Models of type theory in univalent mathematics

B. Ahrens, about jww P. LeF. Lumsdaine, V. Voevodsky

Institut de Recherche en Informatique de Toulouse
Université Paul Sabatier

2015–06–30

Outline

❶ `UniMath`: a library of univalent mathematics

❷ Formalizing models of type theory in `UniMath`

Outline

1. `UniMath`: a library of univalent mathematics
2. Formalizing models of type theory in `UniMath`

What is `UniMath`?

- one of several libraries of univalent mathematics
- using the `Coq` proof assistant (following branch V8.5 atm)
- combines several libraries:
  - `Foundations` by Voevodsky
  - `RezkCompletion` by Ahrens, Kapulkin, Shulman
  - `Ktheory` by Grayson
  - (`PAdics` by Pelayo, Voevodsky, Warren)
- Base for several more libraries:
  - Work on substitution systems by Ahrens, Matthes
  - Formalization of cubical model by Mörtberg
  - Models of type theory by Ahrens, Lumsdaine, Voevodsky (see later)

What is `UniMath`?

- Since V8.5beta2: use of **vanilla `Coq`**, no patches necessary
- Crucial flags `-indices-matter`, `-type-in-type`
- General philosophy of `UniMath`: stay within MLTT fragment of CIC, for kernel:
  - no use of records
  - no use of type classes
  - no use of general inductive declarations given via `Inductive` scheme
- Univalence taken as axiom; no HITs

`https://github.com/UniMath/UniMath`

Constituent pieces I: `Foundations`

- Written by Voevodsky, 2009 – today
- approx. 6500loc (but very long ones), 820k chars

Contents

- basic (and less basic) HoTT stuff
- set quotients
- algebraic hierarchy: from monoids to fields
- naturals, integers, rationals

Constituent pieces II: `RezkCompletion`

- Written by Ahrens, Kapulkin, Shulman, 2012 – today
- approx. 6000loc, 240k chars

Contents

- (pre)categories, functors, natural transformations, adjunctions, equivalences
- Rezk completion: from precategories to categories
- some limits

Constituent pieces III: `Ktheory`

- Written by Grayson, 2013 – 2014
- approx. 5000loc, 260k chars

Contents

- groups by generators and relations, free groups
- abelian groups, group actions, torsors
- definition of $B(G)$ and its covering space $E(G)$, proof (using univalence) that the loop space of $B(G)$ is G
- construction of the circle as $B(\mathbb{Z})$

Constituent pieces IV: `PAdics`

- Written by Pelayo, Voevodsky, Warren, 2011 – 2012
- approx. 3000loc, 230k chars

Contents

- stuff about p-adic numbers?
- code not maintained, does not compile with current `Foundations`

POST-TALK EDIT: Warren is currently updating `PAdics` to the latest version of `UniMath`. For status info see `https://github.com/UniMath/UniMath`.

Outline

1. `UniMath`: a library of univalent mathematics

2. Formalizing models of type theory in `UniMath`

What is a type theory?

What is a type theory?

See Vladimir's talk.

What is a model of type theory?

- “Model”: algebraic structure intended for interpreting syntax
- Various notions of “model” considered in this talk model a skeletal type theory without type/term constructors.
- For now, model just type dependency and substitution.

Data modeled in such a model

- contexts and their morphisms
- types and terms in context
- substitution with respect to context morphisms

Notions of “model of type theory”

The zoo of “models of type theory”

- categories with families
- categories with attributes
- contextual categories
- comprehension categories
- type categories
- categories with display maps
- …

Notions of “model of type theory”

- In general, a model is a category with extra structure.
- The alternatives differ in how the various data are represented, **algebraically** or **categorically**

algebraically given by operations satisfying equations

categorically given as objects satisfying universal property

Notions of “model of type theory”

How do they relate to each other?

In classical set-theoretic foundations

For overview see `http://ncatlab.org/nlab/show/categorical+model+of+dependent+types`

In univalent foundations

Additional parameters:

- strong vs. weak existence
- two notions of “category” (details later)

entail further bifurcations of those notions

Goals

Goal of this project

- Vary some of these parameters and compare the resulting notions
- Formalize in `UniMath`

More specifically, comparing means:

1. construct functions between the various types of models
2. prove properties of maps: injectivity, equivalence, ...

