Univalence for free, not yet

Joint work with Matthieu Sozeau

Nicolas Tabareau
Ascola, Nantes

Weak ω-groupoids

Voevodsky: Provides a new insight on Type Theory

where we can have univalence

Vladimir Voevodsky. Univalent Foundations of Mathematics

Weak ω-groupoids

What is univalence ?

Coarsely, the fact that to isomorphic types are equal.

Weak ω-groupoids

With this model, we can get more extensional principles

- Proof irrelevance : $\forall\ (\pi, \pi' : P),\ \pi = \pi'$
- Propositional extensionality : $P \leftrightarrow Q \rightarrow P = Q$
- Functional extensionality : $\forall\ x,\ f\ x = g\ x \rightarrow f = g$
- Reasoning modulo

Problem with weak ω-groupoids

This interpretation of Type Theory is difficult to understand / analyze / exploit:

- the definition of ω-groupoids (Batanin, Leinster) is (very) difficult to grasp in details
- dependent sums and products are interpreted using sections & projections

Leinster, Higher operads, higher categories

Coq with weak ω-groupoids

Using ω-dimensional (or even 2-dimensional) Type Theory with decidable type checker requires more insight on the interpretation.

Coq with weak ω-groupoids

We advocate for an internalization

of weak ω-groupoids interpretation in Coq

Coq with weak ω-groupoids

Full description of weak ω-groupoids is a difficult task.

Altenkirch and Rypacek: formalization assuming that all diagrams commute, instead of using minimal set of coherences

Problem: no inductive principles to reason on coherences.

Altenkirch and Rypacek : A Syntactical Approach to Weak ω-Groupoids

Why [weak] [ω]-groupoids ?

- Weak:
  - don't want to rely on (Leibniz) equality in the definition

- ω :
  - useful for a complete notion of univalence
  - useful to avoid truncation in the model
  - but not absolutely necessary as a first step

Coq with weak 2-groupoids

We will give an internalization

of weak 2-groupoids interpretation in Coq

Weak 2-groupoids vs groupoids

In Hofmann & Streicher: $[\![T]\!] : GPD$, GPD are 1-groupoids

Morphisms representing identities are identified up to propositional equality.

Weak 2-groupoids:
morphisms representing identities are identified up to another notion of "equivalence".

$$\text{E.g: } [\![\mathtt{Prop}]\!] := (\mathtt{Prop}, \mathrm{iff}, \mathrm{irrel}, \ldots).$$

Weak 2-groupoids vs groupoids

GPD are weak 2-groupoids where the equality of morphisms is propositional equality eq (which has J but not UIP).

Not sticking to identity sets (which are at the origin of the groupoid/homotopy models), we can *realize* a richer model.

Principle	Definition of equality
Proof-irrelevance	Irrelevant equality
Propositional extensionality	Logical equivalence
Functional extensionality	Pointwise equality
Univalence	Isomorphism

Weak 2-groupoids interpretation

The rest of the talk will describe our internalization of

the weak 2-groupoids interpretation in Coq

Weak 2-groupoids interpretation

Our interpretation relies on:

- Type classes
- Polymorphic universes
- Better management of projections in Coq

Goal: Extensional properties as lemmas

$\texttt{Lemma}$ prop_extensional $(P\ Q : [\_\mathrm{Prop}]) : [P] \leftrightarrow [Q] \rightarrow P \sim_1 Q.$

$\texttt{Lemma}$ proof_irrelevant $(P : [\_\mathrm{Prop}])\ (p\ q : [P]) : p \sim_1 q.$

$\texttt{Lemma}$ functional_extensionality $A\ B\ (f\ g : [A \longrightarrow B]) :$
nat_trans $f\ g \rightarrow f \sim_1 g.$

$\texttt{Lemma}$ sum_extensional $T\ F\ (m\ n : [\_\mathrm{Sum}\ (T{:=}T)\ F]) :$
$\forall\ (P : [m] \sim_1 [n])$, eq_rect' $\_\ [F]\ \_\ P \star (\pi_2\ m) \sim_1 \pi_2\ n \rightarrow m \sim_1 n.$

$\texttt{Lemma}$ univalence_statement $(U\ V : [\_\mathrm{Type}]) : (\mathrm{Equiv}\ U\ V) \rightarrow U \sim_1 V.$

Weak 2-groupoids in Coq

Categories

Start with computational notion of family of morphisms

$$\texttt{Definition}\ \mathrm{HomT}\ (A : \texttt{Type}) := A \to A \to \texttt{Type}.$$

Categories

Define type classes for identity, inverse and composition

$\texttt{Class}$ Identity $\{A\}$ $(Hom : \mathrm{HomT}\ A) :=$
  $\mathrm{identity} : \forall\ x,\ Hom\ x\ x.$

$\texttt{Class}$ Inverse $\{A\}$ $(Hom : \mathrm{HomT}\ A) :=$
  $\mathrm{inverse} : \forall\ x\ y{:}A,\ Hom\ x\ y \to Hom\ y\ x.$

$\texttt{Class}$ Composition $\{A\}$ $(Hom : \mathrm{HomT}\ A) :=$
  $\mathrm{composition} : \forall\ \{x\ y\ z{:}A\},\ Hom\ x\ y \to Hom\ y\ z \to Hom\ x\ z.$

