UniMath - its present and its future

by Vladimir Voevodsky

from the Institute for Advanced Study
in Princeton, NJ.

Big Proof program workshop, Cambridge UK,
July 10, 2017

The name UniMath may refer to several things. Among them are:

1. A univalent foundation of mathematics.

2. A subset of the type-in-type mode of the language of proof assistant Coq that implements the formal deductive system of this foundation.

3. The library, freely available on GitHub under the name UniMath, of mathematics formalized in this system.

By a foundation of mathematics we understand an object with three necessary components:

1. A formal deduction system such as the first order logic with a distinguished theory or a Martin-Lof type theory or a subset of the type theory of a proof assistant such as Coq, Agda, Lean or any other proof assistant.

2. An interpretation of the primitives of the formal deduction system as mathematical objects or actions.

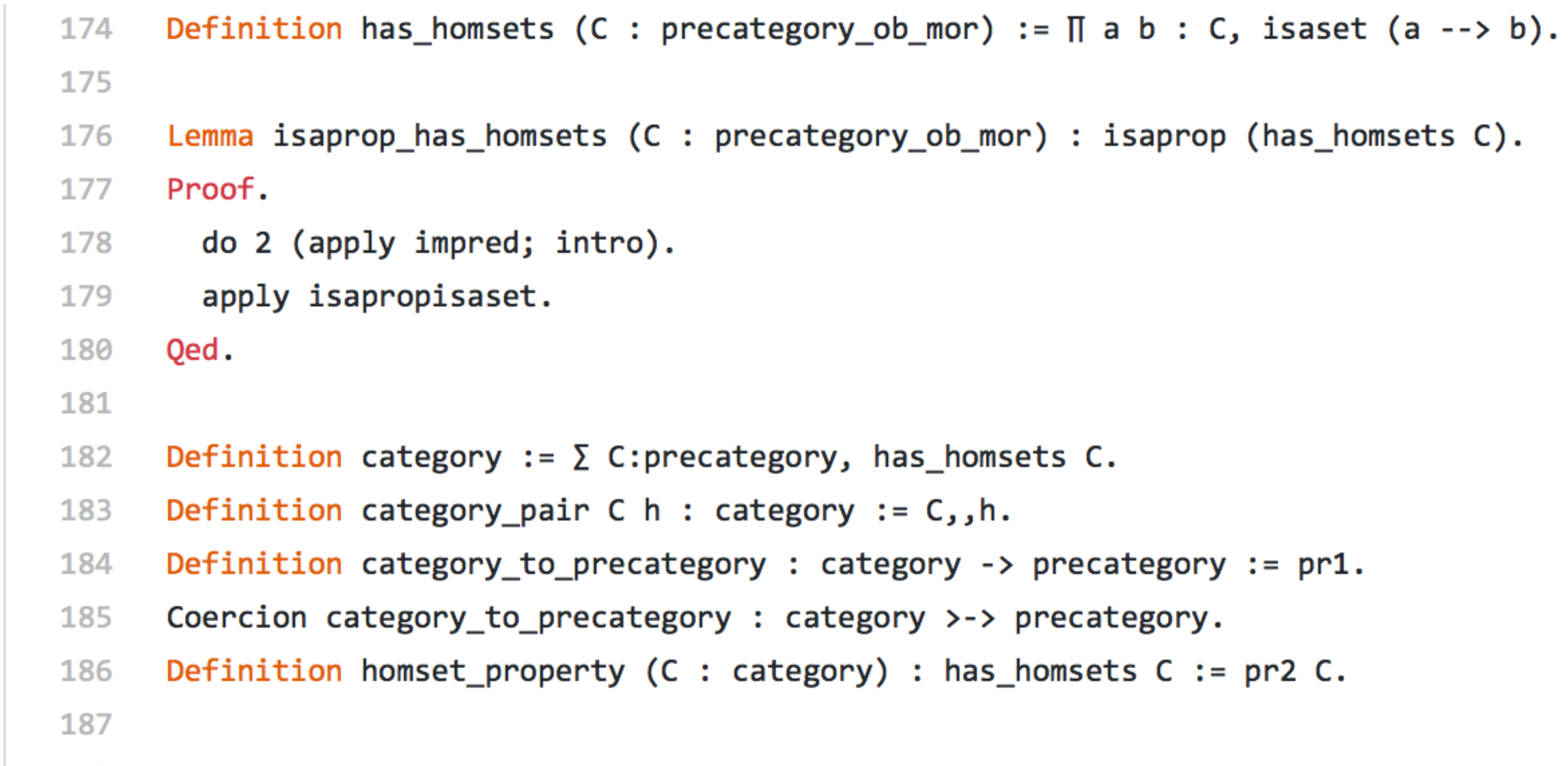
3. A collection of fundamental constructions that, using the interpretation of primitives, defines formal counterparts for the main mathematical concepts such as set, function or natural number.

