HOTTEST SUMMER SCHOOL

EMILY RIEHL

Contents

July 29: The fundamental theorem of identity types 1
- What are identity types really? 1
- Families of equivalences 2
- The fundamental theorem 3
- Equality on $\mathbb{N}$ 4
- Disjointness of coproducts 4
- Embeddings 5
- Identity types of contractible types 6

August 1: Propositions, sets, and general truncation levels 6
- Propositions 6
- Sets 8
- General truncation levels 9
- Subtypes 10
- Truncated maps 10

August 5: Function extensionality & Universal properties 11
- Function extensionality 12
- The type theoretic principle of choice 13
- Universal properties 14
- The universal property of identity types 15
- Univalence 15
- References 16

July 29: The fundamental theorem of identity types

What are identity types really?

On the one hand we have Per Martin-Löf's rules, which characterize the identity type family for *any* type in any context. These rules are summarized as follows: given a type A in context Γ, the identity type family $=_A : A \to A \to \mathcal{U}$ is freely generated by the reflexivity terms $\Gamma, x : A \vdash \mathrm{refl}_x : x =_A x$.

On the other hand, for a *particular* type, we might know what we want the identity type family should be. We might call these *observational equality* type families to distinguish them from Martin-Löf's identity types.

ex. In the empty context we have the type $\mathbb{N}$ of natural numbers. By induction, we may define the observational equality type family $\mathrm{Eq}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N} \to \mathcal{U}$ by

$$\mathrm{Eq}_{\mathbb{N}}(0_{\mathbb{N}}, 0_{\mathbb{N}}) \doteq \mathbb{1} \quad \mathrm{Eq}_{\mathbb{N}}(\mathrm{succ}_{\mathbb{N}}(n), 0_{\mathbb{N}}) \doteq \emptyset \quad \mathrm{Eq}_{\mathbb{N}}(0, \mathrm{succ}_{\mathbb{N}}(n)) \doteq \emptyset \quad \mathrm{Eq}_{\mathbb{N}}(\mathrm{succ}_{\mathbb{N}}(m), \mathrm{succ}_{\mathbb{N}}(n)) \doteq \mathrm{Eq}_{\mathbb{N}}(m, n).$$

Date: Summer 2022.

The author is grateful to receive support from the National Science Foundation via the grant DMS-1652600 for various activities involving the HoTTEST Summer School, including the preparation of these notes.

ex. Given types A and B in context Γ we may define the observational equality type family $\mathrm{Eq}^+_{A,B} : (A + B) \to (A + B) \to \mathcal{U}$ for the coproduct type $\Gamma \vdash A + B$ type by induction by

$$\begin{aligned}
\mathrm{Eq}^+_{A,B}(\mathrm{inl}(a), \mathrm{inl}(a')) &\doteq (a = a') \\
\mathrm{Eq}^+_{A,B}(\mathrm{inl}(a), \mathrm{inr}(b)) &\doteq \emptyset \\
\mathrm{Eq}^+_{A,B}(\mathrm{inr}(b), \mathrm{inr}(a)) &\doteq \emptyset \\
\mathrm{Eq}^+_{A,B}(\mathrm{inr}(b), \mathrm{inr}(b')) &\doteq (b = b')
\end{aligned}$$

Our goal today will be to introduce a general strategy for showing that an observational equality type family for a given type is equivalent to the identity type family. Before stating the theorem that describes the universal property of identity type families, we firstly consider equivalences between type families more generally.

Families of equivalences

For any family of maps $f : \prod_{x:A} B(x) \to C(x)$ there is a map

$$\mathrm{tot}(f) : \sum_{x:A} B(x) \to \sum_{x:A} C(x)$$

defined by $\lambda(x, y).(x, f(x, y))$.

Theorem. *For any family of maps $f : \prod_{x:A} B(x) \to C(x)$ the following are logically equivalent:*

*(i) The family f is a **family of equivalences**: for each $x : A$ the map $f(x) : B(x) \to C(x)$ is an equivalence.*
(ii) The map $\mathrm{tot}(f)$ is an equivalence.

Proof. Recall equivalences are contractible maps: meaning maps whose fibers are all contractible. So it suffices to show that $f(x)$ is a contractible map for each x if and only if $\mathrm{tot}(f)$ is a contractible map. For this, we must show for each $x : A$ and $c : C(x)$ that $\mathrm{fib}_{f(x)}(c)$ is contractible if and only if $\mathrm{fib}_{\mathrm{tot}(f)}(x, c)$ is contractible. But in fact these fibers are always equivalent, as the following lemma shows. □

Lemma. *For any family of maps $f : \prod_{x:A} B(x) \to C(x)$ and any term $t : \sum_{x:A} C(x)$ there is an equivalence*

$$\mathrm{fib}_{\mathrm{tot}(f)}(t) \simeq \mathrm{fib}_{f(\mathrm{pr}_1(t))}\mathrm{pr}_2(t).$$

Proof. Define $\phi : \prod_{t:\sum_{x:A} C(x)} \mathrm{fib}_{\mathrm{tot}(f)}(t) \to \mathrm{fib}_{f(\mathrm{pr}_1(t))}\mathrm{pr}_2(t)$ by pattern matching by taking

$$((x, y), \mathrm{refl}) : \sum_{s:\sum_{x:A} B(x)} \mathrm{tot}(f)s = (x, f(x, y))$$

to $(y, \mathrm{refl}) : \sum_{z:B(x)} f(x, z) = f(x, y)$. We construct an inverse $\psi : \prod_{t:\sum_{x:A} C(x)} \mathrm{fib}_{f(\mathrm{pr}_1(t))}\mathrm{pr}_2(t) \to \mathrm{fib}_{\mathrm{tot}(f)}(t)$ by pattern matching

$$\psi(x, f(x, y), y, \mathrm{refl}) \coloneqq ((x, y), \mathrm{refl})$$

and homotopies by pattern matching in which case both homotopies reduce to refl. □

Remark. More generally, we might consider a map $f : A \to B$, a family of types C over A, a family of types D over B, and a family of maps $g : \prod_{x:A} C(x) \to D(f(x))$. When f is an equivalence, then g defines a family of equivalences over f if and only if the map

$$\mathrm{tot}_f(g) : \sum_{x:A} C(x) \to \sum_{y:B} D(y)$$

defined by $\mathrm{tot}_f(g)(x, z) \coloneqq (f(x), g(x, z))$ is an equivalence. This result, together with others in this section, can be used to show that equivalent types have equivalent identity types, for instance. See [R, §11.1] for more.