Functions vs. functors

- in set theory **functors** are the only meaningful way to compare these notions (constructing adjunctions or similar): equality is too strict, injectivity of functions would not be meaningful

- univalent identity in type theory makes injectivity meaningful as a property of **functions** between the types of models

Interlude: (pre)categories in univalent mathematics

A preprecategory is

- a type $O : \mathcal{U}$ of objects
- a dependent type $A : O \times O \to \mathcal{U}$ of arrows
- $\mathsf{id} : \prod_{(a:O)} A(a, a)$
- $(\circ) : \prod_{(a,b,c:O)} A(a, b) \times A(b, c) \to A(a, c)$
- axioms postulating equalities of arrows

Interlude: (pre)categories in univalent mathematics

A precategory is

- a type $O : \mathcal{U}$ of objects
- a dependent type $A : O \times O \to \mathcal{S}et$ of arrows
- $\mathsf{id} : \prod_{(a:O)} A(a, a)$
- $(\circ) : \prod_{(a,b,c:O)} A(a, b) \times A(b, c) \to A(a, c)$
- axioms postulating equalities of arrows

Interlude: (pre)categories in univalent mathematics

A   category is

- a type $O : \mathcal{U}$ of objects
- a dependent type $A : O \times O \to \mathcal{S}et$ of arrows
- $\mathsf{id} : \prod_{(a:O)} A(a,a)$
- $(\circ) : \prod_{(a,b,c:O)} A(a,b) \times A(b,c) \to A(a,c)$
- axioms postulating equalities of arrows

such that

$$\mathsf{idtoiso} : \prod_{a,b:O} (a = b) \to \mathsf{iso}(a,b)$$

is an equivalence.

Examples of categories

Precategories that are categories:

- hSets
- Groups, rings, ... (Structure Identity Principle)
- Functor category $[C, D]$, if D is a category

Non-example:

$$\begin{tikzcd} \bullet \arrow[r, bend left] & \bullet \arrow[l, bend left] \end{tikzcd}$$

(indiscrete precategory on two objects)

Rezk completion: from precategories to categories

- Every category is a precategory
- Conversely, turn a precategory C into a category via "Rezk completion", a (homotopy) quotient of C

Intuition behind the Rezk completion

add as many identities between objects a and b as there are isomorphisms

Rezk completion and models of type theory

Reminder: notion of model is given by (pre)category with structure.

Interplay between Rezk completion and structure of model

1. Does a given structure on a precategory C induce a structure on its Rezk completion?
2. Does the map $structure_1 \to structure_2$ depend on the underlying precategory being a category?

Uniqueness of limits in categories

Lemma

In a category, limiting cones are unique up to propositional equality.

Put differently,

in a category, “specified pullbacks” is a property.

Notions of models considered

- Categories with Families
- Comprehension Categories, plus the "split" version
- Categories with Display Maps

A short overview…

Categories with Families

A precategory with families is a precategory $\mathcal{C}$ with

- for any $\Gamma : \mathcal{C}_0$, a set $\mathcal{C}(\Gamma)$;
- for any $\Gamma : \mathcal{C}_0$ and $A : \mathcal{C}(\Gamma)$, a set $\mathcal{C}(\Gamma \vdash A)$;
- for any $\gamma : \mathcal{C}(\Gamma', \Gamma)$, a *reindexing* function $\mathcal{C}(\Gamma) \to \mathcal{C}(\Gamma'), \;\; A \mapsto A[\gamma]$;
- for any $\gamma : \mathcal{C}(\Gamma', \Gamma)$ and $A : \mathcal{C}(\Gamma)$, a function $\mathcal{C}(\Gamma \vdash A) \to \mathcal{C}(\Gamma \vdash A[\gamma]), \;\; a \mapsto a[\gamma]$;
- for any $\Gamma : \mathcal{C}_0$ and $A : \mathcal{C}(\Gamma)$, an object $\Gamma.A$ and a *projection* morphism $\pi_A : \mathcal{C}(\Gamma.A, \Gamma)$;
- for any $\Gamma : \mathcal{C}_0$ and $A : \mathcal{C}(\Gamma)$, a *generic element* $\nu : \mathcal{C}(\Gamma.A \vdash A[\pi_A])$;
- *pairing*, corresponding to extension of context morphisms;
- laws …