Categories

The definition of Category is defined with 2 HomTs.

```
Class Category T (Hom : HomT T) (Hom2: _HomT Hom) := {

   Category_Identity :> Identity Hom;
   Category_Composition :> Composition Hom;

   id_R : ∀ x y (f : Hom x y), f ∘ (identity x) ∼ f ;
   id_L : ∀ x y (f : Hom x y), (identity y) ∘ f ∼ f ;
   assoc : ∀ x y z w (f: Hom x y) (g: Hom y z) (h: Hom z w),
           (h ∘ g) ∘ f ∼ h ∘ (g ∘ f);
   comp : ∀ x y z (f f': Hom x y) (g g': Hom y z),
           f ∼ f' → g ∼ g' → g ∘ f ∼ g' ∘ f'
 }.
```

Groupoids

Groupoid is a Category with inverses

```
Class Groupoid T (Hom : HomT T) (Hom₂: _HomT Hom)
                 (Groupoid_Category : Category Hom₂) := {
  Groupoid_Inverse :> Inverse Hom;

  inv_R : ∀ x y (f: Hom x y), f ∘ (inverse _ _ f) ∼ identity _ ;
  inv_L : ∀ x y (f: Hom x y), (inverse _ _ f) ∘ f ∼ identity _ ;
  inv : ∀ x y (f f':Hom x y), f ∼ f' → inverse _ _ f ∼ inverse _ _ f'}.
```

Weak 2-Categories

Using our "open" definition of a category
a weak 2-category is simply as a category at all levels...

```
Class Weak2Category T := {
  Hom1 :> HomT1 T;
  Hom2 :> _HomT eq1;
  Hom3 :> ∀ x y : T, _HomT (eq (x:=x) (y:=y));

  Category_1 :> Category Hom2;
  Category_2 :> ∀ x y, Category (Hom3 x y);
  Equivalence_3 :> ∀ x y (e e' : x ∼1 y), Equivalence (eq (x:=e) (y:=e'));
```

Weak 2-Categories

... plus compatibilities of course

$\text{ExLawId} : \forall\ x\ y\ z\ (f : x \sim_1 y)\ (g : y \sim_1 z),$
$\quad \text{identity}\ f\ {**}\ \text{identity}\ g \sim_3 \text{identity}\ (g \circ f);$

$\text{ExLawComp} : \forall\ x\ y\ z\ (f\ f'\ f'' : x \sim_1 y)\ (g\ g'\ g'': y \sim_1 z)$
$\quad (\alpha : f \sim_2 f')\ (\alpha':f'{\sim_2}f'')\ (\beta:g \sim_2 g')\ (\beta':g'{\sim_2}\ g''),$
$\quad (\alpha' \circ \alpha)\ {**}\ (\beta' \circ \beta) \sim_3 (\alpha'\ {**}\ \beta') \circ (\alpha\ {**}\ \beta);$

$\text{AssociativityCoherence} : \forall\ x\ y\ z\ w\ v$
$\quad (f\!: x \sim_1 y)\ (g\!: y \sim_1 z)\ (h : z \sim_1 w)\ (i\!: w \sim_1 v),$
$\quad \text{assoc'}\ (g \circ f)\ h\ i \circ \text{assoc'}\ f\ g\ (i \circ h) \sim_3$
$\quad (\text{assoc'}\ f\ g\ h\ {**}\ \text{identity}\ i) \circ \text{assoc'}\ f\ (h \circ g)\ i \circ (\text{identity}\ f\ {**}\ \text{assoc'}\ g\ h\ i);$

$\text{IdentityCoherence} : \forall\ x\ y\ z\ (f\!: x \sim_1 y)\ (g\!: y \sim_1 z),$
$\quad (\text{id\_L'}\ f\ {**}\ \text{identity}\ g) \circ \text{assoc'}\ f\ (\text{identity}\ y)\ g \sim_3 \text{identity}\ f\ {**}\ \text{id\_R'}\ g$