The fundamental theorem

The fundamental theorem of identity types characterizes the "one-sided" identity types up to equivalence, where one of the variables is moved into the context:

$$\frac{\Gamma \vdash a : A}{\Gamma, x : A \vdash a =_A x \text{ type}} \qquad \frac{\Gamma \vdash a : A}{\Gamma \vdash \text{refl}_a : a =_A a}$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma, x : A, p : a =_A x \vdash P(x,p) \text{ type}}{\Gamma \vdash \text{path-ind}_a : P(a,\text{refl}_a) \to \prod_{x:A} \prod_{p:a=_A x} P(x,p)} \qquad \frac{\Gamma \vdash a : A \qquad \Gamma, x : A, p : a =_A x \vdash P(x,p) \text{ type}}{\Gamma \vdash \text{path-ind}_a(q,a,\text{refl}_a) \doteq q : P(a,\text{refl}_a)}$$

These rules are summarized as follows: given a type A and a term $a : A$ in any context Γ, the identity type family $\Gamma, x : A \vdash a =_A x$ type is freely generated by the reflexivity term $\Gamma \vdash \text{refl}_a : a =_A a$.

Given a type A and a term $a : A$ in context Γ, the identity type provides a type family $\lambda x.a =_A x : A \to \mathcal{U}$ in context Γ equipped with a special term $\text{refl}_a : a =_A a$. The fundamental theorem of identity types provides necessary and sufficient conditions for a type family $E : A \to \mathcal{U}$ and term $r : E(a)$ to define a family of equivalences $\prod_{x:A}(a = x) \simeq E(x)$ by $(a, \text{refl}) \mapsto r$.

defn. Given a type A and a term $a : A$ a **(unary) identity system** on A at a is given by a type family $E : A \to \mathcal{U}$ and a term $r : E(a)$ so that for any family of types $P : \sum_{x:A} E(x) \to \mathcal{U}$ the function

$$\text{ev}_{a,r} : \prod_{x:A} \prod_{y:E(x)} P(x,y) \to P(a,r)$$

has a section.

That is, if (E, r) is an identity system at (A, a) and P is a family of types over $x : A$ and $y : E(x)$ then for each $p : P(a,r)$ there is a dependent function $f : \prod_{x:A} \prod_{y:E(x)} P(x,y)$ with an identification $h : f(a,r) = p$. This is a variant of the path induction principal where the computation rule is given by an identification rather than by a definitional equality.

Theorem (fundamental theorem of identity types)**.** *Let A be a type, let $a : A$, and let $E : A \to \mathcal{U}$ with $r : E(a)$. Define a family of maps*

$$\text{path-ind}_a r : \prod_{x:A}(a = x) \to E(x)$$

by $\text{refl}_a \mapsto r$. Then the following are logically equivalent:

(i) $\text{path-ind}_a r$ *is a family of equivalences.*
(ii) The total space $\sum_{x:A} E(x)$ is contractible with (a, r) as its center of contraction.
(iii) The family E is an identity system.

Proof. By our theorem characterizing families of equivalences, $\text{path-ind}_a r$ is a family of equivalences iff and only if it induces an equivalence on total spaces

$$\left(\sum_{x:A} a = x\right) \simeq \left(\sum_{x:A} E(x)\right).$$

The left-hand type is contractible so this is the case if and only if $\sum_{x:A} E(x)$ is contractible. This proves the equivalence of (i) and (ii).

For the equivalence of (ii) and (iii) consider the commutative triangle:

$$\begin{array}{ccc} \prod_{t:\sum_{x:A} E(x)} P(t) & \xrightarrow{\text{ev-pair}} & \prod_{x:A} \prod_{y:E(x)} P(x,y) \\ & {\scriptstyle \text{ev-pt}_{(a,r)}} \searrow \quad \swarrow {\scriptstyle \text{ev}_{(a,r)}} & \\ & P(a,r) & \end{array}$$

By Σ-induction, the top map has a section. It follows that the left map has a section if and only if the right map has a section. The left-hand section is the universal property called singleton induction that is satisfied if and only if the type $\sum_{x:A} E(x)$ is contractible, while the right-hand section is the universal property of an identity system. This proves the equivalence of (ii) and (iii). □

Equality on $\mathbb{N}$

As our first application recall the observational equality type family $\mathrm{Eq}_{\mathbb{N}} : \mathbb{N} \to \mathbb{N} \to \mathcal{U}$ satisfying

$$\mathrm{Eq}_{\mathbb{N}}(0_{\mathbb{N}}, 0_{\mathbb{N}}) \doteq \mathbb{1} \quad \mathrm{Eq}_{\mathbb{N}}(\mathrm{succ}_{\mathbb{N}}(n), 0_{\mathbb{N}}) \doteq \emptyset \quad \mathrm{Eq}_{\mathbb{N}}(0, \mathrm{succ}_{\mathbb{N}}(n)) \doteq \emptyset \quad \mathrm{Eq}_{\mathbb{N}}(\mathrm{succ}_{\mathbb{N}}(m), \mathrm{succ}_{\mathbb{N}}(n)) \doteq \mathrm{Eq}_{\mathbb{N}}(m, n).$$

We previously showed that this type family is logically equivalent to the identity type family for the natural numbers, but we can do better. Using the reflexivity term $\text{refl-}\mathrm{Eq}_{\mathbb{N}} : \prod_{n:\mathbb{N}} \mathrm{Eq}_{\mathbb{N}}(n, n)$ defined by induction on $n : \mathbb{N}$, there is a canonical map $\text{eq-id} : \prod_{m,n:\mathbb{N}}(m = n) \to \mathrm{Eq}_{\mathbb{N}}(m, n)$ defined by path induction.

Theorem. *For all $m, n : \mathbb{N}$ the canonical map*

$$\text{eq-id} : (m = n) \to \mathrm{Eq}_{\mathbb{N}}(m, n)$$

is an equivalence.

Proof. It suffices to show for each $m : \mathbb{N}$ that the type

$$\sum_{n:\mathbb{N}} \mathrm{Eq}_{\mathbb{N}}(m, n)$$

is contractible with $(m, \text{refl-}\mathrm{Eq}_{\mathbb{N}}(m))$ as the center of contraction.

The contracting homotopy

$$\gamma(m) : \prod_{n:\mathbb{N}} \prod_{e:\mathrm{Eq}_{\mathbb{N}}(m,n)} (m, \text{refl-}\mathrm{Eq}_{\mathbb{N}}(m)) = (n, e)$$

is defined by induction on $m, n : \mathbb{N}$ from the base case $\gamma(0, 0, \star) :\equiv \mathrm{refl}$. If either m or n is 0 and the other is a successor we can define this using ex-falso.

In the inductive step we seek an identification $\gamma(m+1, n+1, e) : (m+1, \text{refl-}\mathrm{Eq}_{\mathbb{N}}(m+1)) = (n+1, e)$. To define this we use the map

$$\lambda(n, e).(n+1, e) : \sum_{n:\mathbb{N}} \mathrm{Eq}_{\mathbb{N}}(m, n) \to \sum_{n:\mathbb{N}} \mathrm{Eq}_{\mathbb{N}}(m+1, n).$$

Since this map carries $(m, \text{refl-}\mathrm{Eq}_{\mathbb{N}}(m))$ to $(m+1, \text{refl-}\mathrm{Eq}_{\mathbb{N}}(m+1))$, $\mathrm{ap}_{\lambda(n,e).(n+1,e)}\gamma(m, n, e)$ provides the identification we seek. □

Exercise. Peano's axioms for the natural numbers assert that:

- Zero is not the successor of any natural number.
- The successor function is injective.

