Displayed Universal Algebra in UniMath

Basic Definitions and Results

Gianluca Amato
Università "G. d'Annunzio" di Chieti-Pescara

Joint work with M. Calosci, M. Maggesi, C. Perini Brogi

ICTCS 2025
26th Italian Conference on Theoretical Computer Science
Pescara (IT), September 10–12, 2025

Goals of the project

- Formalize **Universal Algebra** in the **UniMath** proof assistant;
- Make all constructions evaluable as much as possible.

What is UniMath ?

Martin-Löf Type Theory

UniMath is a proof assisant for a variant of (intensional) Martin-Löf Type Theory.

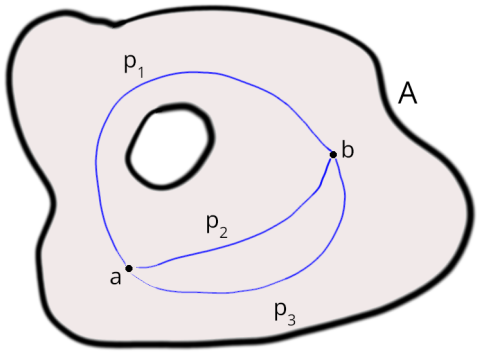
Martin-Löf Type Theory is a type theory for the foundation of mathematics:

- (strongly normalizing) λ-calculus with dependent types;
  - **non-dependent** functions of type $A \to B$;
  - **dependent** functions of type $\prod x : A, B(x)$;
- **propositions as types** interpretation of logic.
  - a **proposition** P is a type;
  - a **proof** of P is a term $t : P$.

Homotopy Type Theory

Homotopy Type Theory (HoTT) is a way to (informally) interpret type theory in which:

- types A are **topological spaces**;
- if $a, b : A$, the equality $a = b$ is the space of all **paths** from a to b;
- if $p_1, p_2 : a = b$ are two proofs for $a = b$, then $p_1 = p_2$ is the space of **homotopies** (continuous deformation of paths) from p_1 to p_2;
- there is no reason while $e : p_1 = p_2$ should exist in general.



UniMath

- UniMath is a proof assistant for HoTT with
  - **univalence axiom**;
  - **propositional resizing**;
  - no general inductive types (only `nat` is builtin).
- Contains a vast collection of formalized mathematics.
- It is not a standalone proof-assistant but an **alternative standard library** for Coq.

https://github.com/UniMathUA/UniMath

Back to the project

Motivation behind our project

Why we want to formalize Universal Algebra in UniMath ?

- a valuable addition to the UniMath library in itself;
- the machinery for formalizing terms may be reused to build **W-types**;
- we were curious to explore what HoTT could give us:
  - building quotients of algebras without using **setoids**.

What we have done

- Multi sorted **signature**;

Signature

```
∑ (S: decSet) (O: hSet), O → list S × S.
```

G. Amato, M. Calosci, M. Maggesi, C. Perini Brogi. **Universal Algebra in UniMath**. MSCS 2024

What we have done

- Multi sorted **signature**;
- **Algebra** over a signature:

Algebra over σ

```
∑ A: sUU (sorts σ), ∏ nm: names σ,
      A⋆ (arity nm) → A (sort nm).
```

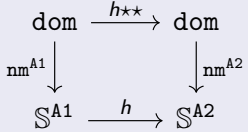
G. Amato, M. Calosci, M. Maggesi, C. Perini Brogi. **Universal Algebra in UniMath**. MSCS 2024

What we have done

- Multi sorted **signature**;
- **Algebra** over a signature:
- Homomorphisms;

Homomorphisms

An **homomorphism** between `A1` and `A2` is a sorts-indexed map `h : A1 s→ A2` such that



G. Amato, M. Calosci, M. Maggesi, C. Perini Brogi. **Universal Algebra in UniMath**. MSCS 2024

What we have done

- Multi sorted **signature**;
- **Algebra** over a signature:
- Homomorphisms;
- Term Algebra;

Terms

In the absence of tree, we represent terms using **reverse polish notation**.

A **stack based machine** determines whether a given sequence of symbols is a valid term or just garbage.

Induction principle for terms is derived.