Comprehension Categories

A comprehension precategory is a precategory $\mathcal{C}$ with

- for any object $\Gamma : \mathcal{C}_0$, a type $\mathcal{C}(\Gamma)$,
- for any $A : \mathcal{C}(\Gamma)$, an object $\Gamma.A : \mathcal{C}_0$,
- *projection* morphisms $\pi_{(\Gamma,A)} : \mathcal{C}(\Gamma.A, \Gamma)$,
- for any morphism $\gamma : \mathcal{C}(\Gamma', \Gamma)$, a *reindexing* function $\mathcal{C}(\Gamma) \to \mathcal{C}(\Gamma'), A \mapsto A[\gamma]$,
- for any $\gamma : \mathcal{C}(\Gamma', \Gamma)$ and $A : \mathcal{C}(\Gamma)$, a morphism $q_{(\gamma,A)} : \mathcal{C}(\Gamma'.A[\gamma], \Gamma.A)$,
- for any $\gamma : \mathcal{C}(\Gamma', \Gamma)$ and $A : \mathcal{C}(\Gamma)$,

$$\begin{tikzcd}
\Gamma'.A[\gamma] \arrow[r, "q_{(\gamma,A)}"] \arrow[d, "\pi_{(\Gamma',A[\gamma])}"'] & \Gamma.A \arrow[d, "\pi_{(\Gamma,A)}"] \\
\Gamma' \arrow[r, "\gamma"'] & \Gamma
\end{tikzcd}$$

- for any $\gamma : \mathcal{C}(\Gamma', \Gamma)$ and $A : \mathcal{C}(\Gamma)$, the above square is a pullback.

Split comprehension precategories

A comprehension category as above is *split* if

- $\mathcal{C}(\Gamma)$ is a set for each Γ,
- reindexing (of types) is functorial
- q is functorial

POST-TALK EDIT: what is called "comprehension category" here should really be called "type category" after A. Pitts, *Categorical Logic*, 2000, Def. 6.3.3. This has since been renamed in our development.

Categories with Display Maps

A precategory with display maps is given by a precategory $\mathcal{C}$ with

- for any $\Delta, \Gamma : \mathcal{C}_0$, a subtype $\mathsf{DM}_{\Delta,\Gamma} : \mathcal{C}(\Delta, \Gamma) \to \mathsf{Prop}$
- DM is closed under isomorphism (in the arrow precategory), and
- display maps have (specified) pullbacks along all maps; and they are again display maps.

Conjectural relation between models

$$\begin{tikzcd}
\mathsf{CwF} \arrow[r, "f"'] \arrow[rrr, bend left, "j"] \arrow[d, bend left, "I"] \arrow[d, phantom, "\dashv"] & \mathsf{splitCompC} \arrow[r, "g"'] \arrow[d, bend left, "I"] \arrow[d, phantom, "\dashv"] & \mathsf{CompC} \arrow[r, "h"'] \arrow[d] & \mathsf{CwDM} \arrow[l, bend left, "k"] \arrow[d, bend left, "I"] \arrow[d, phantom, "\dashv"] \\
\mathsf{PreCwF} \arrow[r, "f"'] \arrow[u, bend left, "R"] & \mathsf{splitCompPreC} \arrow[r, "g"'] \arrow[u, bend left, "R"] & \mathsf{CompPreC} & \mathsf{PreCwDM} \arrow[l, bend left, "k"] \arrow[u, bend left, "R"]
\end{tikzcd}$$

- Maps f, g, h, j, k do not change the underlying (pre)category
- g is injective (forgets splitness)
- $j = h \circ g \circ f$
- Conjecture: f is an equivalence
- Conjecture: left adjoints R to inclusions I exist

Current status of the project

Completed

- Construction of maps between different structures

Not completed

- Proofs of properties of constructed maps
- Compatibility of structures with Rezk completion

Details about the constructed maps

- All the maps constructed between different structures leave the underlying (pre)category unchanged
- Maps $\mathsf{CwF} \to \mathsf{CwDM}$ and $\mathsf{CompC} \to \mathsf{CwDM}$ use the fact that "specified pullbacks" is a property in categories

Details about the formalization

- 2500loc
- needs `-type-in-type`

Rewriting by hand:

- `rewrite lemma` mostly fails
- instead, use `etransitivity`; isolate subterm; `apply lemma`
- side effect: produces nice identity terms
- possible to automate (proof-relevant rewriting)?

Details about the formalization

- 2500loc
- needs `-type-in-type`

Rewriting by hand:

- `rewrite lemma` mostly fails
- instead, use `etransitivity`; isolate subterm; `apply lemma`
- side effect: produces nice identity terms
- possible to automate (proof-relevant rewriting)?

Thanks for your attention.