Prove that these hold for the type $\mathbb{N}$.

It it interesting to compare the set-theoretic axiomatization of the natural numbers, via Peano's postulates, with the type-theoretic axiomatization, using the rules for inductive types. The type theoretic induction principle is manifestly stronger than the set theoretic principle of mathematical induction, since the latter considers only predicates $P : \mathbb{N} \to \mathrm{Prop}_{\mathcal{U}}$—valued in the *universe of propositions*, to be defined in the next lecture—while the latter allows arbitrary families of types $P\colon \mathbb{N} \to \mathcal{U}$. In particular, this strengthened induction principle allows us to define the observational equality type family, which can be used to prove the properties of the successor function mentioned above.

Disjointness of coproducts

For a second application, we characterize the identity types of coproducts.

Theorem. *Let A and B be types. Then for any $a, a' : A$ and $b, b' : B$ there are equivalences*

$$(\mathrm{inl}(a) = \mathrm{inl}(a')) \simeq (a = a')$$
$$(\mathrm{inl}(a) = \mathrm{inr}(b)) \simeq \emptyset$$
$$(\mathrm{inr}(b) = \mathrm{inl}(a)) \simeq \emptyset$$
$$(\mathrm{inr}(b) = \mathrm{inr}(b')) \simeq (b = b')$$

We follow our usual strategy first defining a type family

$$\mathrm{Eq}^+_{A,B} : (A + B) \to (A + B) \to \mathcal{U}$$

by induction by

$$\begin{aligned}
\mathrm{Eq}^+_{A,B}(\mathrm{inl}(a), \mathrm{inl}(a')) &\doteq (a = a') \\
\mathrm{Eq}^+_{A,B}(\mathrm{inl}(a), \mathrm{inr}(b)) &\doteq \emptyset \\
\mathrm{Eq}^+_{A,B}(\mathrm{inr}(b), \mathrm{inr}(a)) &\doteq \emptyset \\
\mathrm{Eq}^+_{A,B}(\mathrm{inr}(b), \mathrm{inr}(b')) &\doteq (b = b')
\end{aligned}$$

Again by induction there is a term $\mathrm{Eq}^+\text{-refl} : \prod_{z:A+B} \mathrm{Eq}^+_{A,B}(z, z)$ defined by refl in both cases. Thus there is a map

$$\text{eq-id} : \prod_{s,t:A+B} (s = t) \to \mathrm{Eq}^+_{A,B}(s, t)$$

defined by path induction.

Proposition. *For any $s : A + B$ the total space*

$$\sum_{t:A+B} \mathrm{Eq}^+_{A,B}(s, t)$$

is contractible with $(s, \mathrm{Eq}^+\text{-refl}_s)$ as its center of contraction.

Proof. By induction on s we have to consider two cases, of which we prove just one: that

$$\sum_{t:A+B} \mathrm{Eq}^+_{A,B}(\mathrm{inl}(a), t)$$

is contractible. From distributivity of dependent pairs of coproducts we have

$$\sum_{t:A+B} \mathrm{Eq}^+_{A,B}(\mathrm{inl}(a), t) \simeq \left(\sum_{x:A} \mathrm{Eq}^+_{A,B}(\mathrm{inl}(a), \mathrm{inl}(x))\right) + \left(\sum_{y:B} \mathrm{Eq}^+_{A,B}(\mathrm{inl}(a), \mathrm{inr}(y))\right)$$

$$\simeq \left(\sum_{x:A} a = x\right) + \left(\sum_{y:B} \emptyset\right) \simeq \left(\sum_{x:A} a = x\right) + \emptyset \simeq \sum_{x:A} a = x.$$

This last type is contractible so the first type is as claimed. Chasing through the equivalences, the center of contraction (a, refl_a) maps to the claimed center of contraction. □

By the fundamental theorem of identity types, this establishes the desired family of equivalences of types.

Embeddings

Our next application will show that equivalences are embeddings, defined as follows:

defn. An **embedding** is a map $f : A \to B$ that satisfies the property that

$$\mathrm{ap}_f : (x = y) \to (f(x) = f(y))$$

is an equivalence for every $x, y : A$.

Write

$$\text{is-emb}(f) :\equiv \prod_{x,y:A} \text{is-equiv}(\mathrm{ap}_f : (x = y) \to (f(x) = f(y))).$$

Theorem. *Equivalences are embeddings.*

Proof. Suppose $e : A \simeq B$ is an equivalence and $x : A$. We wish to show that

$$\mathrm{ap}_e : (x = y) \to (e(x) = e(y))$$

is an equivalence for every $y : A$. For this it suffices to show that $\sum_{y:A} e(x) = e(y)$ is contractible with center of contraction $\mathrm{ap}_e(\mathrm{refl}_x) \doteq \mathrm{refl}_{e(x)}$. We have an equivalence

$$\sum_{y:A} e(x) = e(y) \simeq \sum_{y:A} e(y) = e(x)$$

and the latter type is the fiber $\mathrm{fib}_e(e(x))$. Since e is an equivalence this fiber is contractible so the result follows. □

Identity types of contractible types

We suggest a final exercise, a repeat of an exercise that appeared on a previous worksheet, proving a result that we will make use of in the next lecture:

Exercise. Let $\mathbb{1}$ be the unit type. Show that its identity types are contractible.

August 1: Propositions, sets, and general truncation levels

Topologists are familiar with various spaces built by attaching cells. These cell complexes can also be described in homotopy type theory as higher inductive types.

For instance, we have types like $\emptyset$, $\mathbb{1}$, and $\mathbb{1} + \mathbb{1}$, which are built by freely adding points.

The circle S^1 can be built by freely attaching a basepoint and then a loop identifying the basepoint with itself.

Higher dimensional spheres can be built analogously: the 2-sphere is defined by attaching a free 2-dimensional loop from $\mathrm{refl}_{\mathrm{base}}$ to itself, while the 3-sphere is defined by attaching a free 3-dimensional loop from $\mathrm{refl}_{\mathrm{refl}_{\mathrm{base}}}$ to itself, and so on.

Other spaces that can be formed as cell complexes include the torus T, which is built from one point, two loops a and b, and a 2-dimensional path from $a \cdot b \cdot a^{-1} \cdot b^{-1}$ to refl. Certain spaces, like $\mathbb{R}P^\infty$ and $\mathbb{C}P^\infty$, require cells in arbitrarily large dimensions.

Today we will introduce a hierarchy that classifies all types according to their truncation level, which recovers the topological characterization of "homotopy n-types." Note that being truncated at level n does *not* correspond to the naive notion of having "dimension n." While the types $\emptyset$, $\mathbb{1}$, and $\mathbb{1} + \mathbb{1}$ are all 0-types, as one might expect, the types S^1, T, and $\mathbb{R}P^\infty$ are all 1-types, even though the latter are built using higher cells. The type $\mathbb{C}P^\infty$ is a 2-type, but S^2 is not. Indeed, only the lowest dimensional spheres S^0 and S^1 have finite truncation levels.

