saying $3 : \mathbb{N}$ is a **judgement**: the fact that 3 is a *number* is just part of what we mean by 3.
- $3 : \mathbb{Z}$ and $3 : \mathbb{Q}$ are *different* 3*s*. For example, the second is a unit while the first is not.

Judgements

$$a : A$$

is *not* a "proposition" – it is not up for debate.

- Saying $3 : \mathbb{N}$ is a **judgement**: the fact that 3 is a *number* is just part of what we mean by 3.
- $3 : \mathbb{Z}$ and $3 : \mathbb{Q}$ are *different* 3*s*. For example, the second is a unit while the first is not.
- Similarly, we use "$a \equiv b$" to say that a is *judged* to be equal to b by *definition*.

Judgements

$$\mathsf{a} : \mathsf{A}$$

is *not* a "proposition" – it is not up for debate.

- Saying $3 : \mathbb{N}$ is a **judgement**: the fact that 3 is a *number* is just part of what we mean by 3.
- $3 : \mathbb{Z}$ and $3 : \mathbb{Q}$ are *different* 3*s*. For example, the second is a unit while the first is not.
- Similarly, we use "$\mathsf{a} \equiv \mathsf{b}$" to say that a is *judged* to be equal to b by *definition*. For example,

$$3 \equiv \mathbf{suc}(\mathbf{suc}(\mathbf{suc}(0))).$$

Identifications

For a and $b : A$,

$$a =_A b : \mathbf{Type}$$

is the type of *identifications* of a with b as elements of A.[a]

[a] We'll define this in a few slides

Identifications

For a and $b : A$,

$$a =_A b : \mathbf{Type}$$

is the type of *identifications* of a with b as elements of A.[a]

[a] We'll define this in a few slides

- For V and $W : \mathbf{Vect}_{\mathbb{R}}$, then $V = W$ is the type of $\mathbb{R}$-linear isomorphisms between V and W.

Identifications

For a and $b : A$,

$$a =_A b : \mathbf{Type}$$

is the type of *identifications* of a with b as elements of A.[a]

[a] We'll define this in a few slides

- For V and $W : \mathbf{Vect}_{\mathbb{R}}$, then $V = W$ is the type of $\mathbb{R}$-linear isomorphisms between V and W.
- For X and $Y : \mathbf{Set}$, then $X = Y$ is the type of bijections of between them.

Identifications

For a and $b : A$,

$$a =_A b : \mathbf{Type}$$

is the type of *identifications* of a with b as elements of A.[a]

[a] We'll define this in a few slides

- For V and $W : \mathbf{Vect}_{\mathbb{R}}$, then $V = W$ is the type of $\mathbb{R}$-linear isomorphisms between V and W.
- For X and $Y : \mathbf{Set}$, then $X = Y$ is the type of bijections of between them.
- For M and $N : \mathbf{Mfd}_{\infty}$, then $M = N$ is the type of smooth diffeomorphisms between them.

Identifications

For a and $b : A$,

$$a =_A b : \mathbf{Type}$$

is the type of *identifications* of a with b as elements of A.[a]

[a] We'll define this in a few slides

- For V and $W : \mathbf{Vect}_{\mathbb{R}}$, then $V = W$ is the type of $\mathbb{R}$-linear isomorphisms between V and W.
- For X and $Y : \mathbf{Set}$, then $X = Y$ is the type of bijections of between them.
- For M and $N : \mathbf{Mfd}_{\infty}$, then $M = N$ is the type of smooth diffeomorphisms between them.
- For n and $m : \mathbb{N}$, then $n = m$ has an element if and only if n equals m – there is at most one way to identify two natural numbers.

Identifications

For a and $b : A$,

$$a =_A b : \textbf{Type}$$

is the type of *identifications* of a with b as elements of A.[a]

[a] We'll define this in a few slides

- For V and $W : \mathbf{Vect}_{\mathbb{R}}$, then $V = W$ is the type of $\mathbb{R}$-linear isomorphisms between V and W.
- For X and $Y : \mathbf{Set}$, then $X = Y$ is the type of bijections of between them.
- For M and $N : \mathbf{Mfd}_{\infty}$, then $M = N$ is the type of smooth diffeomorphisms between them.
- For n and $m : \mathbb{N}$, then $n = m$ has an element if and only if n equals m – there is at most one way to identify two natural numbers.