The formal deduction system of the UniMath is a subset of the formal deduction system of the type-in-type mode of Coq. It contains the following type constructors:

Dependent product, unit type, dependent sum, empty type, disjoint sum, booleans, natural numbers, identity types and the universe.

This outlines the first component of the UniMath as a foundation.

Here is an example of UniMath code. In these lines we define the type of categories from the type of precategories. The meaning of these two concepts we will discuss in a few minutes.



```
Definition has_homsets (C : precategory_ob_mor) := ∏ a b : C, isaset (a --> b).

Lemma isaprop_has_homsets (C : precategory_ob_mor) : isaprop (has_homsets C).
Proof.
  do 2 (apply impred; intro).
  apply isapropisaset.
Qed.

Definition category := ∑ C:precategory, has_homsets C.
Definition category_pair C h : category := C,,h.
Definition category_to_precategory : category -> precategory := pr1.
Coercion category_to_precategory : category >-> precategory.
Definition homset_property (C : category) : has_homsets C := pr2 C.

```

If we do not use the resizing rules (a part of the UniMath that we will discuss later if we have enough time) then we believe that we know how to interpret any construction of the UniMath as a construction where

types are Kan simplicial sets, elements of types are points of these sets, elements of the identity types are paths between the corresponding points and the universe is the base of the universal Kan fibration.

The element of belief here comes from the Initiallity Conjecture that we hope to prove in the foreseeable future.

With some experience one can learn to automatically "see" the UniMath sentences as referring to such objects and use geometric intuition to construct proofs.

This outlines the second component of the UniMath as a foundation.

Now we need to say a few words about the third component - the collection of constructions that, using the interpretation of primitives, defines formal counterparts for the main mathematical concepts.

Sometimes I refer to these concepts and constructions as the main concepts and constructions of the univalent type theory.

The first, possibly most fundamental, concept of univalent type theory is the concept of a contractible type:

$$\mathsf{iscontr}(\mathsf{T}) := \sum_{\mathsf{t0:T}} \prod_{\mathsf{t1:T}} \mathsf{paths}\,\mathsf{t1}\,\mathsf{t0}$$

where **paths** is the identity type. An element of $\mathsf{iscontr}(\mathsf{T})$ is interpreted as a proof that T is contractible.

In the simplicial set interpretation $\mathsf{iscontr}(\mathsf{T})$ is a Kan simplicial set that is non-empty if and only if the s. set corresponding to T is contractible in which case it is also contractible.

Next, one defines the concept of h-level by induction such that

$$\mathsf{isofhlevel}\,0\,\mathsf{T} := \mathsf{iscontr}(\mathsf{T})$$

$$\mathsf{isofhlevel}\,(\mathsf{S}\,\mathsf{n})\,\mathsf{T} := \prod_{\mathsf{t0,t1:T}} \mathsf{isofhlevel}\,\mathsf{n}\,(\mathsf{paths}\,\mathsf{t0}\,\mathsf{t1})$$

where $\mathsf{S}\,\mathsf{n}$ is the successor of n in nat that in Coq is definitionally equal to $\mathsf{1}+\mathsf{n}$.