Finally, as we will learn, homotopy type theory extends this hierarchy downwards: $\emptyset$ and $\mathbb{1}$ are both -1-types, aka *propositions*, while $\mathbb{1}$ is also a -2-type, aka contractible.

Propositions

Recall a type A is **contractible** if it has a term of type

$$\text{is-contr}A := \Sigma_{c:A}\Pi_{x:A} c = x.$$

We now formally study propositions in homotopy type theory.

defn. A type A is a **proposition** if all of its identity types are contractible: i.e., if it comes with a term of type

$$\text{is-prop}(A) := \prod_{x,y:A} \text{is-contr}(x = y).$$

Given a universe $\mathcal{U}$ we define $\mathrm{Prop}_{\mathcal{U}}$ to be the type of all small propositions:

$$\mathrm{Prop}_{\mathcal{U}} := \sum_{X:\mathcal{U}} \text{is-prop}(X).$$

ex. We have shown that identity types of contractible types are contractible. Thus contractible types are propositions.

ex. The empty type is also a proposition, for ex-falso inhabits

$$\prod_{x,y:\emptyset} \text{is-contr}(x = y).$$

There are many equivalent ways to assert that a type is a proposition.

Proposition. *For a type A, the following are logically equivalent:*

(i) *A is a proposition.*
(ii) *Any two terms of type A can be identified: i.e., there is a dependent function in the type*

$$\prod_{x,y:A} x = y.$$

(iii) *The type A is contractible as soon as it is inhabited: i.e., there is a term of type $A \to \text{is-contr}(A)$.*
(iv) *The map $\text{const}_\star : A \to \mathbb{1}$ is an embedding.*

Proof. (i) clearly implies (ii), by using the center of contraction. Assuming (ii) we have $p : \prod_{x,y:A} x = y$. Note for any x that $p(x) : \prod_{y:A} x = y$ is a contracting homotopy onto x. Thus, we have a function

$$\lambda x.(x, p(x)) : A \to \text{is-contr}(A).$$

Next assume (iii) and consider a function $c : A \to \text{is-contr}(A)$. To prove

$$\prod_{x,y:A} \text{is-equiv}(\text{ap}_{\text{const}_\star} : (x = y) \to (\star = \star))$$

it suffices to prove

$$A \to \left(\prod_{x,y:A} \text{is-equiv}(\text{ap}_{\text{const}_\star} : (x = y) \to (\star = \star)) \right)$$

because then we can use this function f applied to one of the two terms $x, y : A$ to get the data we want. But now our new goal allows us to assume we have a term $a : A$ and so it follows from our assumption that A is contractible. Thus $\text{const}_\star : A \to \mathbb{1}$ is an equivalence and in particular an embedding. This proves (iv).

Finally, if $\text{const}_\star : A \to \mathbb{1}$ is an embedding, then the types $(x = y)$ and $(\star = \star)$ are equivalent. The latter type is contractible so the former must be as well. This proves that (iv) implies (i). □

The final step of the proof relied on the fact that the notion of contractibility is invariant under equivalence of types. The same is true for the propositions:

Lemma. *Suppose $e : A \simeq B$. Then*

$$\text{is-prop}A \leftrightarrow \text{is-prop}B.$$

Proof. Since e is an equivalence, e is an embedding, meaning that $\text{ap}_e : (x =_A y) \to (e(x) =_B e(y))$ is an equivalence. Now if B is a proposition then $(e(x) =_B e(y))$ is contractible which then implies that $(x =_A y)$ is contractible. Thus $\text{is-prop}(B) \to \text{is-prop}(A)$. The converse is proven similarly using the inverse equivalence to e. □

A useful feature of propositions is that logical equivalences become equivalences:

Proposition. *For propositions P and Q*

$$(P \simeq Q) \leftrightarrow (P \leftrightarrow Q).$$

Proof. Clearly we have $(P \simeq Q) \to (P \leftrightarrow Q)$ so the content is in the converse. Given $f : P \to Q$ and $g : Q \to P$ we obtain homotopies $f \circ g \sim \text{id}$ and $g \circ f \sim \text{id}$ using the fact that any two elements in P and Q can be identified. Thus, f and g are equivalence inverses. □

Sets

defn. A type A is a **set** if its identity types are propositions:

$$\text{is-set}(A) :\equiv \prod_{x,y:A} \text{is-prop}(x = y).$$

ex. The type of natural numbers is a set, since we have $(m = n) \simeq \text{Eq}_{\mathbb{N}}(m, n)$ and, by induction, the latter types are propositions.

Theorem. *For a type A the following are logically equivalent:*

(i) *A is a set.*
(ii) *A satisfies **axiom K**: that is, A comes with a term in the type*

$$\text{axiom-K}(A) :\equiv \prod_{x:A} \prod_{p:x=x} \text{refl}_x = p.$$

Proof. If A is a set then $x = x$ is a proposition so any two terms in it can be identified.

Conversely, if axiom-K holds then for any $p, q : x = y$ we can identify $p \cdot q^{-1}$ and refl_x and it follows that $p = q$ by composing identifications:

$$p = p \cdot \text{refl}_y = p \cdot (q^{-1} \cdot q) = (p \cdot q^{-1}) \cdot q = \text{refl}_x \cdot q = q.$$

This proves that $x = y$ is a proposition so A must be a set. □

The following result can be used to prove that a type A is a set.

Theorem. *Let A be a type and suppose $R : A \to A \to \mathcal{U}$ satisifes:*

(i) *Each $R(x, y)$ is a proposition.*
(ii) *R is reflexive, witnessed by $\rho : \prod_{x:A} R(x, x)$.*
(iii) *There is a map $R(x, y) \to (x = y)$ for all $x, y : A$.*

Then any family of maps $\prod_{x,y:A}(x = y) \to R(x, y)$ is a family of equivalences and A must be a set.

Proof. By hypothesis we have terms $f : \prod_{x,y:A} R(x, y) \to (x = y)$ and also $\text{path-ind}(\rho) : \prod_{x,y}(x = y) \to R(x, y)$. Since each $R(x, y)$ is a proposition we have a homotopy $\text{path-ind}(\rho)(x, y) \circ f(x, y) \sim \text{id}_{R(x,y)}$ proving that $R(x, y)$ is a retract of $x = y$. Thus, $\sum_{y:A} R(x, y)$ is a retract of $\sum_{y:A} x = y$. Since the latter type is contractible the former must be too. Thus any family of maps $\prod_{y:A}(x = y) \to R(x, y)$ is a family of equivalences (since its totalization is a map between contractible types and thus an equivalence).

But now we know that the identity types of A are propositions so A must be a set. □

Recall a type A has decidable equality if the identity type $x =_A y$ is decidable for every $x, y : A$, meaning

$$\prod_{x,y:A} (x = y) + \neg(x = y).$$

Theorem (Hedberg)**.** *Any type with decidable equality is a set.*

Proof. Let $d : \prod_{x,y:A}(x = y) + \neg(x = y)$ be a witness to the fact that A has decidable equality and let $\mathcal{U}$ be a universe containing A.