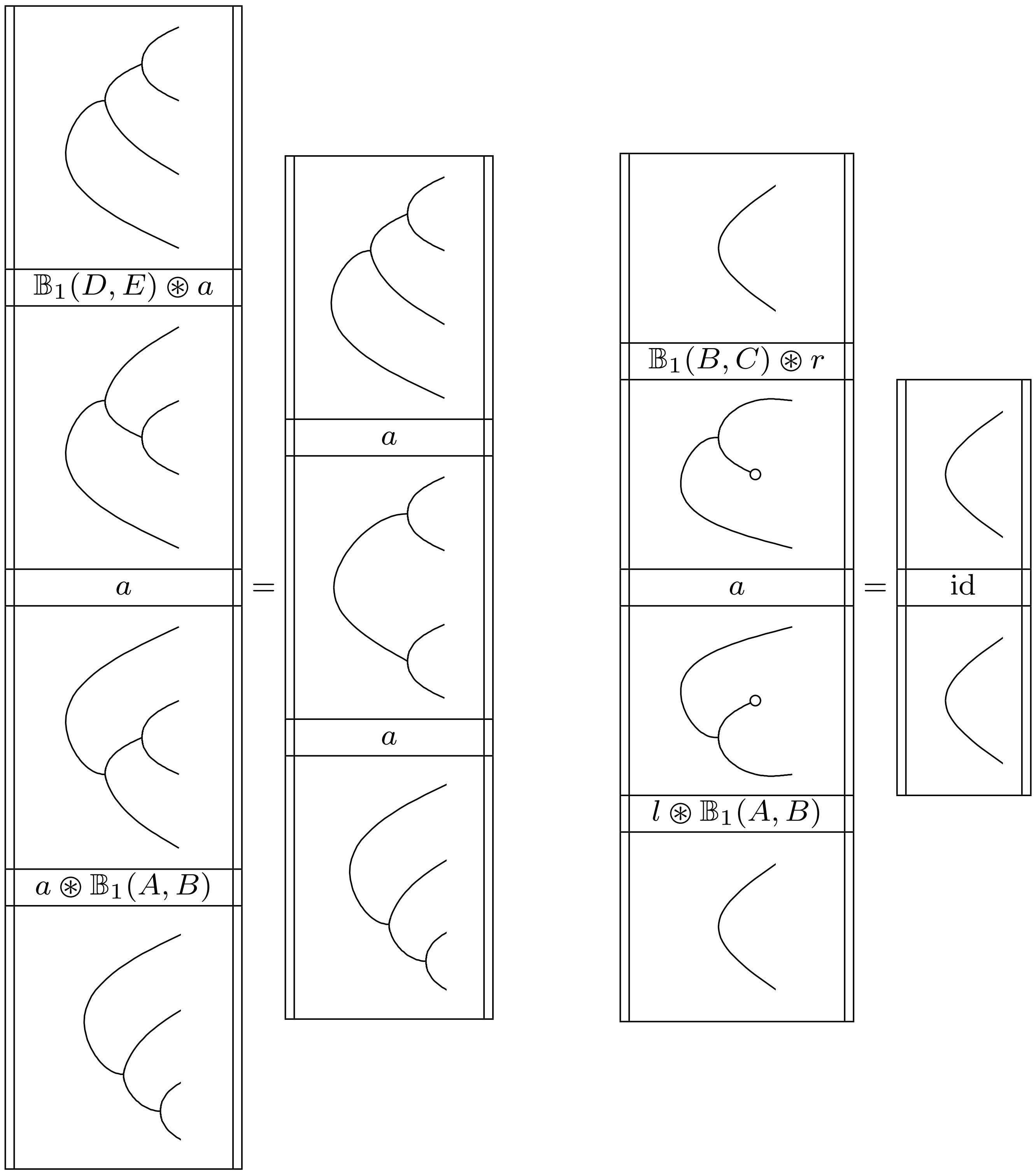
Weak 2-Categories

Horizontal composition of 2 cells is given by the witness of compatibility of composition

Definition HorComp $\{T\}$ $\{\mathrm{Hom}_1$: HomT1 $T\}$ $\{\mathrm{Hom}_2$: _HomT eq1$\}$
$\{\mathrm{Category}_1$: Category $\mathrm{Hom}_2\}$ $\{x\ y\ z\}$ $\{f\ f' : x \sim_1 y\}$ $\{g\ g' : y \sim_1 z\}$:
$f \sim_2 f' \rightarrow g \sim_2 g' \rightarrow g \circ f \sim_2 g' \circ f'$:= comp _ _ _ $f\ f'\ g\ g'$.

Infix "**" := HorComp (at level 50).

Compatibilities in string diagrams



Weak 2-Groupoids

Weak 2-Groupoid is a weak 2-Category where the underlying Categories are Groupoids

$$\mathrm{Groupoid\_1} :> \mathsf{Groupoid}\ \mathrm{Category\_1}$$

$$\mathrm{Groupoid\_2} :> \forall\ x\ \ y,\ \mathsf{Groupoid}\ (\mathrm{Category\_2}\ x\ \ y)$$

`Definition` $\mathrm{Weak2GroupoidType} := \{\, T{:}\mathtt{Type}\ \&\ \mathsf{Weak2Groupoid}\ \ T\ \}.$

Proof irrelevance, Propositional extensionality

The weak 2-groupoid of Props

Prop forms such a degenerated 2-groupoid, with 1-eq logical equivalence of propositions and irrelevant 2-eq representing equality of two proofs of the same proposition.

```
Definition Hom_irr (T : Type) : HomT T := λ _ _, unit.

Class PropIrr (P:Prop) : Type :=
  { prop_irr_groupoid := IrrRelWeak2Groupoid (T:=P) (Hom := Hom_irr P) _ _ _}.
Program Definition Propositions := { P : Prop & PropIrr P }.


Definition iff' : HomT Propositions := fun P Q ⇒ [P] ↔ [Q].

Program Definition _Prop : Weak2GroupoidType :=
  (Propositions ; IrrRelWeak2Groupoid (Hom:=iff') _ _ _).
```

Functions as weak 2-functors

Functor

As for Category, the definition of Functor uses 2 HomTs.

```
Class Functor T U (Hom : HomT T) (Hom₂: _HomT Hom) (Hom' : HomT U) (Hom2': _HomT Hom')
  (Cat : Category Hom₂) (Cat' : Category Hom2') (f : T → U) := {

  map : ∀ {x y}, Hom x y → Hom' (f x) (f y) ;

  map_comp : ∀ x y z (e:Hom x y) (e':Hom y z), map (e' ∘ e) ∼₂ map e' ∘ map e ;
  map_id : ∀ x, map (identity x) ∼₂ identity (f x)
}.
```

Weak2Functor

Using our "open" definition of a Functor
we can a weak 2-functor as a Functor at all levels...

$\texttt{Class}$ Weak2Functor $\{T\ U : \mathrm{Weak2GroupoidType}\}\ (f : [T] \to [U]) : \texttt{Type} :=$
$\{$
$\quad \mathrm{map1} :> \text{Functor}\ (\mathrm{eq\_pi3'}\ T)\ (\mathrm{eq\_pi3'}\ U)\ f;$
$\quad \mathrm{map2} :> \forall\ x\ y,\ \text{Functor}\ (\mathrm{eq\_pi2'}\ T\ x\ y)\ (\mathrm{eq\_pi2'}\ U\ (f\ x)\ (f\ y))\ (\mathrm{map}\ f);$
$\quad \mathrm{map3} : \forall\ (x\ y : [T])\ (e\ e' : x \sim_1 y)\ (E\ E' : e \sim_2 e'),$
$\qquad\qquad (E \sim_3 E') \to \mathrm{map}\ (\mathrm{map}\ f)\ E \sim_3 \mathrm{map}\ (\mathrm{map}\ f)\ E';$