Types of h-level 1 are called *propositions*. Types of h-level 2 are called *sets*. Types of h-level $\mathsf{n} \geq \mathsf{2}$ and higher are also called (n-2)-types. Sometimes one also says (-1)-types about types of h-level 1.

In the s.set interpretation types of h-level 1 are either contractible or empty and thus correspond to truth values and types of h-level 2 are exactly the Kan s.sets homotopy equivalent to sets.

A *homotopy* between two functions $\mathsf{f}, \mathsf{g} : \mathsf{T}_1 \to \mathsf{T}_2$ is an element of the type

$$\mathsf{homot}\,\mathsf{f}\,\mathsf{g} \;:=\; \prod_{\mathsf{t}:\mathsf{T}} \mathsf{paths}\,(\mathsf{f}\,\mathsf{t})\,(\mathsf{g}\,\mathsf{t})$$

A fiber, or *h-fiber*, of a function $\mathsf{f} : \mathsf{T}_1 \to \mathsf{T}_2$ over $\mathsf{t}_2 : \mathsf{T}_2$ is the type

$$\mathsf{hfiber}\,\mathsf{f}\,\mathsf{t}_2 \;:=\; \sum_{\mathsf{t}_1:\mathsf{T}_1} \mathsf{paths}\,(\mathsf{f}\,\mathsf{t}_1)\,\mathsf{t}_2$$

We have two concepts of equivalence. A *naive equivalence* between types T_1 and T_2 is a quadruple $(\mathsf{f}_1, \mathsf{f}_2, \mathsf{h}_{12}, \mathsf{h}_{21})$ where

$$\mathsf{f}_1 : \mathsf{T}_1 \to \mathsf{T}_2$$

$$\mathsf{f}_2 : \mathsf{T}_2 \to \mathsf{T}_1$$

are functions and

$$\begin{aligned} \mathsf{h}_{12} \; &: \; \mathsf{homot}\,(\mathsf{f}_1 \circ \mathsf{f}_2)\,(\mathsf{idfun}\,\mathsf{T}_1) \\ \mathsf{h}_{21} \; &: \; \mathsf{homot}\,(\mathsf{f}_2 \circ \mathsf{f}_1)\,(\mathsf{idfun}\,\mathsf{T}_2) \end{aligned}$$

are homotopies.

An *equivalence* between types T_1 and T_2 is a pair (f, w) where $\mathsf{f} : \mathsf{T}_1 \to \mathsf{T}_2$ is a function and w an element of the type $\mathsf{isweq}\,\mathsf{f}$ where