Define a type family $R'(x, y) : ((x = y) + \neg(x = y)) \to \mathcal{U}$ by

$$R'(x, y, \text{inl}(p)) :\equiv \mathbb{1} \quad R'(x, y, \text{inr}(p)) :\equiv \emptyset.$$

Note that this is a family of propositions. Now define $R(x, y) :\equiv R'(x, y, d(x, y))$. This defines a family of propositions $R : A \to A \to \mathcal{U}$. This is a reflexive binary relation so the apply the previous theorem to conclude that A is a set we must only show that R implies identity.

Since R is defined to be an instance of R' it suffices to construct a function for each $q : (x = y) + \neg(x = y)$ that proves $f(q) : R'(x, y, q) \to (x = y)$. We have this by

$$f(\text{inl}(p), r) :\equiv p \qquad f(\text{inr}(p), r) :\equiv \text{ex-falso}(r).$$ □

General truncation levels

So far we have defined:

$$\text{is-contr}(A) :\equiv \sum_{a:A} \prod_{x:A} a = x$$
$$\text{is-prop}(A) :\equiv \prod_{x,y:A} \text{is-contr}(x = y)$$
$$\text{is-set}(A) :\equiv \prod_{x,y:A} \text{is-prop}(x = y)$$

The is an obvious pattern here that we might continue, with the next definition being $\prod_{x,y:A} \text{is-set}(x = y)$. But what do we call this?

These predicates define the first few layers of the hierarchy of truncation levels. Sets, are often thought of as "0-dimensional," so they form level 0 of the hierarchy. The propositions then live at level -1 and contractible types are at level -2. The next level in the hierarchy defines the 1-types $\text{is-1-type}(A) :\equiv \prod_{x,y:A} \text{is-set}(x = y)$.

Let $\mathbb{T}$ be the inductive type with constructors $-2_\mathbb{T} : \mathbb{T}$ and $\text{succ}_\mathbb{T} : \mathbb{T} \to \mathbb{T}$. The natural inclusion $i : \mathbb{N} \to \mathbb{T}$ is defined recursively by $i(0_\mathbb{N}) :\equiv \text{succ}_\mathbb{T}(\text{succ}_\mathbb{T}(-2_\mathbb{T}))$ and $i(\text{succ}_\mathbb{N}(n)) :\equiv \text{succ}_\mathbb{T}(i(n))$. We abbreviate by writing $-2, -1, 0, 1, 2, \ldots$ for the first few terms of $\mathbb{T}$ when the context is clear.

defn. Define $\text{is-trunc} : \mathbb{T} \to \mathcal{U} \to \mathcal{U}$ recursively by

$$\text{is-trunc}_{-2}(A) :\equiv \text{is-contr}(A) \qquad \text{is-trunc}_{k+1}(A) :\equiv \prod_{x,y:A} \text{is-trunc}_k(x =_A y).$$

When $\text{is-trunc}_k(A)$ holds we say A is k-**truncated** or is a k-**type**. You can prove, inductively in $k : \mathbb{T}$, that this is logically independent of the universe being used to define $\text{is-trunc}_k(A)$.

For $k \geq 0$, we may also say that A is a **proper** k-**type** if $\text{is-trunk}_k(A)$ holds but $\text{is-trunk}_{k-1}(A)$ does not.

Given a universe $\mathcal{U}$, we may also define a universe of k-truncated types by

$$\mathcal{U}^k :\equiv \sum_{X:\mathcal{U}} \text{is-trunc}_k(X).$$

The truncation levels are successively contained in one another:

Proposition. *If A is a k-type then A is also a $k+1$-type.*

Proof. We use induction on $k : \mathbb{T}$. In the base case, we have shown already that contractible types are propositions. For the inductive step, note that if any k-type is a $k+1$-type then this applies to show that the identity types of a $k+1$-type, which are known to be k-types, are also $k+1$-types. This proves that any $k+1$-type is a $k+2$-type. □

In particular:

Corollary. *If A is a k-type its identity types are also k-types.*

General truncation levels are stable under equivalence:

Lemma. *If $e : A \simeq B$ then* $\text{is-trunc}_k A \leftrightarrow \text{is-trunc}_k B$.

Proof. As before we show that if B is a k-type then so is A. The converse follows by using $(A \simeq B) \to (B \simeq A)$. We prove this by induction on $k : \mathbb{T}$.

We know this for contractible types, which is the base case. For the inductive step, $e : A \simeq B$ provides an equivalence $\text{ap}_e : (x = y) \to (e(x) = e(y))$ for any $x, y : A$. If B is a $k+1$-type its identity types are k-types so the inductive hypothesis implies that $(x = y)$ is also a k-type. This proves that A is a $k+1$-type. □

A similar argument shows:

Corollary. *If $f : A \to B$ is an embedding and B is a $k+1$-type, then so is A.*

Our contractible types came with an analogous notion of contractible maps, aka equivalences, whose fibers are contractible types. We'll now extend these notions to the higher truncation levels, paying particular attention to level -1 where contractible types are replaced by propositions.

Subtypes

Now that we know about propositions we can say that a type family $P : A \to \mathcal{U}$ is a "predicate" if for each $a : A$, $P(a)$ is a proposition. In other words, predicates are type families $P : A \to \mathrm{Prop}_{\mathcal{U}}$. Other terminology is commonly in use in this situation.

defn. A type family B over A is a **subtype** of A if for each $x : A$, $B(x)$ is a proposition. In this situation we say that $B(x)$ is a **property** of $x : A$.

We'll show that for subtypes B over A the map $\mathrm{pr}_1 : \sum_{x:A} B(x) \to A$ is an embedding. It follows, then, that $(x, y) = (x', y')$ if and only if $x =_A x'$.

Theorem. *For $f : A \to B$ the following are logically equivalent:*

(i) *f is an embedding.*
(ii) *$\mathrm{fib}_f(b)$ is a proposition for all $b : B$*

Proof. By the fundamental theorem of identity types, f is an embedding if and only if $\sum_{x:A} f(x) =_B f(y)$ is contractible for each $y : A$. This is the fiber of f over the point $f(y) : B$. Thus condition (i) is logically equivalent to:

(i') $\mathrm{fib}_f(f(a))$ is contractible for each $a : A$.

At the same time, by our logically equivalent ways of characterizing the propositions, (ii) is logically equivalent to:

(ii') $\mathrm{fib}_f(b) \to \text{is-contr}(\mathrm{fib}_f(b))$ for all $b : B$.

We'll prove that (i') and (ii') are logically equivalent.

Assume (i'). let $b : B$, and suppose $\mathrm{fib}_f(b)$ holds, which by Σ-induction provides terms $x : A$ and $p : f(x) = b$. By (i'), $\mathrm{fib}_f f(x)$ is contractible and transport along p then defines an equivalence

$$\mathrm{fib}_f(f(x)) \simeq \mathrm{fib}_f(b).$$

Thus $\mathrm{fib}_f(b)$ must be contractible as well, proving (ii').