Weak2Functor

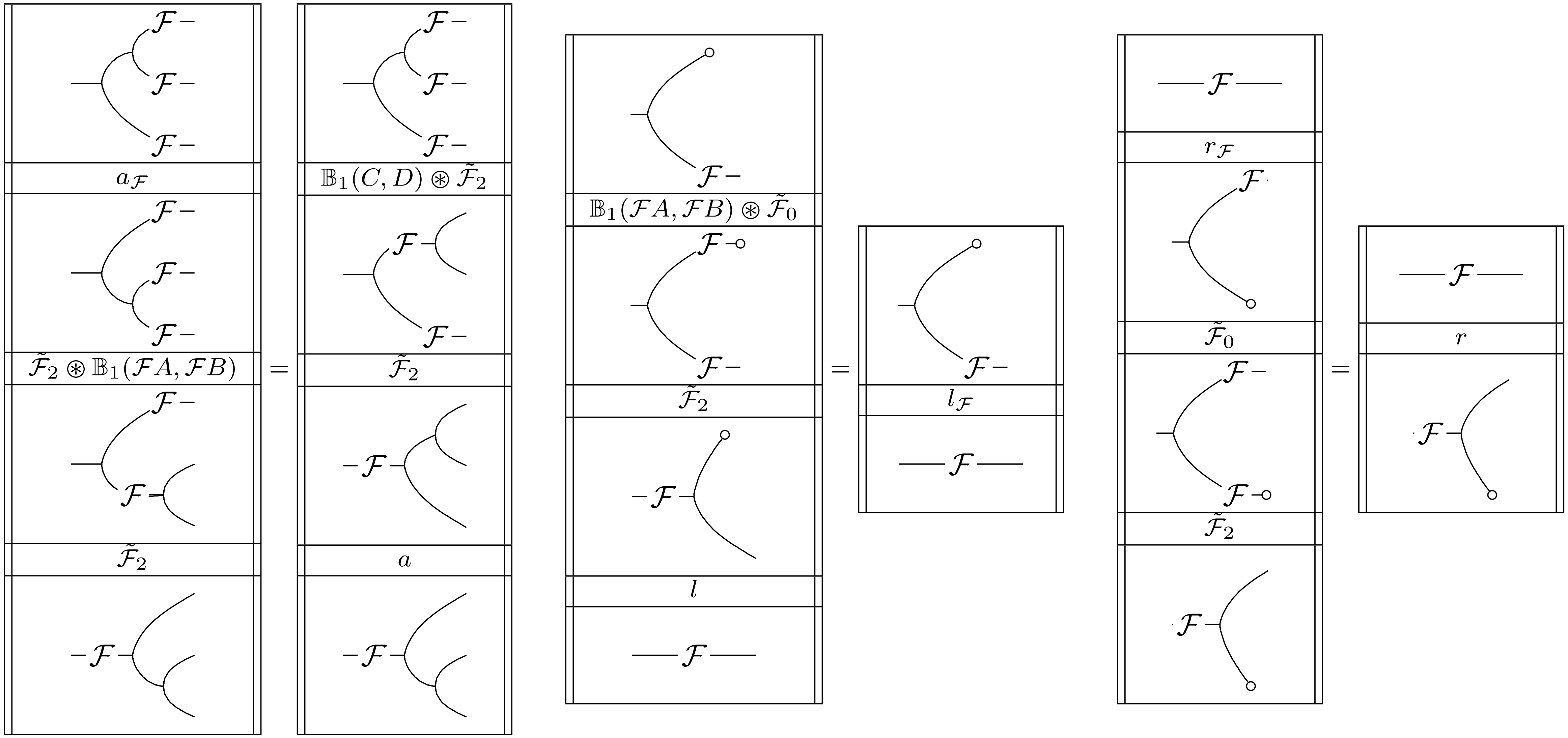
... plus compatibilities, of course

map2_id_L : ∀ (x y : [T]) (e:x $\sim_1$ y),
 map (map f) (id_L' e) $\sim_3$
 id_L' (map f e) ∘ (identity (map f e) ** map_id f _) ∘ map_comp f _ _ ;

map2_id_R : ∀ (x y : [T]) (e:x $\sim_1$ y),
 map (map f) (id_R' e) $\sim_3$
 id_R' (map f e) ∘ (map_id f _ ** identity (map f e)) ∘ map_comp f _ _;

map2_assoc : ∀ (x y z w : [T]) (e:x $\sim_1$ y) (e':y $\sim_1$ z) (e'':z $\sim_1$ w),
 assoc'' ∘ (identity _ ** map_comp f e' e'') ∘ map_comp f e (e'' ∘ e') $\sim_3$
 (map_comp f _ _ ** identity _) ∘ map_comp f (e' ∘ e) e'' ∘ map (map f) assoc''

Compatibilities for Weak2Functor



Weak2Functor

Functions are then functions with a weak 2-functor structure

$$\texttt{Definition}\ \mathrm{Fun\_Type}\ (T\ U : \mathrm{Weak2GroupoidType}) := \{f : [T] \to [U]\ \&\ \textsf{Weak2Functor}\ f\}.$$

$$\texttt{Infix}\ \text{"}\longrightarrow\text{"} := \mathrm{Fun\_Type}\ (\texttt{at}\ \texttt{level}\ 55).$$

Functional Extensionality
vs
natural transformation

Natural Transformation

Equality between weak 2 functor is given by natural transformations

$$\texttt{Definition}\ \text{nat\_trans}\ T\ U\ (f\ g : T \longrightarrow U) :=$$
$$\{\alpha : \forall\ t : [T],\ f \star t \sim_1 g \star t\ \&\ \textsf{NaturalTransformation}\ \alpha\}.$$