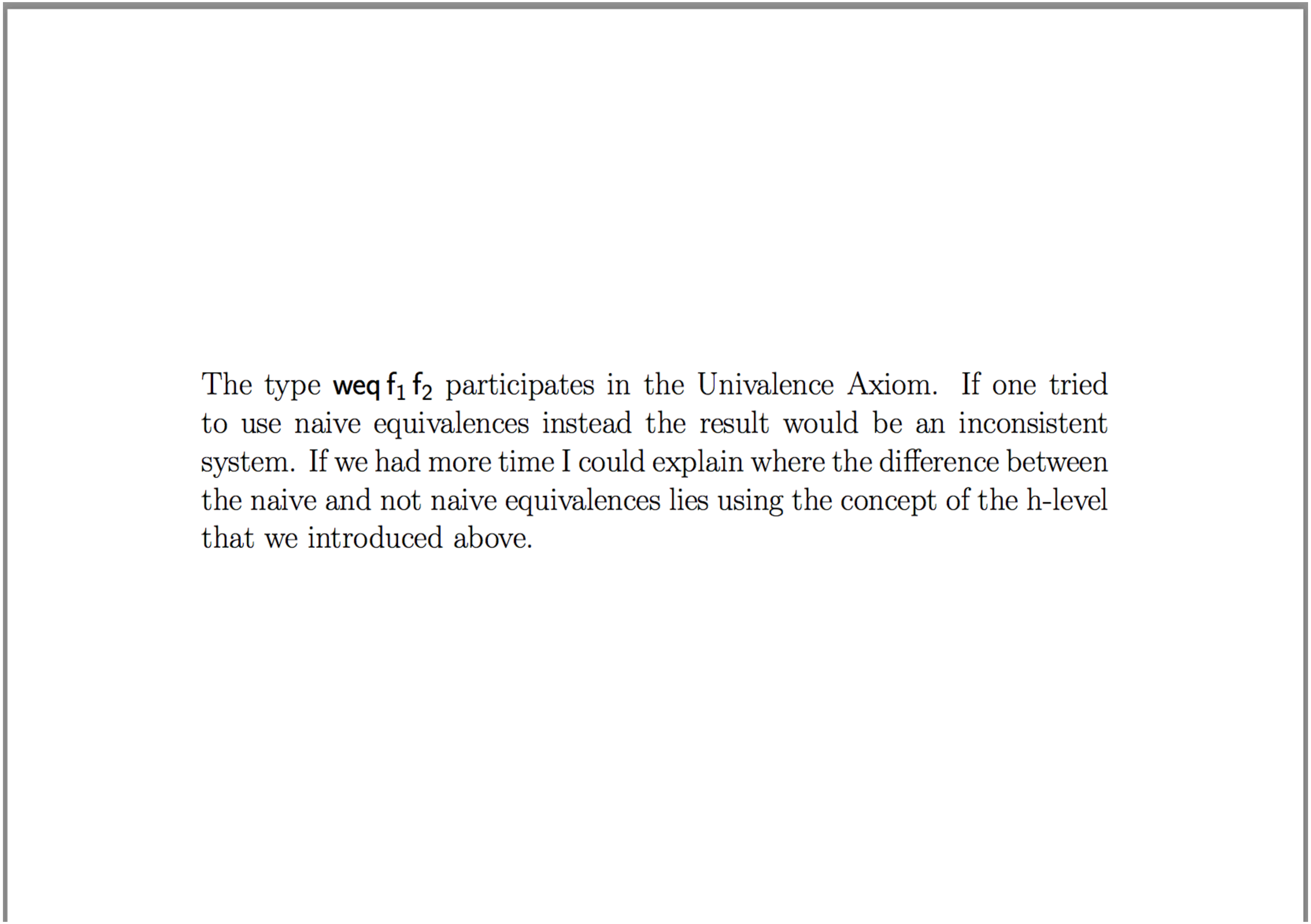
$$\mathsf{isweq}\,\mathsf{f} \;:=\; \prod_{\mathsf{t}_2 : \mathsf{T}_2} \mathsf{iscontr}(\mathsf{hfiber}\,\mathsf{f}\,\mathsf{t}_2)$$

The type of equivalences is written as $\mathsf{weq}\,\mathsf{T}_1\,\mathsf{T}_2$.

There is a theorem, that is actually a construction, called gradth of the form

$$\mathsf{gradth}(\mathsf{f}_1 : \mathsf{T}_1 \to \mathsf{T}_2)(\mathsf{f}_2 : \mathsf{T}_2 \to \mathsf{T}_1)(\mathsf{h}_{12} : \mathsf{homot}\,(\mathsf{f}_1 \circ \mathsf{f}_2)\,(\mathsf{idfun}\,\mathsf{T}_1))$$
$$(\mathsf{h}_{21} \;:\; \mathsf{homot}\,(\mathsf{f}_2 \circ \mathsf{f}_1)\,(\mathsf{idfun}\,\mathsf{T}_2)) : \mathsf{isweq}\,\mathsf{f}_1$$

that allows us to construct equivalences from the naive equivalences.



The type **weq** f_1 f_2 participates in the Univalence Axiom. If one tried to use naive equivalences instead the result would be an inconsistent system. If we had more time I could explain where the difference between the naive and not naive equivalences lies using the concept of the h-level that we introduced above.

After these necessarily brief explanations of the UniMath as a foundation and of the implementation of the UniMath formal deduction system in Coq let me pass to the main topic of the lecture, the ***UniMath library***.