G. Amato, M. Calosci, M. Maggesi, C. Perini Brogi. **Universal Algebra in UniMath**. MSCS 2024

What we have done

- Multi sorted **signature**;
- **Algebra** over a signature:
- Homomorphisms;
- Term Algebra;
- Equations.

Equations

An **equation** is just a pair of terms (with variables) of the same sort;

An equational specification is an indexed collection of equations;

An **equational algebra** is an algebra such that any equation (of a given equational specification) holds.

G. Amato, M. Calosci, M. Maggesi, C. Perini Brogi. **Universal Algebra in UniMath**. MSCS 2024

What we are working on now

Displayed Algebras

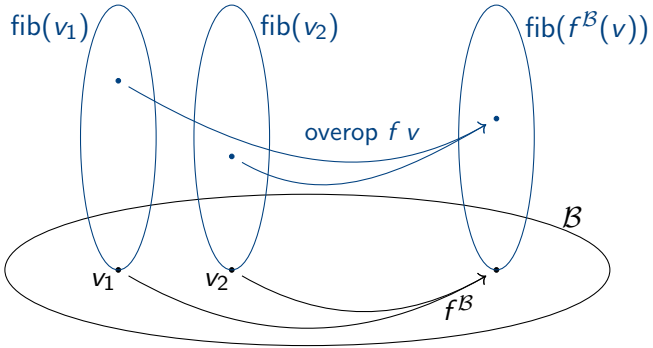
- Introduce a generic mechanism for building new algebras;
- Inspired by *displayed categories* in HoTT.

Code already **merged** in the standard UniMath distribution.

Displayed Algebra

A **displayed algebra** over `B : algebra` σ is given by:

- a family of **fiber types** indexed over terms of `B`
- a family of **"over operations"** functions indexed over any operation name `nm` and any vector `v` of terms of sort specified by the arity of `nm`.



The data of a displayed algebra yelds a **total algebra** with the same signature σ

```
total_alg {A: algebra σ} (D: disp_alg A): algebra σ.
```

together with a **forgetful homomorphism** from the total algebra to `A`.

On the other hand, from any homomorphism `h: hom A B` of algebras over σ, one gets the **displayed algebra** over B **of its fibers**:

- The fiber types are the fiber under `h` of the specific index term;
- The over operations is given by `h`. They are well typed because `h` is an homomorphism.

Use Cases

Future works we aim to develop featuring displayed algebras include

- **Subalgebra**;
- **Cartesian Product** and **Pullbacks**;
- **Semidirect product** of groups;
- relations with **quotients** and **homomorphism theorems**;
- results about **composition of displayed constructions**.

Future Work

- Prove theorems!
  - Initial algebra of terms modulo equational congruence;
  - Birkhoff's variety theorem;
  - Generalise the relation between our term algebras and homotopy W-types.
- Technicalities: Streamline the interface;
  - Eg: ground term algebra should be a special case of free algebra;
  - readdress heterogeneous vectors.
- More applications and examples of univalent reasoning.

Thanks for your attention!

Equality in Martin-Löf Type Theory

A proposition may have different proofs. Example: $\sum(x : \mathbb{N}), 2 * x < 3$ has

- $(0, a\ \textit{proof that}\ 0 < 3)$;
- $(1, a\ \textit{proof that}\ 2 < 3)$;

Equality in Martin-Löf Type Theory

A proposition may have different proofs. Example: $\sum(x : \mathbb{N}), 2 * x < 3$ has

- $(0, a\ \textit{proof that}\ 0 < 3)$;
- $(1, a\ \textit{proof that}\ 2 < 3)$;

Equality $a = b$ is a proposition.

If $p_1, p_2 : a = b$ are two proofs for $a = b$, what about $p_1 = p_2$?

Equality in Martin-Löf Type Theory

A proposition may have different proofs. Example: $\sum(x : \mathbb{N}), 2 * x < 3$ has

- $(0, a\ \textit{proof that}\ 0 < 3)$;
- $(1, a\ \textit{proof that}\ 2 < 3)$;

Equality $a = b$ is a proposition.

If $p_1, p_2 : a = b$ are two proofs for $a = b$, what about $p_1 = p_2$?

If we think to types as sets, equality means that a and b are exactly the same object. It seems reasonable that there is a single proof of $a = b$.