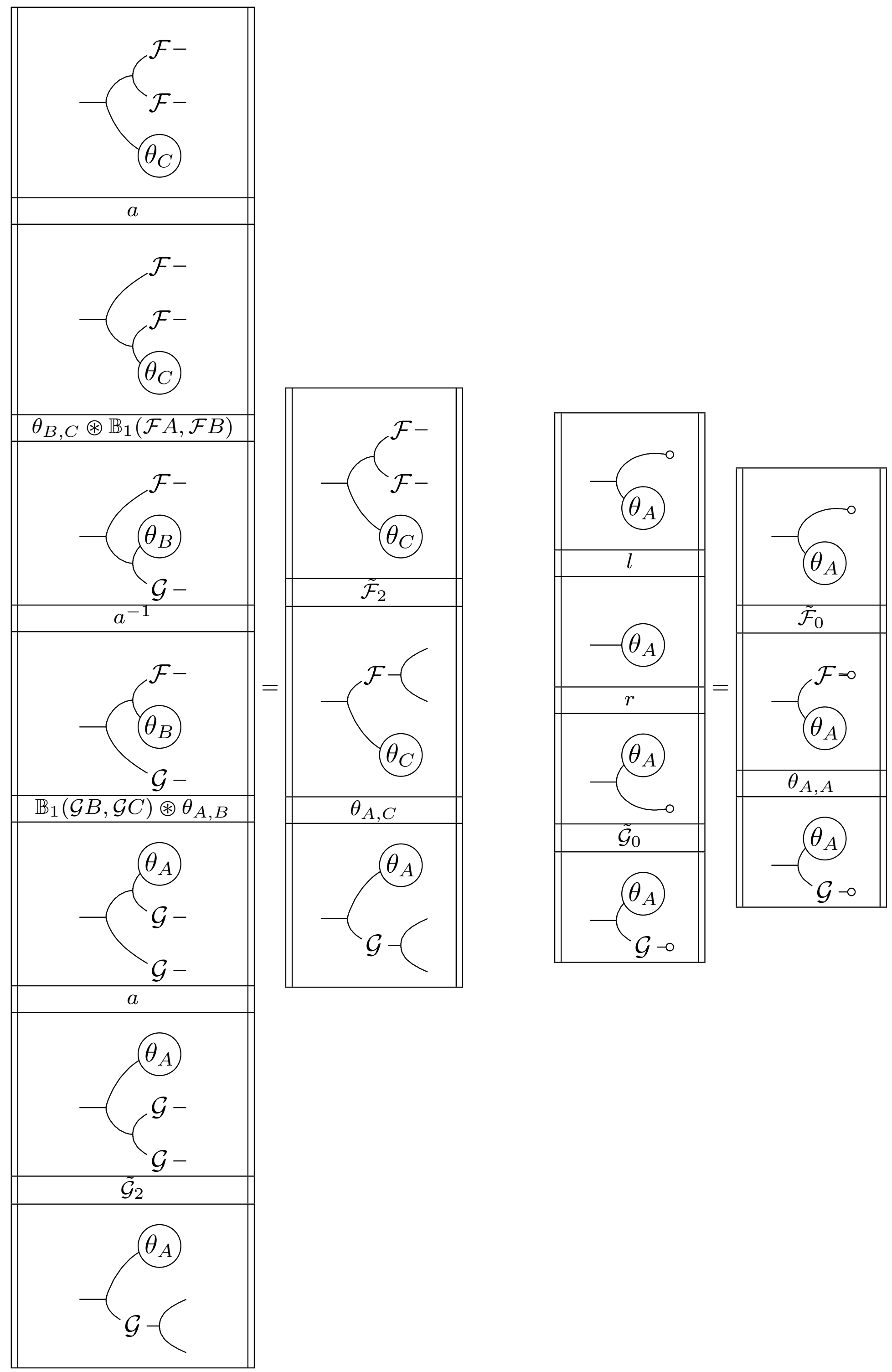
$$\texttt{Notation}\ \text{"M} \star \text{N"} := ([M]\ N)\ (\texttt{at}\ \texttt{level}\ 55).$$

Natural Transformation

Equality between weak 2 functors is given by natural transformation

```
Class NaturalTransformation T U {f g : T ⟶ U} (α : ∀ t : [T], f ⋆ t ~1 g ⋆ t) := {
  α_map : ∀ {t t'} (e : t ~1 t'), (α t') ∘ (map [f] e) ~ (map [g] e) ∘ (α t) ;
  NatTrans_comp : ∀ t t' t'' (e : t ~1 t') (e' : t' ~1 t''),
    (identity _ ** map_comp [g] e e') ∘ α_map (e' ∘ e) ~2
    inverse' (assoc'') ∘ (α_map e ** identity _ ) ∘ assoc'' ∘
    (identity _ ** α_map e') ∘ inverse' (assoc'') ∘
    (map_comp [f] e e' ** identity _);
  NatTrans_id : ∀ t,
      (identity _ ** map_id [g] t) ∘ α_map (identity _) ~2
      inverse' (id_L' _) ∘ id_R' _ ∘ (map_id [f] t ** identity _)
```

Compatibilities for natural trans.