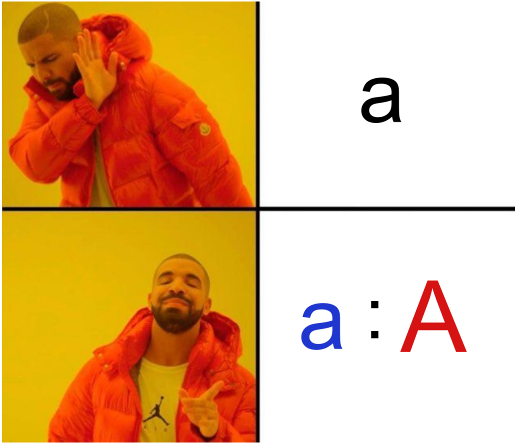
Axiom (Univalence)

If X and Y are types, then $X = Y$ is the type of equivalences of X with Y.

Functions

Every thing is a certain kind of thing.

- In a type theory, every free variable must be annotated with its type.



Functions

Every thing is a certain kind of thing.

- In a type theory, every free variable must be annotated with its type.
- Given types A and B depending on A,

  $$(a : A) \to B(a) \qquad \text{or, sometimes, } \Pi_{a:A} B(a)$$

  is the **type of functions from** A **to** B. (We write $A \to B$ if B doesn't depend on A.)

Functions

Every thing is a certain kind of thing.

- In a type theory, every free variable must be annotated with its type.
- Given types A and B depending on A,

  $$(a : A) \to B(a) \qquad \text{or, sometimes, } \Pi_{a:A} B(a)$$

  is the **type of functions from** A **to** B. (We write $A \to B$ if B doesn't depend on A.)
- To define a function $f : (a : A) \to B(a)$, assume a free $a : A$, and write down an element $f(a) : B(a)$; then f is the function $a \mapsto f(a)$.

Functions

Every thing is a certain kind of thing.

- In a type theory, every free variable must be annotated with its type.
- Given types A and B depending on A,

$$(a : A) \to B(a) \qquad \text{or, sometimes, } \Pi_{a:A} B(a)$$

  is the **type of functions from A to B**. (We write $A \to B$ if B doesn't depend on A.)
- To define a function $f : (a : A) \to B(a)$, assume a free $a : A$, and write down an element $f(a) : B(a)$; then f is the function $a \mapsto f(a)$.
- E.g., if M : **Manifold** and $p : M$, then we can define the tangent space T_pM : **VectorSpace**. So $p \mapsto T_pM : M \to$ **VectorSpace**.

Functions

Every thing is a certain kind of thing.

- In a type theory, every free variable must be annotated with its type.
- Given types A and B depending on A,

$$(a : A) \to B(a) \qquad \text{or, sometimes, } \Pi_{a:A} B(a)$$

  is the **type of functions from** A **to** B. (We write $A \to B$ if B doesn't depend on A.)
- To define a function $f : (a : A) \to B(a)$, assume a free $a : A$, and write down an element $f(a) : B(a)$; then f is the function $a \mapsto f(a)$.
- E.g., if M : **Manifold** and $p : M$, then we can define the tangent space T_pM : **VectorSpace**. So $p \mapsto T_pM : M \to$ **VectorSpace**.
- Since for any $p : M$, we have that $0 : T_pM$, we get a function $p \mapsto 0 : (p : M) \to T_pM$ – the zero vector field.

Pairs

Given A : $\mathbf{Type}$ with $B(a)$ depending on $a : A$, then

$$(a : A) \times B(a) \qquad \text{or, sometimes, } \Sigma_{a:A} B(a)$$

is the type of pairs (a, b) with $a : A$ and $b : B(a)$.

Pairs

Given A : **Type** with $B(a)$ depending on $a : A$, then

$$(a : A) \times B(a) \qquad \text{or, sometimes, } \Sigma_{a:A} B(a)$$

is the type of pairs (a, b) with $a : A$ and $b : B(a)$.