I will attempt to give a feeling about this library by giving examples.

The UniMath library today grows in two way - directed and independent. That is, either someone who has already been working on the library and knows what is missing and needs to be added gives a task to someone who wants to start working on it, or someone who wants to formalize some piece of mathematics decides that the UniMath is the right tool for it and does it.

Two examples of the second kind are the work of Tomi Pannila who has formalized a part of the theory of triangulated categories and the work of Anthony Bordg who is formalizing the theory of modules over rings and some other things.

Another example of the second kind is the work that Benedikt Ahrens, Peter Lumsdaine and myself are doing on developing and formalizing the theory of categorical structures of type theories in univalent foundations. More about this last example later.

Here are some examples of the first kind. They are study tasks that were done by Matthew Weaver, a graduate student at Princeton.

Problem 1. Let HSET be the category of sets. Given a set S to construct an equivalence between categories HSET/S and Funct S HSET where in the latter S is considered as a discrete category.

In the UniMath there are three different types of objects corresponding to the concept of a (small) category. There are precategory, category and univalent_category. Basics of the univalent category theory can be found in a paper by Ahrens, Kapulkin and Shulman "Univalent categories and the Rezk completion".

A ***precategory*** has a type of objects, a family of types of morphisms, a family of composition functions, identity morphisms and the three usual axioms.

A ***category*** is a precategory where types of morphisms are sets.

A ***univalent_category*** is a category where the type of isomorphisms between two objects is equivalent to the type of paths between these two objects.

Note that types form a precategory. Sets form a category. If we add the univalence axiom than we can prove that the category of sets is univalent.

Matt's construction to this problem given in

UniMath/UniMath/CategoryTheory/set_slice_fam_equiv.v

uses the univalent definition of a set. However, the problem can be reformulated for all types. I do not know if there is an equivalence in this more general setting.

Problem 2. Let C be a precategory and P a presheaf of sets on C. Let $\int P$ be the category of elements of P.To construct an equivalence between precategories $\int P$ and (PreShv C)/P.

