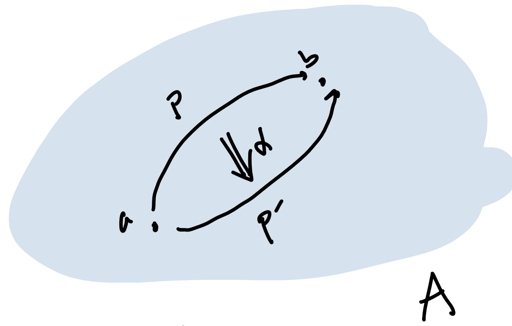
The groupoidal behaviour of types

(The first homotopical phenomena)



We can now think of types as collections of points (terms)
connected by homotopies/paths (equalities).

We can:

- have multiple equalities of the same type (ex: $p, p' : a =_A b$)

(Ex.) take the inverse of an equality (if $q : b =_A c$, then $q^{-1} : c =_A b$)

(Ex.) take composition of equalities (if $p : a =_A b$ and $q : b =_A c$, then $p \cdot q : a =_A c$)

(Ex.) have equalities of equalities ($\alpha : p =_{a =_A b} p'$)

Moreover: (Ex.) functions $A \to B$ respect equality (i.e. map $a =_A a'$ to $fa =_B fa'$)

This is how homotopies in spaces behave.

The space interpretation

Thm. (Voevodsky) There is an interpretation of dependent type theory
into Spaces (the category of Kan complexes) in which

types $\rightsquigarrow$ spaces
terms $\rightsquigarrow$ points
equalities $\rightsquigarrow$ paths

Transport

Prop. (Ex) For any dependent type $x:B \vdash E(x)$ type, any terms $b, b' : B$, and any equality $p : b =_B b'$, there is a function $tr_p : E(b) \to E(b')$.

- This ensures that everything respects propositional equality. If we think of E as a predicate on B, then if $E(b)$ is true and $b =_B b'$, so is $E(b')$.

- This is part of a more sophisticated relationship between type theo and homotopy theory (Quillen model category theory). Transport sa that $\pi : \sum_{b:B} E(b) \xrightarrow{\sigma} B$ behaves like a fibration in a QMC.

Equivalence. For types S, T, there is a notion of equivalence

$$S \simeq T$$

Similar to

$$\sum_{f:S\to T} \sum_{g:T\to S} \left(\prod_{x:S} gfx = x\right) \times \left(\prod_{y:T} fgy = y\right).$$

(To be revisited later.)

Characterizing equality in standard types

bool: We can show false = false, true = true, false ≠ true.

$\mathbb{N}$: We have similar: $sn = sm \Rightarrow n = m$, $0 \neq sn$

Σ-types: For $s, t : \sum_{a:A} B(a)$, have $(s = t) \simeq \sum_{p : \pi_1 s = \pi_1 t} tr_p\, \pi_2 s = \pi_2 t$.

Π-types: For $f, g : \prod_{a:A} B(a)$, maybe want $(f = g) \simeq \prod_{x:A} fx = gx$.

Not provable. Called <u>functional extensionality</u>. (funext)

Validated by interpretations in logic, sets, spaces.

$=$-types: For $p, q : a = b$, maybe want

$$(p = q) \simeq \mathbb{1}.$$

Not provable. Called <u>uniqueness of identity proofs</u>. (UIP)

Validated by interpretations in logic, sets.

U-types: For $S, T : U$, maybe want

$$(S = T) \simeq (S = T).$$

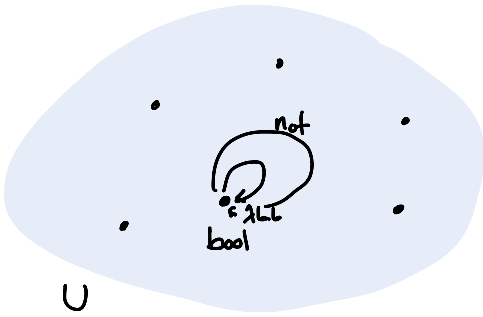
Not provable. Called <u>univalence</u>. (UA)

Validated by interpretation in spaces.

- UA $\Rightarrow$ funext.
- UIP $\wedge$ funext $\not\Rightarrow \bot$
- UA + UIP $\Rightarrow \bot$.

We choose UA.

Homotopy levels



We want to say things like "U is not a set".

- A set is something whose $=$-types don't have structure.

Def. A type T's h-level is 0 if

$$\mathrm{hlevel}\ 0\ T := \sum_{t:T} \prod_{s:T} s =_T t.$$

A type T's h-level is $\mathrm{s}n$ if

$$\mathrm{hlevel}\ \mathrm{s}n\ T := \prod_{s,t:T} \mathrm{hlevel}\ n\ s=t.$$

We've defined a function $\mathrm{hlevel} : \mathbb{N} \to \mathrm{Type} \to \mathrm{Type}$.

h-level 0. AKA contractible, isContr

Ex. $\mathbb{1}$ is contractible.

Most boring.

h-level 1. AKA propositions, isProp

Fact. Equivalent to $\prod_{x,y:P} x=y$.

__Ex__. $\emptyset, \mathbb{1}$ are propositions

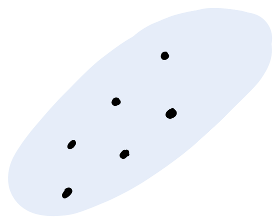
__Ex__. In fact, any contractible type is a proposition.

__Ex__. If a proposition is inhabited, it is contractible.

$\rightarrow$ So roughly, a proposition is $=$ to $\emptyset$ or $\mathbb{1}$.

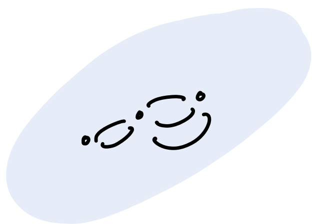
So these behave like logical propositions where Σ behaves like $\exists$, etc.

__h-level 2__. AKA sets, isSet



__Fact__. bool, $\mathbb{N}$ are propositions

__h-level 3__. AKA groupoids



__Fact__. Type has h-level at least 3.

__Ex__. If a type T has h-level n, then it has h-level $n+1$.

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:

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

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, is of hlevel n are all predicates $U \longrightarrow \mathrm{Prop}$.

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.

Cor. Set is a groupoid.

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

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)