Higher extensionality
vs
modification

Modification

Equality between natural transformations is given by modifications

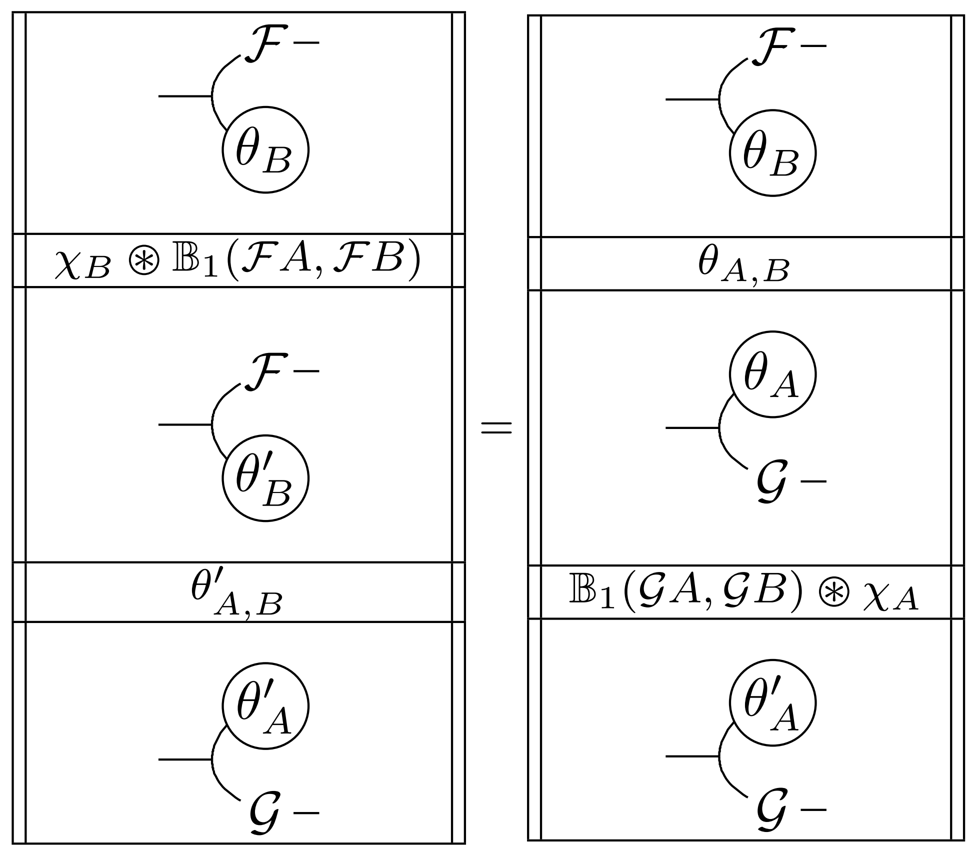
$$\texttt{Definition}\ \text{modification}\ T\ U\ (f\ g : T \longrightarrow U)\ (\alpha\ \beta : \text{nat\_trans}\ f\ g) :=$$
$$\{\chi : \forall\ t : [T],\ \alpha \star t \sim \beta \star t\ \&\ \mathsf{Modification}\ \chi\}.$$

Modification

Equality between natural transformations is given by modifications

$$
\begin{array}{l}
\texttt{Class}\ \mathsf{Modification}\ T\ U\ \{f\ g : T \longrightarrow U\}\ \{\alpha\ \beta : \mathrm{nat\_trans}\ f\ g\} \\
\qquad (\chi : \forall\ \{t\},\ \alpha \star t \sim \beta \star t) := \{ \\
\quad \chi\mathrm{\_map} : \forall\ \{t\ t'\}\ (e : t \sim_1 t'), \\
\qquad\qquad \alpha\mathrm{\_map}\ \_\ e \circ (\mathrm{identity}\ \_\ {**}\ \chi) \sim_3 \\
\qquad\qquad (\chi\ {**}\ \mathrm{identity}\ \_) \circ \alpha\mathrm{\_map}\ \_\ e \\
\}.
\end{array}
$$

Compatibility for modifications



We can form a weak 2-1 category whose:

- objects are types with a weak-2-groupoid structure
- 1-cells are weak 2-functors
- 2-cells are natural transformation
- 3-cells are modications

Equality on Types vs Homotopic Equivalence

Homotopic equivalence

Equivalence of weak 2-groupoids is homotopic equivalence:

- a map with its adjoint
- 2 proofs that they form a section and a retraction
- 2 triangle identities relating section and retraction.

Homotopic equivalence

$\texttt{Class}$ $\mathsf{Equiv\_struct}$ $T\ U\ (f : [T \longrightarrow U]) := \{$
$\quad$ adjoint : $[U \longrightarrow T]$;
$\quad$ section : $f \circ$ adjoint $\sim_1$ identity U ;
$\quad$ retraction : identity $T \sim_1$ adjoint $\circ f$;
$\quad$ triangle : $\forall\ t$, (section $\star$ _) $\circ$ map _ (retraction $\star\ t$) $\sim$ identity _;
$\quad$ triangle' : $\forall\ u$, map _ (section $\star\ u$) $\circ$ (retraction $\star$ _) $\sim$ identity _$\}$.

$\texttt{Definition}$ Equiv $A\ B := \{f : A \longrightarrow B\ \&\ \mathsf{Equiv\_struct}\ f\}$.