- The type $(p : M) \times T_p M$ is the total space of tangent bundle.

Pairs

Given A : **Type** with $B(a)$ depending on $a : A$, then

$$(a : A) \times B(a) \qquad \text{or, sometimes, } \Sigma_{a:A} B(a)$$

is the type of pairs (a, b) with $a : A$ and $b : B(a)$.

- The type $(p : M) \times T_p M$ is the total space of tangent bundle.
- Note that $(p : M) \to T_p M$ is the type of sections to $(p, v) \mapsto p : (p : M) \times T_p M \to M$

Inductive Types: Natural Numbers

If a type A is an *inductive type*, we may assume that a free variable $a : A$ is of one several prescribed forms.

- We may assume a free natural number $n : \mathbb{N}$ is either of the form
  1. $n \equiv 0$, or
  2. $n \equiv \mathbf{suc}(m)$ with $m : \mathbb{N}$.

Inductive Types: Natural Numbers

If a type A is an *inductive type*, we may assume that a free variable $a : A$ is of one several prescribed forms.

- We may assume a free natural number $n : \mathbb{N}$ is either of the form
  1. $n \equiv 0$, or
  2. $n \equiv \mathbf{suc}(m)$ with $m : \mathbb{N}$.
- To define $+ : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$, we assume a free variable $n : \mathbb{N}$ and seek a function of type $\mathbb{N} \to \mathbb{N}$.

Inductive Types: Natural Numbers

If a type A is an *inductive type*, we may assume that a free variable $a : A$ is of one several prescribed forms.

- We may assume a free natural number $n : \mathbb{N}$ is either of the form
  1. $n \equiv 0$, or
  2. $n \equiv \mathbf{suc}(m)$ with $m : \mathbb{N}$.
- To define $+ : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$, we assume a free variable $n : \mathbb{N}$ and seek a function of type $\mathbb{N} \to \mathbb{N}$.
  1. If $n \equiv 0$, then we have $\mathrm{id} \equiv x \mapsto x : \mathbb{N} \to \mathbb{N}$, or

Inductive Types: Natural Numbers

If a type A is an *inductive type*, we may assume that a free variable $a : A$ is of one several prescribed forms.

- We may assume a free natural number $n : \mathbb{N}$ is either of the form
  1. $n \equiv 0$, or
  2. $n \equiv \mathbf{suc}(m)$ with $m : \mathbb{N}$.
- To define $+ : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$, we assume a free variable $n : \mathbb{N}$ and seek a function of type $\mathbb{N} \to \mathbb{N}$.
  1. If $n \equiv 0$, then we have $\mathrm{id} \equiv x \mapsto x : \mathbb{N} \to \mathbb{N}$, or
  2. If $n \equiv \mathbf{suc}(m)$, then we have $x \mapsto \mathbf{suc}(x + m) : \mathbb{N} \to \mathbb{N}$

Inductive Types: Natural Numbers

If a type A is an *inductive type*, we may assume that a free variable $a : A$ is of one several prescribed forms.

- We may assume a free natural number $n : \mathbb{N}$ is either of the form
  1. $n \equiv 0$, or
  2. $n \equiv \mathbf{suc}(m)$ with $m : \mathbb{N}$.
- To define $+ : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$, we assume a free variable $n : \mathbb{N}$ and seek a function of type $\mathbb{N} \to \mathbb{N}$.
  1. If $n \equiv 0$, then we have $\mathrm{id} \equiv x \mapsto x : \mathbb{N} \to \mathbb{N}$, or
  2. If $n \equiv \mathbf{suc}(m)$, then we have $x \mapsto \mathbf{suc}(x + m) : \mathbb{N} \to \mathbb{N}$

  In total, we have

$$n \mapsto \begin{cases} x \mapsto x & \text{if } n \equiv 0 \\ x \mapsto \mathbf{suc}(x + m) & \text{if } n \equiv \mathbf{suc}(m). \end{cases} : \mathbb{N} \to (\mathbb{N} \to \mathbb{N})$$

The Type of Identifications

Given any two terms $a, b : A$, we have a **type** $a =_A b$ **of identifications of** a **with** b.