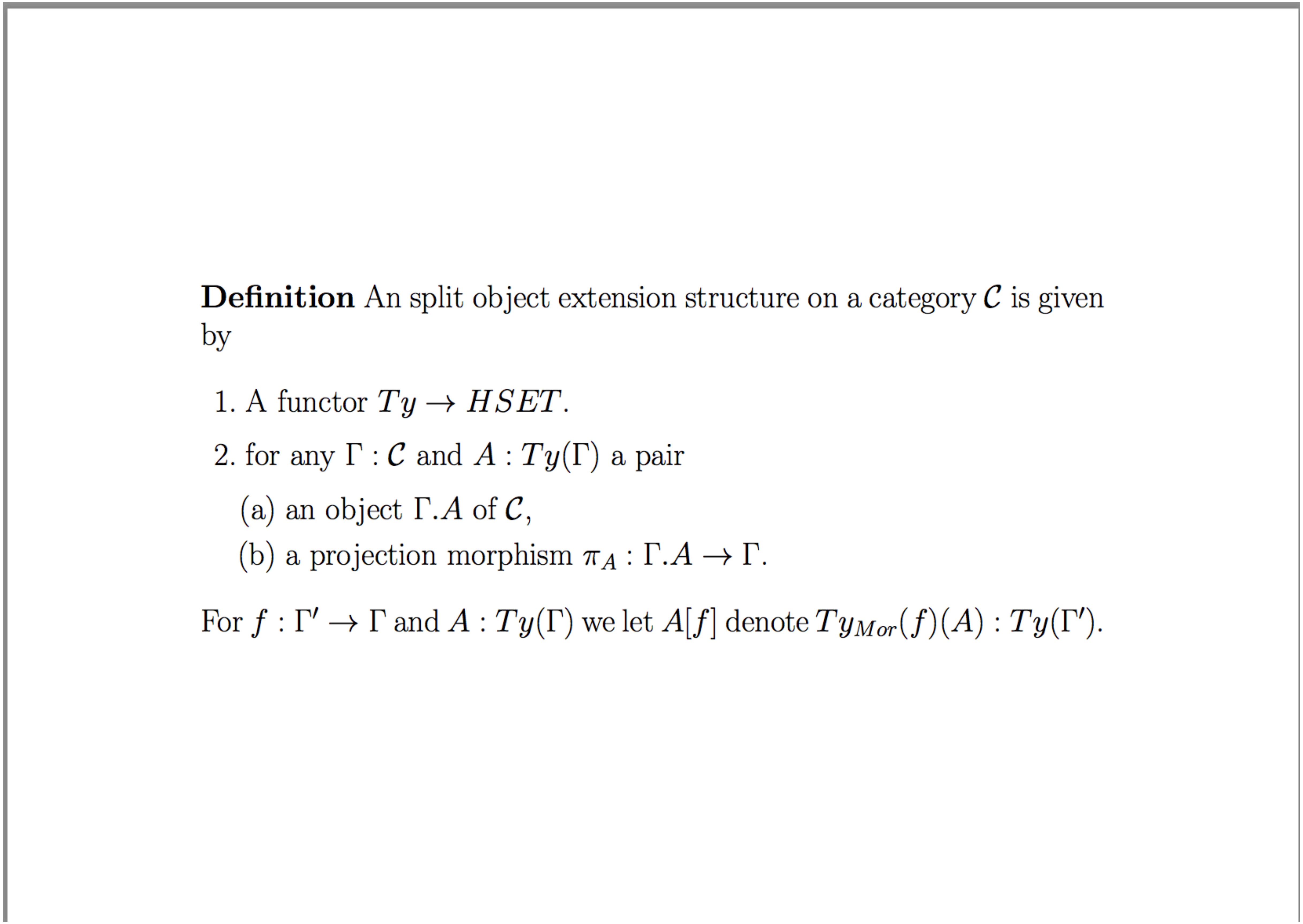
Here the construction given in

UniMath/UniMath/CategoryTheory/elems_slice_equiv.v

works for all precategories C.

The problem in our next example is more complex and a formalized construction for it is at the center of a recent paper by Benedikt Ahrens, Peter LeFanu Lumsdaine and myself "*Categorical structures for type theory in univalent foundations*".

It can be found in the second repository TypeTheory of the UniMath "organization" in UniMath/TypeTheory/



Definition An split object extension structure on a category $\mathcal{C}$ is given by

1. A functor $Ty \to HSET$.
2. for any $\Gamma : \mathcal{C}$ and $A : Ty(\Gamma)$ a pair
   (a) an object $\Gamma.A$ of $\mathcal{C}$,
   (b) a projection morphism $\pi_A : \Gamma.A \to \Gamma$.

For $f : \Gamma' \to \Gamma$ and $A : Ty(\Gamma)$ we let $A[f]$ denote $Ty_{Mor}(f)(A) : Ty(\Gamma')$.

Definition Let $\mathcal{C}$ be a category with an object extension structure X. A functorial term structure on X is given by

1. A functor $Tm : \mathcal{C} \to HSET$.
2. A natural transformation $p : Tm \to Ty$,
3. For any Γ and $A : Ty(\Gamma)$, an element $te_A : Tm(\Gamma.A)$ such that the square

$$\begin{tikzcd} Yo(\Gamma.A) \arrow[r, "y(te_A)"] \arrow[d, "Yo(\pi_A)"'] & Tm \arrow[d, "p"] \\ Yo(\Gamma) \arrow[r, "y(A)"'] & Ty \end{tikzcd}$$

is a pullback.

Definition Let $\mathcal{C}$ be a category and X an object extension structure on $\mathcal{C}$. A split q-morphism structure on X is given by

1. For any $f : \Gamma' \to \Gamma$ in $\mathcal{C}$ and $A : Ty(\Gamma)$ a morphism $\mathsf{q}_{(f,A)} : \Gamma.A \to \Gamma$ such that the square

$$\begin{tikzcd}
\Gamma'.A[f] \arrow[r, "\mathsf{q}_{(f,A)}"] \arrow[d, "\pi_{A[f]}"'] & \Gamma.A \arrow[d, "\pi_A"] \\
\Gamma' \arrow[r, "f"'] & \Gamma
\end{tikzcd}$$

   is pullback.