Equivalence of adjunction

Two adjunctions are equivalent if their left adjoint are equivalent and they agree on their section and retraction (the right adjoints always agree then)

```
Definition Equiv_adjoint A B (f f': Equiv A B) :
  [f] ~1 [f'] → adjoint [f] ~1 adjoint [f'].

Record Equiv_eq T U (f g : T <~> U) : Type :=
  {equiv :> nat_trans [f] [g] ;
   eq_section : ∀ u,
      section [f] ⋆ u ~
      section [g] ∘ (nat_comp' (Equiv_adjoint _ _ equiv) equiv) ⋆ u;
   eq_retraction : ∀ t,
      (nat_comp' equiv (Equiv_adjoint _ _ equiv)) ∘ retraction [f] ⋆ t ~
      retraction [g] ⋆ t
}.
```

The weak 2-groupoid of Types

$\texttt{Program Definition}\ \_\mathrm{Type} : \mathrm{Weak2GroupoidType}$
$:= (\mathrm{Weak2GroupoidType}\ ;\ \_).$

_Type is a weak 2-groupoid whose:

- objects are types with a weak-2-groupoid structure
- 1-equivalences are homotopic equivalences
- 2-equivalences are adjoint equivalences
- 3-equivalences are modifications

The weak 2-groupoid of Types

Note that Weak2GroupoidType is used at two different universe levels here.

Note also that _Type is a weak 3-groupoid, but it cannot be expressed in our formalism.

Rewriting in Homotopy Type Theory

Rewriting with compatibility maps

The *map* function on F: $[A \longrightarrow \_\mathrm{Type}]$ gives an (homotopic) equivalence $F \star x \sim_2 F \star y$ if $x \sim_1 y$.
This means we can rewrite $[F \star x]$ into $[F \star y]$ using the function part of the equivalence.

$$\texttt{Definition}\ \mathrm{eq\_rect'}\ (A : [\_\mathrm{Type}])\ (x : [A])\ (F\!: [A \longrightarrow \_\mathrm{Type}])\ (y : [A])$$
$$(e : x \sim_1 y) := [\mathrm{map}\ [F]\ e] : (F \star x) \longrightarrow (F \star y).$$

Rewriting with compatibility maps

We derive (some, not all) coherence theorems on $\mathrm{eq\_rect}$ according to the ones on *map*. E.g., the usual reduction for $\mathrm{eq\_rect}$ is derivable:

$$\begin{array}{l}
\texttt{Definition}\ \mathrm{eq_rect_id}\ \{T{:}[\mathrm{_Type}]\}\ \{F : [T \longrightarrow \mathrm{_Type}]\} \\
\quad (x : [T])\ (p : [F \star x]) \\
\quad : \mathrm{eq_rect}\ (\mathrm{identity}\ x)\ p \sim_1 p := (\mathrm{equiv}\ (\mathrm{map_id}\ F\ x)) \star p.
\end{array}$$

$\mathrm{eq\_rect}$ is compatible with higher equivalences as well:

$$\begin{array}{l}
\texttt{Definition}\ \mathrm{eq_rect_eq}\ \{A : [\mathrm{_Type}]\}\ \{x : [A]\} \\
\qquad\qquad\qquad \{F{:}\ [A \longrightarrow \mathrm{_Type}]\} \\
\quad \{y : [A]\}\ \{e\ e'{:}\ x \sim_1 y\}\ (H : e \sim_2 e')\ (p : [F \star x]) : \\
\quad \mathrm{eq_rect}\ e\ p \sim_1 \mathrm{eq_rect}\ e'\ p := (\mathrm{equiv}\ (\texttt{map2}\ [F]\ H)) \star p.
\end{array}$$

Dependent product (work in progress)

Traditional interpretation

In homotopy type theory, dependent sums and products are interpreted using sections and projections.

We look for more direct/computational definitions

Dependent Functors

The dependent product gives rise to dependent functors.

The map component needs some adjustment by equalities due to dependencies.

```
Class DependentFunctor (T:[_Type]) (U : [T ⟶ _Type]) (f : ∀ t, [U ⋆ t]) := {

  Dmap : ∀ {x y} (e: x ~1 y), eq_rect' _ U _ e ⋆ (f x) ~1 f y ;

  Dmap_comp : ∀ x y z (e : x ~1 y) (e' : y ~1 z),
      Dmap (e' ∘ e) ∘ (inverse _ _ (eq_rect'_comp _ _ _ U e e') ⋆ _) ~
      Dmap e' ∘ eq_rect'_map _ U _ _ _ _ (Dmap e);

  Dmap_id : ∀ x, Dmap (identity x) ~ eq_rect'_id _ x ⋆ (f x)
 }.
```

Dependent Functors

For the moment, the formalism is not uniform in all level, because of the different rewriting terms

```
Class DependentFunctor2 (T:[_Type]) (U : [T ⟶ _Type]) (f : ∀ t, [U ⋆ t]) x y
      (F : ∀ (e: x ~1 y), eq_rect' _ U _ e ⋆ (f x) ~1 f y) := {

  Dmap2 : ∀ {e e': x ~1 y} (H: e ~ e'),
      F e ~ F e' ∘ (eq_rect'_eq x U y e e' H ⋆ (f x)) ;

  Dmap2_comp : ∀ (e e' e'' : x ~1 y) (H: e ~2 e') (H' : e' ~ e''),
      ((eq_rect'_eq_comp _ _ _ _ _ _ _ _ _ ⋆ _) ** identity _) ∘ Dmap2 (H' ∘ H) ~
      assoc _ _ _ _ _ _ _ ∘ (identity _ ** Dmap2 H') ∘ Dmap2 H;

  Dmap2_id : ∀ (e : x ~1 y),
    ((eq_rect'_eq_id _ x y e ⋆ (f x)) ** identity (F e)) ∘ Dmap2 (identity e) ~2
    inverse _ _ (id_R _ _ (F e))

}.
```

