Equivalences

Sometimes we want types to be propositions (no structure). Sometimes we're interested in structure.

Given $f : A \to B$, want a proposition $\mathrm{isEquiv}(f)$.

The type $\sum_{g : B \to A} fg = 1 \times gf = 1$ is not a proposition.
→ would ask for adjoint equivalence, or equivalently:

Def. A function $f : A \to B$ is an equivalence if:

$$\mathrm{isEquiv}(f) := \prod_{b:B} \mathrm{isContr}\left(\sum_{a:A} fa = b \right).$$

1 = fiber

Write

$$A \simeq B := \sum_{f : A \to B} \mathrm{isEquiv}(f).$$

Ex. Every contractible type is equivalent to $\mathbb{1}$.

Fact. For every type A, $A \simeq A$, so we can define

$$\mathrm{idtoequiv} : A = B \to A \simeq B.$$

Def. The univalence axiom asserts

$$\mathrm{ua} : \mathrm{isEquiv}(\mathrm{idtoequiv}).$$

Univalence for logic and sets

Def. Given a type T and a predicate $P : T \to \mathrm{Prop}$, the <u>subtype</u> of T given by P is

$$\sum_{t:T} P(t).$$

The projection

$$\pi_1 : \sum_{t:T} P(t) \longrightarrow T$$

gives us the 'inclusion'.

Ex. isContr, isProp, isOfHLevel n are all predicates $\mathcal{U} \to \mathrm{Prop}$.

Def. $\mathrm{Prop} := \sum_{P:\mathrm{Type}} \mathrm{isProp}(P)$

Ex. The univalence axiom implies

$$(P =_{\mathrm{Prop}} Q) \simeq (P \leftrightarrow Q).$$

Lem. $(P \leftrightarrow Q)$ is a proposition.

Cor. Prop is a set.

Def. $\mathrm{Set} := \sum_{S:\mathrm{Type}} \mathrm{isSet}(S)$

Fact. The univalence axiom implies

$$(P =_{\mathrm{Set}} Q) \simeq (P \cong Q).$$

Lem. $P \cong Q$ is a set

[illegible] is a set.

Cor. Set is a groupoid.

Def.
$$\mathrm{Grp} := \sum_{G:\mathrm{Set}} \ \sum_{e:G} \ \sum_{m:G\to G\to G} \ \sum_{i:G\to G} \prod_{x:G} (m(e,x) = x) \times (m(x,e) = x) \times \prod_{x,y,z:G} ((xy)z = x(yz)) \times \prod_{x:G} (m(ix,x) = e) \times (m(x,ix) = e).$$

Q. Why do we ask G to be a set?

Fact. The univalence axiom implies

$$(G =_{\mathrm{Grp}} H) \simeq (G \cong H)$$

Cor. Grp is a groupoid.

Fact. We have the same univalence principle for any algebraic structure on a set.

Moral: univalence allows us to do mathematics <u>up to the appropriate notion of</u> sameness in a type (in these examples).

→ 'Structure identity principle' (Aczel, Coquand)

→ 'identity of indiscernables' (Leibniz)

Category theory

Def. $\mathrm{Grpd} :\equiv \sum_{T:U}$ is of hlevel 3 T.

We have a univalence principle

Thm. $(G =_{\mathrm{Grpd}} H) \simeq (G \cong H)$.

But what about categories?

How do we define categories in UF?

- A category is normally defined as a bunch of sets.
- We could do this, but
  - we stay in the set level (everything in mathematics is a bunch of sets)
  - the SIP for structures on sets tells us that

    $$(\mathcal{C} =_{\mathrm{Cat}} \mathcal{D}) \simeq (\mathcal{C} \cong \mathcal{D})$$

    and isomorphism is not the right kind of sameness for categories.

Instead, we take a groupoid and put extra structure on it.
- Every category has a 'core groupoid'
  - the objects and all invertible morphisms.

Def. A <u>category</u> $\mathcal{C}$ consists of

- a ~~set~~ groupoid $\mathrm{ob}\,\mathcal{C}$
- a set $\hom(X,Y)$ for every pair $X,Y \in \mathrm{ob}\,\mathcal{C}$

  $X,Y : \mathrm{ob}\,\mathcal{C} \vdash \hom(X,Y) : \mathrm{Set}$
- an ~~element~~ term $1_X \in \hom(X,X)$ for every $X \in \mathrm{ob}\,\mathcal{C}$

  $X : \mathrm{ob}\,\mathcal{C} \vdash 1_X : \hom(X,X)$
- a function $\circ : \hom(X,Y) \times \hom(Y,Z) \to \hom(X,Z)$ for every $X,Y,Z \in \mathrm{ob}\,\mathcal{C}$.

  $X,Y,Z : \mathrm{ob}\,\mathcal{C} \vdash \circ : \hom(X,Y) \to \hom(Y,Z) \to \hom(X,Z)$
- such that
- the morphism $\mathrm{idtoiso} : (X = Y) \to \mathrm{iso}(X,Y)$ is an equivalence

  $\left(\mathrm{iso}(X,Y) := \sum_{f:\hom(X,Y)} \sum_{g:\hom(X,Y)} g\circ f = 1_X \times f \circ g = 1_Y\right)$.

  . . .

i.e., the type $\mathcal{C}at := \sum_{\mathrm{ob}\,\mathcal{C}:\mathrm{Grpd}} \sum_{\hom : \mathrm{ob}\,\mathcal{C} \to \mathrm{ob}\,\mathcal{C} \to \mathrm{Set}} \sum_{1 : \prod_{X:\mathrm{ob}\,\mathcal{C}} \hom X} \cdots$

<u>NB</u>. This is often called a <u>univalent category</u> and the requirement that idtoiso is an equivalence is called (internal) univalence.

<u>Thm (univalence for categories)</u> $(\mathcal{C} =_{\mathcal{C}at} \mathcal{D}) \simeq (\mathcal{C} \simeq \mathcal{D})$.

Cor $\mathcal{C}at$ is a 2-groupoid (h-level 4).

This is a great acheivement for UF.
→ evil vs non evil, practice of CT
→ now part of the theory

The type $\mathcal{C}at$ consists of
- terms → categories
- equalities → equivalence of categories

What about functors, natural transformations?

Def. A functor $F : \mathcal{C} \to \mathcal{D}$ consists of
- a function $\mathrm{ob}F : \mathrm{ob}\,\mathcal{C} \to \mathrm{ob}\,\mathcal{D}$
- functions $\hom_{\mathcal{C}}(X,Y) \to \hom_{\mathcal{D}}(FX, FY)$ for all $X, Y \in \mathcal{C}$
- such that ...

The type of functors $[\mathcal{C}, \mathcal{D}]$ is not a set
- terms → functors
- equalities → natural transformations

Def. A natural transformation $\eta : F \Rightarrow G : \mathcal{C} \to \mathcal{D}$ consists of

- $\eta_X : \hom(FX, GX)$ for all $X : \mathrm{ob}\,\mathcal{C}$
- $$\begin{tikzcd} FX \arrow[r, "Ff"] \arrow[d, "\eta_X"'] \arrow[dr, phantom, "\varepsilon_f ="] & FY \arrow[d, "\eta_Y"] \\ GX \arrow[r, "Gf"'] & GY \end{tikzcd}$$ for all $X, Y : \mathrm{ob}\,\mathcal{C}$, $f : \hom(X, Y)$

The type of natural transformations $F \Rightarrow G$ is a <u>set</u>.

We can form a <u>bicategory</u> of categories.

→ Have univalence for bicategories.

→ Have univalence for any higher (but finite) algebraic str.