2. For any $A : Ty(\Gamma)$ we have $e : A[1_\Gamma] = A$ and

$$\mathsf{q}_{(1_\Gamma,A)} = idtoiso(ap_{x \mapsto \Gamma.x}(e))$$

3. For $f' : \Gamma'' \to \Gamma'$, $f : \Gamma' \to \Gamma$ and $A : Ty(\Gamma)$ we have $e' : A[f'f] = A[f][f']$ and

$$\mathsf{q}_{(f'f,A)} = idtoiso(ap_{x \to \Gamma''.x}(e'))\,\mathsf{q}_{(f',A[f])}\,\mathsf{q}_{(f,A)}$$

Problem For a category $\mathcal{C}$ and a split object extension structure X on $\mathcal{C}$ to construct an equivalence between the types of functorial term structures and split q-morphism structures on X.

Construction See UniMath/TypeTheory and ”*Categorical structures for type theory in univalent foundations*”.

The type of the structures of a category with families on a category is easily shown to be equivalent to the type of pairs of an object extension structure and a functorial term structure on it.

The type of structures of split type categories on a category is easily shows to be equivalent to the type of pairs of an object extension structure and a split q-morphism structure on it.

Consequently, our construction provides a construction between the types of category with families structures and split type category structures on a given category and opens up a way to study these structures in the univalent foundations.

As I have mentioned it is impossible to fully predict the direction in which the UniMath will develop in the future because an important part of the UniMath development arises from new participants with their own ideas.

What I can outline are the directions that I see as interesting and important and whose development has either already started or is expected to start.

The first direction is the development and formalization of the mathematics surrounding the study of syntax and semantics of dependent type theories.

This direction itself has now branched into several subdirections.The most clearly aimed among those is the one whose goal is to formalize the construction of the univalent simplicial set model.

Its development progresses well.

In one branch it calls for the development of a 2-category library in the UniMath.

There are two approaches to 2-precategories. Either a 2-precategory is a precategory with precategory structures on the types of morphisms compatible with compositions or a 2-precategory is a type with a family of precategories of morphisms where compositions are functors.

We currently have one library written from the second perspective by Mitchell Riley. However for several reasons the first perspective seems more natural for the UniMath and we plan to start writing a library based on this perspective as well.

Another direction is the one that I have stated in my Bernays lectures at the ETH in 2014 - to formalize a proof of Milnor's conjecture on Galois cohomology.

It has not been developing much. On the one hand, I discovered that formalizing it classically it is not very interesting to me because I am quite confident in that proof and in its extension to the Bloch-Kato Conjecture.

On the other hand, when planning a development of a constructive version of this proof one soon encounters a problem. The proof uses the so called Markurjev-Suslin transfinite argument that relies on the Zermelo's well-ordering theorem that in turn relies on the axiom of choice for sets.

There has been attempts to avoid this argument since the very first proofs of particular cases by Merkurjev and Suslin in the early nineteen eighties. However, it remains an open problem. Solving this problem would be very beneficial for the whole field. The interest in it now increases and we may hope for some progress.

Until then we can develop UniMath formalizations of other components of the proof but the proof in its totality will remain out of reach for constructive formalization.

Finally, there is the third direction that I think UniMath can and should develop. It is the direction towards the modern theory of geometry and topology of manifolds.

The first step in this direction can be a definition of a univalent category of smooth manifolds. Univalence will force all the constructions relying on this category to be invariant, or maybe better to say equivariant, with respect to diffeomorphisms.

No one knows how much of the theory of smooth manifolds can be developed constructively which creates an additional challenge.

At the end of my lecture I want to acknowledge the extremely important role that the Coq development team played in the recent years in the successes of the UniMath.

Coq today is faster and better than it was a few years ago and without this progress building UniMath would be much more difficult.

Thank you!