- We may assume that free variables $b : A$ and $p : a =_A b$ are of the form
  1. $\mathsf{refl} : a =_A a$.

The Type of Identifications

Given any two terms $a, b : A$, we have a **type** $a =_A b$ **of identifications of** a **with** b.

- We may assume that free variables $b : A$ and $p : a =_A b$ are of the form
  1. $\mathsf{refl} : a =_A a$.
- To define **sym** : $(a, b : A) \to a =_A b \to b =_A a$, assume that $a, b : A$ and $p : a =_A b$ are free variables.

The Type of Identifications

Given any two terms $a, b : A$, we have a **type** $a =_A b$ **of identifications of** a **with** b.

- We may assume that free variables $b : A$ and $p : a =_A b$ are of the form
  1. $\mathsf{refl} : a =_A a$.
- To define **sym** : $(a, b : A) \to a =_A b \to b =_A a$, assume that $a, b : A$ and $p : a =_A b$ are free variables.
  1. If $b \equiv a$ and $p \equiv \mathsf{refl}$, then $\mathsf{refl} : b =_A a$.

The Type of Identifications

Given any two terms $a, b : A$, we have a **type** $a =_A b$ **of identifications of** a **with** b.

- We may assume that free variables $b : A$ and $p : a =_A b$ are of the form
  1. $\mathsf{refl} : a =_A a$.
- To define **sym** $: (a, b : A) \to a =_A b \to b =_A a$, assume that $a, b : A$ and $p : a =_A b$ are free variables.
  1. If $b \equiv a$ and $p \equiv \mathsf{refl}$, then $\mathsf{refl} : b =_A a$.

  So,

$$a, b, p \mapsto \begin{cases} \mathsf{refl} & \text{if } p \equiv \mathsf{refl} \end{cases} : (a, b : A) \to a =_A b \to b =_A a.$$

Composing Identifications

Proposition

Given $p : a = b$ and $q : b = c$, we get an identification $p \bullet q : a = c$.

Proposition

Given $p : a = b$, we have $\mathsf{reflleft}_p : \mathsf{refl} \bullet p = p$, and similarly on the right.

Proposition

Given $p : a = b$, $q : b = c$, and $r : c = d$, we have an identification

$$\mathsf{assoc}_{p,q,r} : (p \bullet q) \bullet r = p \bullet (q \bullet r)$$

Composing Identifications

Proposition

Given $p : a = b$ and $q : b = c$, we get an identification $p \bullet q : a = c$.

Proposition

Given $p : a = b$, we have $\mathsf{reflleft}_p : \mathsf{refl} \bullet p = p$, and similarly on the right.

Proposition

Given $p : a = b$, $q : b = c$, and $r : c = d$, we have an identification

$$\mathsf{assoc}_{p,q,r} : (p \bullet q) \bullet r = p \bullet (q \bullet r)$$

Proposition

. . . and as my coherences as you like!

Every Function a Functor

Proposition

Let $f : A \to B$ be a function and suppose $p : a_1 = a_2$. Then we have

$$f_* p : f(a_1) = f(a_2).$$

Proposition

Let $f : A \to B$ be a function and suppose $p : a_1 = a_2$ and $q : a_2 = a_3$. Then we have

$$\mathsf{funct}_{p,q} : f_*(p \bullet q) = f_* p \bullet f_* q.$$

Identifying Functions and Pairs

Proposition

Let $f, g : (a : A) \to B(a)$ be functions. Then

$$(f = g) = (a : A) \to (fa = ga).$$

Identifying Functions and Pairs

Proposition

Let $f, g : (a : A) \to B(a)$ be functions. Then

$$(f = g) = (a : A) \to (fa = ga).$$

Proposition

Let $(a_1, b_1), (a_2, b_2) : (a : A) \times B(a)$ be pairs. Then

$$((a_1, b_1) = (a_2, b_2)) = (p : a_1 = a_2) \times (B_* p(b_1) = b_2).$$

Identifying Magmas

Definition

A *magma* is a type A together with a binary operation $+ : A \times A \to A$:

$$\mathbf{Magma} :\equiv (A : \mathbf{Type}) \times (A \times A \to A).$$

Let $\mathbf{BinOp}(A) :\equiv (A \times A) \to A$.