Now assume (ii') and let $a : A$. By our assumption (ii'), to show that $\mathrm{fib}_f(f(a))$ is contractible it suffices to show that it is inhabited, which we do with the pair $(a, \mathrm{refl}_{f(a)})$. □

Corollary. *For any family $B : A \to \mathcal{U}$ the following are logically equivalent:*

(i) *$\mathrm{pr}_1 : \sum_{x:A} B(x) \to A$ is an embedding.*
(ii) *$B(x)$ is a proposition for each $x : A$.*

Proof. Since $\mathrm{fib}_{\mathrm{pr}_1}(x) \simeq B(x)$ this follows immediately from the previous theorem. □

Truncated maps

defn. A map $f : A \to B$ is **k-truncated** if its fibers are k-truncated.

We have seen that the -2-truncated maps are the equivalences, while the -1-truncated maps are the embeddings. Our final theorem generalizes this to higher truncation levels.

Theorem. *For $f : A \to B$ the following are logically equivalent:*

(i) *f is $(k+1)$-truncated.*
(ii) *For each $x, y : A$, $\mathrm{ap}_f : (x = y) \to (f(x) = f(y))$ is k-truncated.*

Proof. Both directions use the characterization of identity types of fibers:

$$((x, p) =_{\mathrm{fib}_f(b)} (y, q)) \simeq \sum_{\alpha : x = y} p = \mathrm{ap}_f(\alpha) \cdot q.$$

The first statement is about identity types of fibers so consider $s, t : \mathrm{fib}_f(b)$. We claim there is an equivalence

$$(s = t) \simeq \mathrm{fib}_{\mathrm{ap}_f}(\mathrm{pr}_2(s) \cdot \mathrm{pr}_2(t)^{-1}).$$

By Σ-induction we can construct this for pairs $(x,p),(y,q) : \mathrm{fib}_f(b)$ for which we calculate

$$\begin{aligned}((x,p) = (y,q)) &\simeq \sum_{\alpha:x=y} p = \mathrm{ap}_f(\alpha) \cdot q \\ &\simeq \sum_{\alpha:x=y} \mathrm{ap}_f(\alpha) \cdot q = p \\ &\simeq \sum_{\alpha:x=y} \mathrm{ap}_f(\alpha) = p \cdot q^{-1} \\ &=: \mathrm{fib}_{\mathrm{ap}_f}(p \cdot q^{-1}).\end{aligned}$$

It follows that if ap_f is k-truncated then each identity type $(s = t)$ of $\mathrm{fib}_f(b)$ is equivalent to a k-truncated type and thus the fibers $\mathrm{fib}_f(b)$ are $k+1$-types, which means that the map f is $k+1$-truncated.

For the converse, we have an equivalence between $\mathrm{fib}_{\mathrm{ap}_f}(p)$ and the identity type $(x,p) =_{\mathrm{fib}_f(f(y))} (y, \mathrm{refl}_{f(y)})$. So if f is $(k+1)$-truncated these fibers are $k+1$-truncated and thus their identity types are k-types. This proves that the fiber $\mathrm{fib}_{\mathrm{ap}_f}(p)$ is k-truncated so the map ap_f is k-truncated. □

August 5: Function extensionality & Universal properties

Let A, B, and C be types. It is natural to ask whether the types

$$(A \times B \to C) \qquad \text{and} \qquad (A \to B \to C)$$

are equivalent, perhaps via "currying" and "uncurrying"?

Without too much difficulty, we can construct a logical equivalence between these types. One of these maps is

$$\text{ev-pair} : (A \times B \to C) \to (A \to B \to C)$$

defined by $(\text{ev-pair} g)(a,b) :\equiv g(a,b)$. Note this definition uses the introduction rule

$$(\_,\_) : A \to B \to A \times B$$

for product types. In the other direction we have

$$\mathrm{ind}_\times : (A \to B \to C) \to (A \times B \to C)$$

defined by the elimination rule for product types. Recall that this can be expressed in terms of an induction principle for any type family $x : A, y : B \vdash P(x,y)$, in the form of a map

$$\mathrm{ind}_\times : \left(\Pi_{x:A}\Pi_{y:B} P(x,y)\right) \to \left(\Pi_{p:A\times B} P(p)\right).$$

This specializes to the map we need in the case of a constant type family.

It remains to show that ev-pair and $\mathrm{ind}_\times$ are inverse equivalences. To show that $\text{ev-pair} \circ \mathrm{ind}_\times \simeq \mathrm{id}$ we must construct an identification

$$\text{ev-pair}(\mathrm{ind}_\times f) =_{A\to B\to C} f$$

for each $f : A \to B \to C$. For any $a : A$ and $b : B$ we know that

$$(\text{ev-pair}(\mathrm{ind}_\times f))(a,b) \doteq (\mathrm{ind}_\times f)(a,b) \doteq f(a,b),$$

by the definition of ev-pair and by the computation rule for product types. Thus by the η-rule for function types,

$$\text{ev-pair}(\mathrm{ind}_\times f) \doteq \lambda a \lambda b. f(a,b) \doteq f$$

Thus refl defines an identification

$$\mathrm{refl} : \text{ev-pair}(\mathrm{ind}_\times f) =_{A\to B\to C} f.$$

For the other homotopy $\mathrm{ind}_\times \circ \text{ev-pair} \simeq \mathrm{id}$ we must construct an identification

$$\mathrm{ind}_\times(\text{ev-pair} g) =_{A\times B\to C} g$$

for each $g : A \times B \to C$. Here we can't compute the value of of the left hand side at a generic term $p : A \times B$ but in the case of a pair $(a,b) : A \times B$ we do have definitional equalities

$$(\mathrm{ind}_\times(\text{ev-pair} g))(a,b) \doteq g(a,b).$$

Thus refl defines a term

$$\lambda a.\lambda b.\text{refl} : \Pi_{a:A}\Pi_{b:B}(\text{ind}_\times(\text{ev-pair} g))(a,b) = g(a,b).$$

By the induction principle for product types, we have a map

$$\text{ind}_\times : \left(\Pi_{a:A}\Pi_{b:B}(\text{ind}_\times(\text{ev-pair} g))(a,b) = g(a,b)\right) \to \left(\Pi_{p:A\times B}(\text{ind}_\times(\text{ev-pair} g))p = g(p)\right).$$

Thus, we have a homotopy

$$\text{ind}_\times(\lambda a.\lambda b.\text{refl}) : \text{ind}_\times(\text{ev-pair} g) \sim g.$$

However, we require an identification $\text{ind}_\times(\text{ev-pair} g) = g$ and we do not have a mechanism to promote homotopies to identifications between functions. So our proof gets stuck here.

We'll be able to complete this argument in the presence of the **function extensionality** axiom, which asserts that the canonical map that converts an identification between functions into a homotopy is a family of equivalences of types.