Axiom UIP (Uniqueness of Identity Proofs)

All terms of the identity types are equals.

Let `B` be **set-supported**. The **displayed algebra of the fibers** and the **forgetful morphism** of a displayed algebras are inverse to each other:

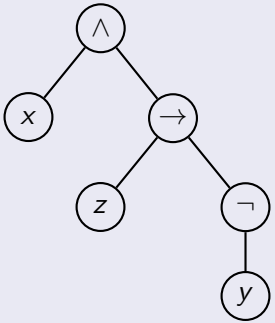
$$\sum \texttt{(A:algebra } \sigma\texttt{), hom A B} \quad \simeq \quad \texttt{(disp\_alg B)}$$

Terms have a tree-like structure

Ground Term Algebra

Let's take a signature σ with a single sort $\mathbb{S}$, constants x, y, z, a unary operation $\neg$, and two binary operation $\wedge$ and $\rightarrow$.

Example: the term $x \wedge (z \rightarrow \neg y)$



How to formalize the type of these structures?

Our formalization

Ground Term Algebra

σ has a single sort $\mathbb{S}$, and operation names x, y, z, $\neg$, $\wedge$ and $\rightarrow$.

We use **lists**, **stacks** and **monads**.

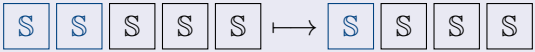
- Polish notation: we express $x \wedge (z \rightarrow \neg y)$ as

$$\boxed{\wedge}\ \boxed{x}\ \boxed{\rightarrow}\ \boxed{z}\ \boxed{\neg}\ \boxed{y} \quad : \texttt{list (opnames } \sigma\texttt{)}$$

- We need a proposition to identify which lists of operation represent a term.

$$\texttt{opexec}\colon \texttt{opnames}\ \sigma \to \texttt{sortstack}\ \sigma \to \texttt{sortstack}\ \sigma$$

Example: opexec $\wedge$



- We are able to "catch" and propagate errors:

$$\texttt{opexec}\ \wedge\ \boxed{\mathbb{S}} \equiv \times$$

$$\texttt{opexec}\ \wedge\ \times \equiv \times$$

$$\texttt{opexec}\ \wedge\ \boxed{\mathbb{T}}\,\boxed{\mathbb{T}} \equiv \times$$

List Operation execution

Ground Term Algebra

$$\texttt{oplistexec}\colon \texttt{oplist}\ \sigma \to \texttt{sortstack}\ \sigma$$

$$\texttt{oplistexec}\ \emptyset :\equiv \emptyset$$

$$\texttt{oplistexec}\ \boxed{\texttt{nm}}\boxed{\text{rest of the list}} :\equiv \texttt{opexec nm (oplistexec}\ \boxed{\text{rest of the list}}\ \texttt{)}$$

List Operation execution: Examples

Ground Term Algebra

Let's calculate `oplistexec` $\boxed{\wedge}\ \boxed{x}\ \boxed{\to}\ \boxed{z}\ \boxed{\neg}\ \boxed{y}$.

$$\emptyset \overset{y}{\mapsto} \boxed{\mathbb{S}} \overset{\neg}{\mapsto} \boxed{\mathbb{S}} \overset{z}{\mapsto} \boxed{\mathbb{S}}\,\boxed{\mathbb{S}} \overset{\to}{\mapsto} \boxed{\mathbb{S}} \overset{x}{\mapsto} \boxed{\mathbb{S}}\,\boxed{\mathbb{S}} \overset{\wedge}{\mapsto} \boxed{\mathbb{S}}$$

- We are still able to "catch" errors: `oplistexec` $\boxed{\wedge}\ \boxed{x}\ \boxed{\to}\ \boxed{\neg}\ \boxed{y}$

$$\emptyset \overset{y}{\mapsto} \boxed{\mathbb{S}} \overset{\neg}{\mapsto} \boxed{\mathbb{S}} \overset{\to}{\mapsto} \times \overset{x}{\mapsto} \times \overset{\wedge}{\mapsto} \times$$

- A list of operations which does not return an error state does not necessarily represent a valid term.