$$((A, +) = (B, \oplus))$$

Identifying Magmas

Definition

A *magma* is a type A together with a binary operation $+ : A \times A \to A$:

$$\mathbf{Magma} :\equiv (A : \mathbf{Type}) \times (A \times A \to A).$$

Let $\mathbf{BinOp}(A) :\equiv (A \times A) \to A$.

$$\begin{aligned} &((A, +) = (B, \oplus)) \\ = &\ (e : A = B) \times (\mathbf{BinOp}_* e(+) = \oplus) \end{aligned}$$

Identifying Magmas

Definition

A *magma* is a type A together with a binary operation $+ : A \times A \to A$:

$$\mathbf{Magma} :\equiv (A : \mathbf{Type}) \times (A \times A \to A).$$

Let $\mathbf{BinOp}(A) :\equiv (A \times A) \to A$.

$$\begin{aligned}
&((A, +) = (B, \oplus)) \\
&= (e : A = B) \times (\mathbf{BinOp}_* e(+) = \oplus) \\
&= (e : A = B) \times ((b_1, b_2) : B \times B) \to (\mathbf{BinOp}_* e(+)(b_1, b_2) = b_1 \oplus b_2)
\end{aligned}$$

Identifying Magmas

Definition

A *magma* is a type A together with a binary operation $+ : A \times A \to A$:

$$\mathbf{Magma} :\equiv (A : \mathbf{Type}) \times (A \times A \to A).$$

Let $\mathbf{BinOp}(A) :\equiv (A \times A) \to A$.

$$\begin{aligned}
&((A, +) = (B, \oplus)) \\
&= (e : A = B) \times (\mathbf{BinOp}_* e(+) = \oplus) \\
&= (e : A = B) \times ((b_1, b_2) : B \times B) \to (\mathbf{BinOp}_* e(+)(b_1, b_2) = b_1 \oplus b_2) \\
&= (e : A = B) \times ((b_1, b_2) : B \times B) \to (e(e^{-1} b_1 + e^{-1} b_2) = b_1 \oplus b_2)
\end{aligned}$$

Identifying Magmas

Definition

A *magma* is a type A together with a binary operation $+ : A \times A \to A$:

$$\mathbf{Magma} :\equiv (A : \mathbf{Type}) \times (A \times A \to A).$$

Let $\mathbf{BinOp}(A) :\equiv (A \times A) \to A$.

$$\begin{aligned}
&((A, +) = (B, \oplus)) \\
&= (e : A = B) \times (\mathbf{BinOp}_* e(+) = \oplus) \\
&= (e : A = B) \times ((b_1, b_2) : B \times B) \to (\mathbf{BinOp}_* e(+)(b_1, b_2) = b_1 \oplus b_2) \\
&= (e : A = B) \times ((b_1, b_2) : B \times B) \to (e(e^{-1} b_1 + e^{-1} b_2) = b_1 \oplus b_2) \\
&= (e : A = B) \times ((b_1, b_2) : B \times B) \to (e^{-1} b_1 + e^{-1} b_2 = e^{-1}(b_1 \oplus b_2))
\end{aligned}$$

Identifying Magmas

Definition

A *magma* is a type A together with a binary operation $+ : A \times A \to A$:

$$\textbf{Magma} :\equiv (A : \textbf{Type}) \times (A \times A \to A).$$

Let $\textbf{BinOp}(A) :\equiv (A \times A) \to A$.