Function extensionality

The function extensionality principle characterizes the identity type of an arbitrary dependent function type, asserting that the type $f = g$ of identifications between dependent functions $f, g : \prod_{x:A} B(x)$ is equivalent to the type of homotopies $f \sim g$. It has several equivalent forms:

Proposition. *For a type family $B : A \to \mathcal{U}$ the following are logically equivalent:*

*(i) The **function extensionality principle holds** for $f, g : \prod_{x:A} B(x)$: the family of maps*

$$\text{htpy-id} : (f = g) \to (f \sim g)$$

defined by sending refl *to* refl-htpy *is a family of equivalences.*

(ii) For any $f : \prod_{x:A} B(x)$, the total space

$$\sum_{g:\prod_{x:A} B(x)} f \sim g$$

is contractible with $(f, \text{refl-htpy}_f)$ as its center of contraction.

*(iii) The principle of **homotopy induction** holds: for any family of types P depending on $f, g : \prod_{x:A} B(x)$ and $H : f \sim g$ the evaluation function*

$$\text{ev}\colon \left(\prod_{f,g:\prod_{x:A} B(x)} \prod_{H:f\sim g} P(f,g,H)\right) \to \prod_{f:\prod_{x:A} B(x)} P(f,f,\text{refl-htpy}_f)$$

has a section.

Proof. This follows by applying the fundamental theorem of identity types to the type $\prod_{x:A} B(x)$, term $f : \prod_{x:A} B(x)$, and type family $g : \prod_{x:A} B(x) \vdash f \sim g$ type . □

A fourth equivalent condition is more surprising because it appears to express only a weak function extensionality principle.

Theorem. *For any universe $\mathcal{U}$ the following are logically equivalent:*

*(i) The **function extensionality principle** holds in $\mathcal{U}$: for any type family B over A and dependent functions the map*

$$\text{htpy-id} : (f = g) \to (f \sim g)$$

is an equivalence.

*(ii) The **weak function extensionality principle** holds in $\mathcal{U}$: for any type family B over A one has*

$$\left(\prod_{x:A} \text{is-contr}(B(x))\right) \to \text{is-contr}\left(\prod_{x:A} B(x)\right).$$

Proof. Assume (i) and suppose each fiber $B(x)$ is contractible with center of contraction $c(x)$ and contracting homotopy $C_x : \prod_{y:B(x)} c(x) = y$. Define $c = \lambda x.c(x)$ to be the center of contraction of $\prod_{x:A} B(x)$. For the contraction we require a term of type

$$\prod_{f:\prod_{x:A} B(x)} c = f.$$

By function extensionality, $(c \sim f) \to (c = f)$ so it suffices to construct a term of type $c \sim f :\equiv \prod_{x:A} c(x) = f(x)$ and $\lambda x.C_x(f(x))$ is just such a term.

For the converse, assume (ii). By the previous result it suffices to show that the type

$$\sum_{g:\prod_{x:A} B(x)} f \sim g$$

is contractible. Note we have a section-retraction pair:

$$\left(\sum_{g:\prod_{x:A} B(x)} f \sim g\right) \xrightarrow{s} \left(\prod_{x:A} \sum_{y:B(x)} f(x) = y\right) \xrightarrow{r} \left(\sum_{g:\prod_{x:A} B(x)} f \sim g\right)$$

defined by

$$s :\equiv \lambda(g, H).\lambda x.(g(x), H(x)) \quad \text{and} \quad r :\equiv \lambda p.(\lambda x.\mathrm{pr}_1(p(x)), \lambda x.\mathrm{pr}_2(p(x))).$$

The composite is homotopic to the identity function by the computation rules for Σ and Π-types. Here the central type is a product of contractible types so must be contractible by (ii). Since retracts of contractible types are contractible, the claim follows. □

Henceforth, we will assume the function extensionality principle as an axiom:

Axiom (function extensionality)**.** For any type family B over A and any pair of dependent functions $f, g : \prod_{x:A} B(x)$ the map

$$\text{htpy-id} : (f = g) \to (f \sim g)$$

is an equivalence, with inverse id-htpy.

That is, we add the following rule to type theory:

$$\frac{\Gamma, x : A \vdash B(x)\ \text{type} \qquad \Gamma \vdash f : \prod_{x:A} B(x) \qquad \Gamma \vdash g : \prod_{x:A} B(x)}{\Gamma \vdash \text{funext} : \text{is-equiv}(\text{htpy-id}_{f,g})}$$

There are myriad consequences of the function extensionality axiom. Firstly:

Theorem. *For any type family B over A one has*

$$\left(\prod_{x:A} \text{is-trunc}_k(B(x))\right) \to \text{is-trunc}_k\left(\prod_{x:A} B(x)\right).$$

Proof. The theorem states that k-types are closed under arbitrary dependent products. We prove this by induction on $k \geq -2$. The base case is the weak function extensionality principle.

For the inductive step assume k-types are closed under products and consider a family B of $(k+1)$-types. To show that $\prod_{x:A} B(x)$ is $(k+1)$-truncated we must show that $f = g$ is k-truncated for every $f, g : \prod_{x:A} B(x)$. By function extensionality this is equivalent to the type $f \sim g :\equiv \prod_{x:A} f(x) = g(x)$ which is defined to be a dependent product of k-truncated types and thus is k-truncated by hypothesis. Since k-truncated types are closed under equivalence, the result follows. □

For a non-dependent family we conclude that:

Corollary. *Suppose B is a k-type. Then for any type A, the type of functions $A \to B$ is a k-type.*

In particular, $\neg A$ is a proposition for any type A!

The type theoretic principle of choice

There's a result that's sometimes called the "type theoretic principle of choice" that is just true, asserting that Π-types distribute over Σ-types. Without function extensionality we can give a logical equivalence between the following types, but we can now show that it is a full equivalence.

Theorem. *For any family of types* $x : A, y : B(x) \vdash C(x,y)$ type *the map*

$$\text{choice} : \left(\prod_{x:A} \sum_{y:B(x)} C(x,y)\right) \to \left(\sum_{f:\prod_{x:A} B(x)} \prod_{x:A} C(x, f(x))\right)$$

defined by

$$\text{choice}(h) :\equiv (\lambda x.\text{pr}_1(h(x)), \lambda x.\text{pr}_2(h(x)))$$

is an equivalence.

Consequently, whenever we have types A and B and a type family C over B there is an equivalence

$$\left(A \to \sum_{y:B} C(y)\right) \simeq \left(\sum_{f:A\to B} \prod_{x:A} C(f(x))\right)$$

Proof. Define the inverse map choice^{-1} by

$$\text{choice}^{-1}(f,g) :\equiv \lambda x.(f(x).g(x)).$$

For the first homotopy it suffices to define an identification $\text{choice}(\text{choice}^{-1}(f,g)) = (f,g)$. The left-hand side computes to

$$\text{choice}(\text{choice}^{-1}(f,g)) \doteq \text{choice}(\lambda x.(f(x).g(x))) \doteq (\lambda x.f(x), \lambda x.g(x))$$

which is definitely equal to the right-hand side by the computation rules for function types.

