Introducing Displayed Universal Algebra in UniMath

Calosci Matteo

Joint work with: G. Amato, M. Maggesi, C. Perini Brogi

15/04/2025 HoTT/UF 2025

UNIVERSITÀ DEGLI STUDI FIRENZE

Da un secolo, oltre.

UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

Goals of the project

- Formalize a pre-categorical presentation of Universal Algebra;
- Evaluable Constructions.

Code available at **github.com/UniMath/UniMath** and at **github.com/UniMathUA/UniMath**

Univalent Mathematics

What it is and why we choose it

Martin-Löf Type Theory

Basic inductive types:

$$\prod \qquad \sum \qquad + \qquad \mathbf{0} \qquad \mathbf{1} \qquad \mathbb{N} \qquad =$$

Two kinds of equality: Judgmental and Propositional.

Univalence Axiom

$$A \simeq B \quad \simeq \quad A = B$$

Reasoning up to isomorphism.

UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

Coq/UniMath

- Minimalist implementation of Univalent Mathematics:
  - General inductive definitions are avoided.
- Contains vast collection of formalized results:
  - Ready-made structures as target for applicartions;
  - Categorical notions;
  - Algebraic Theories.

Universal Algebra

What we have formalized

- Multi sorted **signature**;

Signature

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

UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

Universal Algebra

What we have formalized

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

Algebra over σ

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

Universal Algebra

What we have formalized

- Multi sorted **signature**;
- **Algebra** over a signature:
  - Unit Algebra;
  - Term Algebra;
  - Free Algebra;

Ground Terms

We use **lists**, **stack** and **monad**.

An operation symbols **acts** on a stack of sorts by checking that the arity sorts are at the top of the stack and replacing them with the output sort.

Terms are those list of operation names (prefix notation) which act on the empty stack by returning the expected output sort.

- Principles for terms are derived.
- This construction can be encoded as an homotopy W-type

Universal Algebra

What we have formalized

- Multi sorted **signature**;
- **Algebra** over a signature:
  - Unit Algebra;
  - Term Algebra;
  - Free Algebra;
  - Homomorphisms.
- Categorical Structures;

Homomorphisms

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

$$\begin{tikzcd} \mathtt{dom} \arrow[r, "h\star\star"] \arrow[d, "\mathtt{nm}^{\mathtt{A1}}"'] & \mathtt{dom} \arrow[d, "\mathtt{nm}^{\mathtt{A2}}"] \\ \mathbb{S}^{\mathtt{A1}} \arrow[r, "h"] & \mathbb{S}^{\mathtt{A2}} \end{tikzcd}$$

- 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.

Universal Algebra

What we have formalized

- Multi sorted **signature**;
- **Algebra** over a signature:
  - Unit Algebra;
  - Term Algebra;
  - Free Algebra;
  - Homomorphisms.
- Categorical Structures;
- Basics for equations;
- Examples;

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.

Universal Algebra

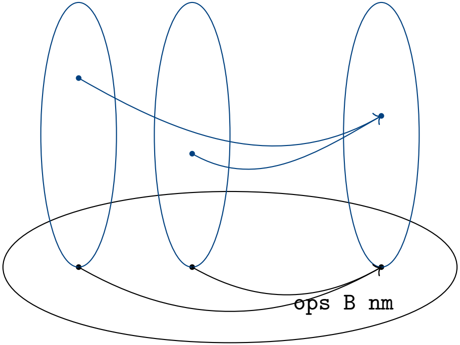
What we have formalized

- Multi sorted **signature**;
- **Algebra** over a signature:
  - Unit Algebra;
  - Term Algebra;
  - Free Algebra;
  - Homomorphisms.
- Categorical Structures;
- Basics for equations;
- Examples;
- **Displayed Algebras**.

Displayed Algebra

A **displayed algebra** over `B :algebra` σ is the data of

- 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`.



Total Algebras and Forgetful Morphism

Displayed Algebras

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

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

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

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.

Morphisms and Displayed Algebras

Displayed Algebras

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

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

Example: Subalgebra

There are two natural ways to formalize the notion of a subalgebra of `B`

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

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

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

Use Cases

Future works we aim to develop featuring displayed algebras include

- **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.

Thank you for listening!
Any questions?

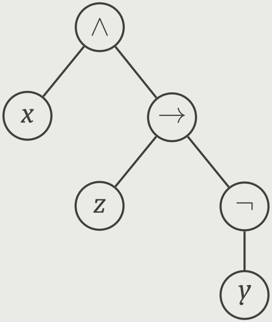
UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

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 $\to$.

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



How to formalize the type of these structures?

UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

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**, **stack** and **monad**.

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

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

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

Operation execution

Ground Term Algebra

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

Example: $\texttt{opexec}\quad \wedge$

$$\boxed{\mathbb{S}}\ \boxed{\mathbb{S}}\ \boxed{\mathbb{S}}\ \boxed{\mathbb{S}}\ \boxed{\mathbb{S}} \longmapsto \boxed{\mathbb{S}}\ \boxed{\mathbb{S}}\ \boxed{\mathbb{S}}\ \boxed{\mathbb{S}}$$

- 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

$$\texttt{oplistexec}\textcolor{orange}{:}\ \texttt{oplist}\ \sigma \rightarrow \texttt{sortstack}\ \sigma$$

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

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

UNIVERSITÀ DEGLI STUDI FIRENZE
Da un secolo, oltre.

List Operation execution: Examples

Ground Term Algebra

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

$$\emptyset \overset{y}{\longmapsto} \boxed{\mathbb{S}} \overset{\neg}{\longmapsto} \boxed{\mathbb{S}} \overset{z}{\longmapsto} \boxed{\mathbb{S}}\ \boxed{\mathbb{S}} \overset{\rightarrow}{\longmapsto} \boxed{\mathbb{S}} \overset{x}{\longmapsto} \boxed{\mathbb{S}}\ \boxed{\mathbb{S}} \overset{\wedge}{\longmapsto} \boxed{\mathbb{S}}$$

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

$$\emptyset \overset{y}{\longmapsto} \boxed{\mathbb{S}} \overset{\neg}{\longmapsto} \boxed{\mathbb{S}} \overset{\rightarrow}{\longmapsto} \times \overset{x}{\longmapsto} \times \overset{\wedge}{\longmapsto} \times$$

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

Terms

Ground Term Algebra

Definition: Term

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

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

- It is a subtype of `oplist` σ.

Principles for Terms

Ground Term Algebra

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

Principles for Terms

Ground Term Algebra

- `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.

Principles for Terms

Ground Term Algebra

- `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:

$$\texttt{fromterm}:\ \texttt{term}\ \sigma\ \texttt{s} \rightarrow \texttt{A}\ \texttt{s}$$

- 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**.

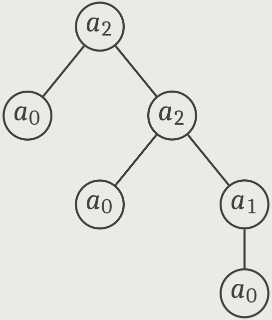
Relation between terms and W types.

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

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

- 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

$$\begin{tikzcd} \mathtt{dom} \arrow[r, "h\star\star"] \arrow[d, "\mathtt{nm}^{\mathtt{A1}}"'] & \mathtt{dom} \arrow[d, "\mathtt{nm}^{\mathtt{A2}}"] \\ \mathbb{S}^{\mathtt{A1}} \arrow[r, "h"] & \mathbb{S}^{\mathtt{A2}} \end{tikzcd}$$

  - 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.