$$
\begin{aligned}
&((A, +) = (B, \oplus)) \\
&= (e : A = B) \times (\textbf{BinOp}_* e(+) = \oplus) \\
&= (e : A = B) \times ((b_1, b_2) : B \times B) \to (\textbf{BinOp}_* e(+)(b_1, b_2) = b_1 \oplus b_2) \\
&= (e : A = B) \times ((b_1, b_2) : B \times B) \to (e(e^{-1} b_1 + e^{-1} b_2) = b_1 \oplus b_2) \\
&= (e : A = B) \times ((b_1, b_2) : B \times B) \to (e^{-1} b_1 + e^{-1} b_2 = e^{-1}(b_1 \oplus b_2)) \\
&= (e : A = B) \times ((a_1, a_2) : A \times A) \to (ea_1 \oplus ea_2 = e(a_1 + a_2))
\end{aligned}
$$

Contractible Types and Equivalences

Definition

For a function $f : A \to B$ and $b : B$, its *fiber* is the type

$$\mathrm{fib}_f(b) :\equiv (a : A) \times (f(a) = b)$$

together with the map $(a, p) \mapsto a : \mathrm{fib}_f(b) \to A$.

Definition

A *center of contraction* for a type A is an element $c : A$ such that for every other element $a : A$, we have an identification of a with c.

$$\mathbf{Contr}(A) :\equiv (c : A) \times \big((a : A) \to (a = c)\big)$$

Definition

A map $f : A \to B$ is an *equivalence* if its fibers are contractible:

$$\mathbf{Equiv}(f) :\equiv (y : Y) \to \mathbf{Contr}(\mathrm{fib}_f(y))$$

Propositions, Sets, and More

Lemma (UFP)

For any two centers of contraction $c, d : \textbf{Contr}(A)$, $c = d$ is contractible.

Definition

- A *proposition* is a type P such that for any $a, b : P$, $a = b$ is contractible.

Propositions, Sets, and More

Lemma (UFP)

For any two centers of contraction $c, d : \mathbf{Contr}(A)$, $c = d$ is contractible.

Definition

- A *proposition* is a type P such that for any $a, b : P$, $a = b$ is contractible.
- A *set* is a type S such that for any $a, b : S$, $a = b$ is a proposition.

Propositions, Sets, and More

Lemma (UFP)

For any two centers of contraction $c, d : \mathbf{Contr}(A)$, $c = d$ is contractible.

Definition

- A *proposition* is a type P such that for any $a, b : P$, $a = b$ is contractible.
- A *set* is a type S such that for any $a, b : S$, $a = b$ is a proposition.
- A *groupoid* is a type G such that for any $a, b : G$, $a = b$ is a set.

Propositions, Sets, and More

Lemma (UFP)

For any two centers of contraction $c, d : \mathbf{Contr}(A)$, $c = d$ is contractible.

Definition

- A *proposition* is a type P such that for any $a, b : P$, $a = b$ is contractible.
- A *set* is a type S such that for any $a, b : S$, $a = b$ is a proposition.
- A *groupoid* is a type G such that for any $a, b : G$, $a = b$ is a set.
- ...
- An n-*type* is a type X such that for any $a, b : X$, $a = b$ is an $(n-1)$-type (with -2-types being contractible).

Truncation

Theorem (UFP)

For any type X, there is a proposition $\|X\|$ and a map $|\cdot| : X \to \|X\|$ and such that

$$\begin{tikzcd} X \arrow[r, "\forall"] \arrow[d, "|\cdot|"'] & P \\ \|X\| \arrow[ur, dashed, "\exists!"'] & \end{tikzcd}$$

for any proposition P.

Truncation

Theorem (UFP)

For any type X, there is a proposition $\|X\|$ and a map $|\cdot| : X \to \|X\|$ and such that

$$\begin{tikzcd} X \arrow[r, "\forall"] \arrow[d, "|\cdot|"'] & P \\ \|X\| \arrow[ur, dashed, "\exists!"'] & \end{tikzcd}$$

for any proposition P.

- $\|(a : A) \times B(a)\|$ represents the proposition $\exists a : A.\, B(a)$.
- If $B(a)$ is a proposition for all $a : A$, then $(a : A) \to B(a)$ represents the proposition $\forall a : A.\, B(a)$.