For the second homotopy, we require an identification $\text{choice}^{-1}(\text{choice}(h)) = h$. The left-hand side computes to

$$\text{choice}^{-1}(\lambda x.\text{pr}_1(h(x)), \lambda x.\text{pr}_2(h(x))) \doteq \lambda x.(\text{pr}_1(h(x)), \text{pr}_2(h(x))).$$

We do not have a definitional equality relating $h(x)$ and $(\text{pr}_1(h(x)), \text{pr}_2(h(x)))$ but in our characterization of the identity type of Σ-types we do have an identification between them called eq-pair(refl, refl). By function extensionality, the homotopy $\lambda x.\text{eq-pair}(\text{refl}, \text{refl}) : \text{choice}^{-1}(\text{choice}(h)) \sim h$ can be turned into an identification and thus a homotopy $\text{choice}^{-1} \circ \text{choice} \sim \text{id}$. □

With function extensionality, the rules for the other inductive types can similarly be expressed as universal properties that characterize the type of dependent functions out of a particular type. One map of the equivalence is defined by evaluating at the constructors, while the other is given by the induction principle. The computation rule shows that the induction map is a section of the evaluation map, and the remaining required homotopy can also be constructed from this computation rule, combined with function extensionality.

Universal properties

More generally, the function extensionality axiom allows us to prove universal properties, which characterize maps out of or into a given type, and characterize that type up to equivalence. Some examples follow.

In our first example, we consider the maps out of Σ-types. The universal property states that the map

$$\text{ev-pair} : \left(\left(\sum_{x:A} B(x)\right) \to C\right) \to \left(\prod_{x:A} B(x) \to C\right)$$

given by $f \mapsto \lambda x.\lambda y.f(x,y)$ is an equivalence for any type C. More generally, the type C might depend on the type $\sum_{x:A} B(x)$, so we prove the result in that form.

Theorem (universal property of Σ-types)**.** *Let B be a type family over A and let C be a type family over $\sum_{x:A} B(x)$. Then the map*

$$\text{ev-pair} : \left(\prod_{z:\sum_{x:A} B(x)} C(z)\right) \to \left(\prod_{x:A} \prod_{y:B(x)} C(x,y)\right)$$

given by $f \mapsto \lambda x.\lambda y.f(x,y)$ is an equivalence

Proof. The inverse map is given by the induction principle for Σ-types:

$$\text{ind}_\Sigma : \left(\prod_{x:A} \prod_{y:B(x)} C(x,y)\right) \to \left(\prod_{z:\sum_{x:A} B(x)} C(z)\right).$$

By the computation rules for Σ types and Π types, we have the homotopy

$$\text{refl-htpy} : \text{ev-pair} \circ \text{ind}_\Sigma \sim \text{id},$$

which shows that ind_Σ is a section of ev-pair.

Function extensionality is used to construct the other homotopy. To define a homotopy $\text{ind}_\Sigma \circ \text{ev-pair} \sim \text{id}$ requires identifications $\text{ind}_\Sigma(\lambda x.\lambda y.f(x,y)) = f$. By function extensionality it suffices to show that

$$\prod_{z:\sum_{x:A} B(x)} \text{ind}_\Sigma(\lambda x.\lambda y.f(x,y))(z) = f(z).$$

By Σ-induction it suffices to prove this for pairs in which case we require identifications

$$\text{ind}_\Sigma(\lambda x.\lambda y.f(x,y))(a,b) = f(a,b),$$

but this holds definitionally by the computation rule for Σ types. □

In the non-dependent case we have as a corollary:

Corollary. *For types A and B and C,*

$$\text{ev-pair} : (A \times B \to C) \to (A \to B \to C)$$

given by $f \mapsto \lambda a.\lambda b.f(a,b)$ is an equivalence. □

The universal property of identity types

The universal property for identity types can be understood as an (undirected) type theoretic version of the Yoneda lemma. In the most familiar case, when B is a type family over A, it says that the map

$$\text{ev-refl} : \left(\prod_{x:A} (a = x) \to B(x)\right) \to B(a)$$

given by $f \mapsto f(a, \text{refl}_a)$ is an equivalence. As before, though, it generalizes to a dependent version of the undirected Yoneda lemma, where the type family B is allowed to depend on $x : A$ and $p : a = x$.

Theorem. *Consider a type A, a term $a : A$, and a type family $B(x,p)$ over $x : A$ and $p : a = x$. Then the map*

$$\text{ev-refl} : \left(\prod_{x:A} \prod_{p:a=x} B(x,p)\right) \to B(a, \text{refl}_a)$$

defined by $f \mapsto f(a, \text{refl}_a)$ is an equivalence.

Proof. The inverse map is

$$\text{path-ind}_a : B(a, \text{refl}_a) \to \left(\prod_{x:A} \prod_{p:a=x} B(x,p)\right),$$

which is a section by the computation rule of the path induction principle.

For the other homotopy $\text{path-ind}_a \circ \text{ev-refl} \simeq \text{id}$ let $f : \prod_{x:A} \prod_{p:a=x} B(x,p)$. To prove that $\text{path-ind}_a(f(a, \text{refl}_a)) = f$ we apply function extensionality twice so that it suffices to show that

$$\prod_{x:A} \prod_{p:a=x} \text{path-ind}_a(f(a, \text{refl}_a), x, p) = f(x,p).$$

This follows from path induction on p since path-ind$_a(f(a, \mathrm{refl}_a), a, \mathrm{refl}_a) \doteq f(a, \mathrm{refl}_a)$ by the computation rule for path induction. □

Univalence

We have now characterized the identity types of every family of types that we have introduced with a single exception. It remains to identify the identity types of the universe $\mathcal{U}$. As with the case of identity types of dependent function types, we cannot prove a characterization in type theory. Instead, we need an axiom. What differentiates *homotopy* type theory from Martin-Löf's dependent type theory is our adoption of Voevodsky's *univalence axiom* for this purpose.

defn. A universe $\mathcal{U}$ is **univalent** if for any $A, B : \mathcal{U}$, the map

$$\text{equiv-id} : (A = B) \to (A \simeq B)$$

defined by path induction by sending refl_A to the identity equivalence is an equivalence.

In other words, univalence asserts that the family $\lambda A, \lambda B, A \simeq B : \mathcal{U} \to \mathcal{U} \to \mathcal{U}$ with the identity equivalences gives an identity system for $\mathcal{U}$.

Exercise. Apply the fundamental theorem of identity types, to obtain various equivalent forms of the univalence axiom.

As Voevodsky first observed, univalence implies function extensionality, through a non-obvious proof. So the applications we've seen of function extensionality, to prove various universal properties, can also be understood as consequence of the univalence axiom. But there are myriad other consequences of univalence as we shall soon discover.

References

[R] Egbert Rijke, *Introduction to Homotopy Type Theory*, 2022.

Dept. of Mathematics, Johns Hopkins University, 3400 N Charles St, Baltimore, MD 21218
Email address: eriehl@jhu.edu