Definition: Term

A Term of sort $\mathbb{S}$ is a list of operations that, when "executed" with `oplistexec`, returns the stack $\boxed{\mathbb{S}}$.

$$\texttt{term}\ \sigma\ \ \mathbb{S}\ \equiv \sum_{(l:\texttt{oplist}\ \sigma)} \texttt{oplistexec}\ l = \boxed{\mathbb{S}}$$

- It is a subtype of `oplist` σ.

- `build_term` (**introduction** principle): We can introduce a new term from an operation name and a vector of terms with appropriate sorts.

- `build_term` (**introduction** principle): We can introduce a new term from an operation name and a vector of terms with appropriate sorts.
- `term_ind` (**induction** principle): Given a predicate P on terms satisfying the opportune inductive hypothesis, we have that P holds for any term.

- `build_term` (**introduction** principle): We can introduce a new term from an operation name and a vector of terms with appropriate sorts.
- `term_ind` (**induction** principle): Given a predicate P on terms satisfying the opportune inductive hypothesis, we have that P holds for any term.
- `term_ind_step` (**computation** path):

```
term_ind P R (build_term nm v)
= R nm v (h2map (λ s t q, term_ind P R t) (h1lift v))
```

Use cases for induction on terms

Ground Term Algebra

- Definition of the ground term algebra;
- `depth` and `fromterm` functions:

```
fromterm: term σ s → A s
```

- Terms with variables: definition of free algebras;

Use cases for induction on terms

Ground Term Algebra

- Definition of the ground term algebra;
- `depth` and `fromterm` functions:

```
fromterm: term σ s → A s
```

- Terms with variables: definition of free algebras;
- Relation between terms and **(homotopy) W-types**.

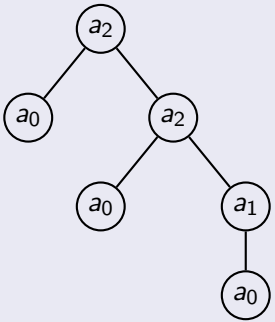
Relation between terms and homotopy W-types.

Let `A: 𝒰` and `B: A → 𝒰`. `W A B` is the inductive type with one constructor

$$\texttt{sup}\colon \prod \texttt{(x:A), (B(x) → W A B) → W A B}.$$

- It is useful to think of its terms as trees.

Example



Basic for equations

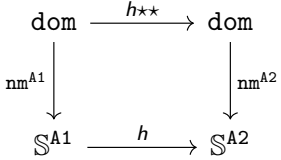
- An `equation` is just a pair of terms (with variables) of the same sort;
- An equational specification is an indexed collection of equations;
- We have a predicate

```
holds (a: algebra σ) (e: equation σ V) : UU
  := ∏ α, fromterm (ops a) α (eqsort e) (lhs e)
          = fromterm (ops a) α (eqsort e) (rhs e).
```

- An **equational algebra** is an algebra such that any equation (of a given equational specification) holds.

Categorical Structures

- An **homomorphism** between `A1` and `A2` is a sorts-indexed map `h : A1 s→ A2` such that



  - Identities and composition of homomorphism are homomorphism.
  - If the support types are sets then homomorphisms constitute a set.
- We have a **univalent** category of Algebras.

Example

Dummett's tautology:

Consider a signature for the algebra of booleans. We have

- The free term algebra (with variables `x` and `y`);
- The boolean algebra built from the type `bool` of UniMath;

```
Lemma Dummett : ∏ i, interp i (disj (impl x y) (impl y x)) = true.
Proof.
  intro i. lazy.
  induction (i 0); induction (i 1); apply idpath.
Qed.
```

- The evaluation is done by the computing mechanism of Coq.

Subalgebra as a Displayed Algebra

Examples

There are two natural ways to formalize the notion of an algebra `A`

- as an **embedding** targetting `A`. That is an homomorphism `i: hom B A` which is injective on any support;
- as a **subuniverse** of `A`. That is a collection of support subtypes closed wrt the operations. That is a displayed algebra over `A` in which **any fiber type is a proposition**

Assuming `A` to be set-supported, the previous equivalence is specialized to

$$\sum \texttt{(B: algebra } \sigma\texttt{), embedding B A} \simeq$$
$$\sum \texttt{(PB: shsubtype A), issubuniverse A PB.}$$