Part 2: Category Theory

Pre-categories

Definition

A *pre-category* $\mathcal{C}$ consists of:

- A type $\mathsf{ob}\,\mathcal{C}$ of *objects*.
- For each $A, B : \mathsf{ob}\,\mathcal{C}$, a set $\mathcal{C}(A, B)$ of morphisms.
- Composition functions $\circ : \mathcal{C}(B, C) \to \mathcal{C}(A, B) \to \mathcal{C}(A, C)$.
- Identities $\mathsf{id}_A : \mathcal{C}(A, A)$.
- All the usual identities.

…And the Obvious Morphisms

An "obvious morphism" is one whose definition can be derived from the definition of the objects of a category.

...And the Obvious Morphisms

An "obvious morphism" is one whose definition can be derived from the definition of the objects of a category.

Definition

A pre-category $\mathcal{C}$ is a *category* if the map

$$\mathsf{idtoiso} : (A = B) \to (A \cong_{\mathcal{C}} B)$$

defined inductively by $\mathsf{refl} \mapsto \mathsf{id}_A$ is an equivalence.

...And the Obvious Morphisms

An "obvious morphism" is one whose definition can be derived from the definition of the objects of a category.

Definition

A pre-category $\mathcal{C}$ is a *category* if the map

$$\mathsf{idtoiso} : (A = B) \to (A \cong_{\mathcal{C}} B)$$

defined inductively by $\mathsf{refl} \mapsto \mathsf{id}_A$ is an equivalence.

A category is a pre-category where the identifications between objects are precisely the isomorphisms.

- The category of sets.
- The category of groups, abelian groups, rings, vector spaces...
- The category of topological spaces, smooth manifolds, schemes...
- Functors from a pre-category to a category form a category – hence any presheaf category.

Universal Properties are Properties

Proposition

Let $\mathcal{C}$ be a category. Then the type

$$\mathbf{Terminal}(A) :\equiv \forall X : \mathrm{ob}\,\mathcal{C}\,.\,\exists! f : \mathcal{C}(X, A).$$

is a proposition.

Corollary

The type of limits to a diagram is a proposition.

Categorical Choice

Theorem (AKS)

If $F : \mathcal{C} \to \mathcal{D}$ is a fully faithful functor between categories, then for any $Y : \mathcal{D}$,

$$(X : \mathrm{ob}\,\mathcal{C}) \times (FX \cong Y)$$

is a proposition. That is,

$$\exists X : \mathrm{ob}\,\mathcal{C}\,.\,(FX \cong Y) = (X : \mathrm{ob}\,\mathcal{C}) \times (FX \cong Y).$$

Categorical Choice

Theorem (AKS)

If $F : \mathcal{C} \to \mathcal{D}$ is a fully faithful functor between categories, then for any $Y : \mathcal{D}$,

$$(X : \mathrm{ob}\,\mathcal{C}) \times (FX \cong Y)$$

is a proposition. That is,

$$\exists X : \mathrm{ob}\,\mathcal{C}\,.\,(FX \cong Y) = (X : \mathrm{ob}\,\mathcal{C}) \times (FX \cong Y).$$

Corollary

For categories $\mathcal{C}$ and $\mathcal{D}$, we have

$$(\mathcal{C} = \mathcal{D}) = (\mathcal{C} \simeq \mathcal{D}) = (F : \mathbf{Fun}(\mathcal{C}, \mathcal{D})) \times (F \text{ is ess. surj. fully faithful}).$$

No More "With a Choice of Pullbacks"

Theorem

Suppose that every diagram of shape $\mathcal{D}$ in $\mathcal{C}$ admits a limit. Then there is a functor $\lim : \mathbf{Fun}(\mathcal{C}^{\mathcal{D}}, \mathcal{C})$ taking a diagram to its limit.

Proof.

The category of diagrams with a limit is a full subcategory of diagrams. If every diagram has a limit, then this fully faithful functor is an equivalence. The composite of the inverse with the functor that projects out the limit is then the desired limit functor. $\square$

References

- *Homotopy Type Theory*, Univalent Foundations Project, 2013
- *Univalent Categories and the Rezk Completion*, Ahrens, Kapulkin, Shulman, 2014