Dependent Functors

Higher compatibilities are not present for the moment in the model, are they required ?

```
Class Weak2DependentFunctor (T:[_Type]) (U : [T ⟶ _Type]) (f : ∀ t, [U ⋆ t]) : Type
:=
{
 _Dmap1 :> DependentFunctor U f;
 _Dmap2 :> ∀ (x y : [T]), DependentFunctor2 U f x y (Dmap f);
 _Dmap3 : ∀ (x y : [T]) (e e' : x ∼1 y) (E E' : e ∼2 e') (H : E ∼3 E'),
      ((eq_rect'_eq_eq U x y e e' E E' H ⋆ (f x)) ** identity _) ∘
        Dmap2 (Dmap f) E ∼3
        Dmap2 (Dmap f) E'
}.
```

Dependent Sums

The interpretation of dependent sums is more direct

$$\texttt{Definition}\ \mathrm{sum\_type}\ (T\!: [\_\mathrm{Type}])\ (F : \mathrm{Weak1Fibration}\ T) :=$$
$$\{t : [T]\ \&\ [[F] \star t]\}.$$

$$\texttt{Definition}\ \mathrm{sum\_eq}\ (T\!: [\_\mathrm{Type}])\ (F : \mathrm{Weak1Fibration}\ T) :=$$
$$\lambda\ (m\ n : \mathrm{sum\_type}\ F),\ \{P : [m] \sim_1 [n]\ \&\ \mathrm{eq\_rect'}\ \_\ [F]\ \_\ P \star (\pi_2\ m) \sim_1 \pi_2\ n\}.$$

Dependent Sums

Problem: To show that it gives rise to a weak 2-groupoid, we are missing higher compatibilities on _Type, because we forgot it 3-dimensional structure.

(partial) solution: restrict the definition of dependent sums to weak 1-groupoids

$$\texttt{Definition}\ \mathrm{Weak1Groupoid}\ (T : [\_\mathrm{Type}]) : \texttt{Type} :=$$
$$\forall\ (x\ y : [T])\ (f\ g\colon x \sim_1 y)\ (\alpha\ \beta\colon f \sim_2 g),\ \alpha \sim_3 \beta.$$

$$\texttt{Definition}\ \mathrm{Weak1Fibration}\ (T : [\_\mathrm{Type}]) : \texttt{Type} :=$$
$$\{\ F : [T \longrightarrow \_\mathrm{Type}]\ \&\ \forall\ t,\ \mathrm{Weak1Groupoid}\ (F \star t)\}.$$

The translation
(work in progress)

The translation

The translation is still incomplete.

Underscores represent obligations to show functoriality/naturality conditions ⇒ should be automatic

$$
\begin{aligned}
&[\![\ \texttt{Type}\ ]\!] \equiv \_\mathrm{Type}\\
&[\![\ \texttt{Prop}\ ]\!] \equiv \_\mathrm{Prop}\\
&[\![\ T \to U\ ]\!] \equiv [\![\ T\ ]\!] \longrightarrow [\![\ U\ ]\!]\\
&[\![\ \forall\ t:\ T,\ U\ ]\!] \equiv \_\mathrm{Prod}\ [\![\ (\lambda\ t,\ U\ ;\ \_)\ ]\!]\\
&[\![\ \lambda\ t:\ T,\ M\ ]\!] \equiv (\lambda\ t:\ [\![\ T\ ]\!]\ ,\ [\![\ M\ ]\!]\ ;\ \_)\\
&[\![\ x{:}A\ ]\!] \equiv x:\ [\ [\![\ T\ ]\!]\ ]\\
&[\![\ M\ \ N\ ]\!] \equiv [\![\ M\ ]\!] \star [\![\ N\ ]\!]\\
&[\![\ \{t:\ T,\ U\}\ ]\!] \equiv \_\mathrm{Sum}\ [\![\ (\lambda\ t,\ U\ ;\ \_)\ ]\!]\\
&[\![\ \pi_i\ M\ ]\!] \equiv \pi_i\ [\![\ M\ ]\!]
\end{aligned}
$$

Extensional properties as lemmas (again)

$\texttt{Lemma}$ prop_extensional $(P\ Q : [\_\mathrm{Prop}]) : [P] \leftrightarrow [Q] \rightarrow P \sim_1 Q.$

$\texttt{Lemma}$ proof_irrelevant $(P : [\_\mathrm{Prop}])\ (p\ q : [P]) : p \sim_1 q.$

$\texttt{Lemma}$ functional_extensionality $A\ B\ (f\ g : [A \longrightarrow B]) :$
nat_trans $f\ g \rightarrow f \sim_1 g.$

$\texttt{Lemma}$ sum_extensional $T\ F\ (m\ n : [\_\mathrm{Sum}\ (T{:=}T)\ F]) :$
$\forall\ (P : [m] \sim_1 [n]),$ eq_rect' $\_\ [F]\ \_\ P \star (\pi_2\ m) \sim_1 \pi_2\ n \rightarrow m \sim_1 n.$

$\texttt{Lemma}$ univalence_statement $(U\ V : [\_\mathrm{Type}]) : (\mathrm{Equiv}\ U\ V) \rightarrow U \sim_1 V.$

An Example

Church naturals vs inductive naturals

- Define the (discrete) weak 2-groupoid of inductive natural numbers: $(\mathsf{nat}, \mathsf{eq}, \mathrm{Hom\_irr}, \ldots)$.
  We need to use UIP on nat to show this forms a groupoid.
- Derive the weak 2-groupoid of church naturals $\mathrm{cnat} := \Pi X : \mathtt{Type}, X \to (X \to X) \to X$ using the function space and dependent product groupoid constructors.
- Prove equivalence (i.e., isomorphism) of the two groupoids.
  This requires parametricity at cnat (see Keller and Lasson).

We can now transport the translation of any theorem on nat to cnat... This requires showing these theorems are functorial of course.

Conclusion

Univalence should be for free

Direction:

- Automate compatibility conditions
- Internalize this, 2-dimensional type theory/OTT-style