Universal Algebra in UniMath

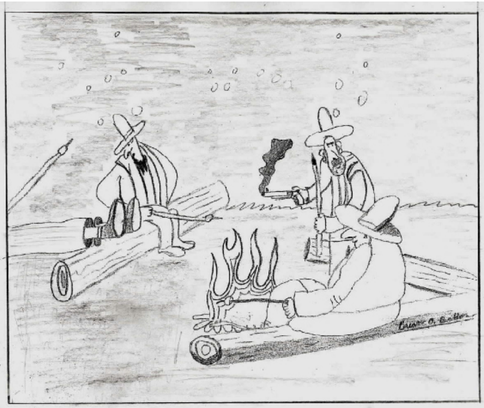
Cosimo Perini Brogi
(j.w.w. G. Amato, M. Maggesi, M. Parton)

University of Genoa

HoTT-UF 2020 – July 7, 2020

Univalent Foundations

Foundations of mathematics:

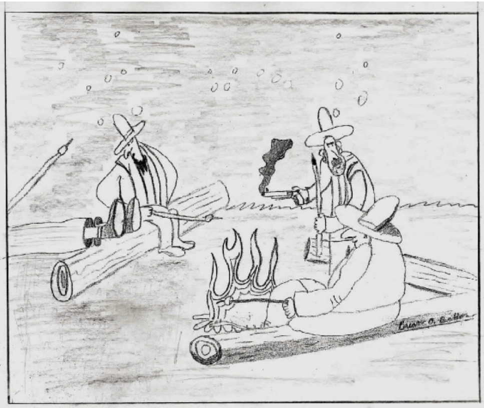


You guys are both my witnesses... He insinuated that ZFC set theory is superior to Type Theory!

- Syntax for mathematical objects
- Logic (i.e. notions of proposition and proof)
- Interpretation of the syntax in the context of mathematical objects

Univalent Foundations

Foundations of mathematics:

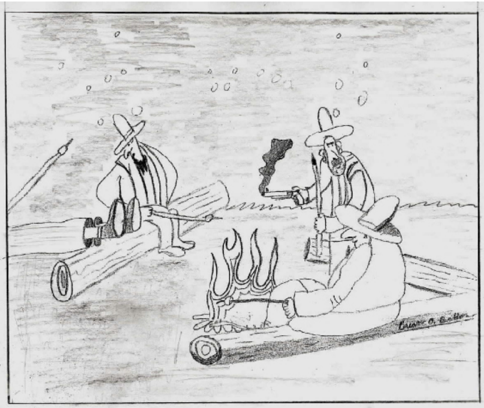


You guys are both my witnesses... He insinuated that ZFC set theory is superior to Type Theory!

- Syntax for mathematical objects
- Logic (i.e. notions of proposition and proof)
- Interpretation of the syntax in the context of mathematical objects

Univalent Foundations

Foundations of mathematics:



You guys are both my witnesses... He insinuated that ZFC set theory is superior to Type Theory!

- Syntax for mathematical objects
- Logic (i.e. notions of proposition and proof)
- Interpretation of the syntax in the context of mathematical objects

Univalent Foundations

Foundations of mathematics:

You guys are both my witnesses... He insinuated that
ZFC set theory is superior to Type Theory!

- Syntax for mathematical objects
- Logic (i.e. notions of proposition and proof)
- Interpretation of the syntax in the context of mathematical objects

UniMath – Origins

UniMath origin dates back to 2014 when three Coq libraries were combined:

- Foundations (Voevodsky, 2010)
- RezkCompletion (Ahrens, Kapulkin, Shulman, 2013)
- Ktheory (Grayson, 2013)

UniMath – Underlying Language

Martin-Löf Type Theory / subsystem of Coq:

- no `record` types
- no `inductive` types
- no `match` construct

Extended by:

- Univalence (and Function Extensionality) Axiom(s)
- Propositional Resizing

Warning

$\mathcal{U}_0 : \mathcal{U}_0$

UniMath – Underlying Language

Martin-Löf Type Theory / subsystem of Coq:

- no `record` types
- no `inductive` types
- no `match` construct

Extended by:

- Univalence (and Function Extensionality) Axiom(s)
- Propositional Resizing

Warning

$\mathcal{U}_0 : \mathcal{U}_0$

UniMath – Underlying Language

Martin-Löf Type Theory / subsystem of Coq:

- no `record` types
- no `inductive` types
- no `match` construct

Extended by:

- Univalence (and Function Extensionality) Axiom(s)
- Propositional Resizing

Warning

$\mathcal{U}_0 : \mathcal{U}_0$

Algebras and Signatures

A (single-sorted) signature σ consists of an *set* of *symbols* each one having a specific *arity*.

An algebraic structure over σ is given by an *set* A together with a *collection* of *operations* on A, corresponding to symbols of σ.

```
Definition Arity: UU := nat.

Definition signature: UU :=
 ∑ (names: hSet), names → Arity.

Definition names (σ: signature): hSet :=
 pr1 σ.

Definition arity {σ: signature}
  (nm: names σ): Arity :=
 pr2 σ nm.

Definition dom {σ: signature}
  (support: UU) (nm: names σ): UU :=
   Vector support (arity nm).

Definition cod {σ: signature}
  (support: UU) (nm: names σ): UU :=
   support.

Definition algebra (σ: signature): UU :=
   ∑ (support: hSet), ∏ (nm: names σ),
    dom support nm → cod support nm.
```

General Idea

$$\frac{\text{Signature}}{\text{Syntax}} \quad \text{and} \quad \frac{\text{Algebra}}{\text{Semantics}}$$

Algebras and Signatures

A (single-sorted) signature σ consists of an *set* of *symbols* each one having a specific *arity*.

An algebraic structure over σ is given by an *set* A together with a *collection* of *operations* on A, corresponding to symbols of σ.

```
Definition Arity: UU := nat.

Definition signature: UU :=
 ∑ (names: hSet), names → Arity.

Definition names (σ: signature): hSet :=
 pr1 σ.

Definition arity {σ: signature}
  (nm: names σ): Arity :=
 pr2 σ nm.

Definition dom {σ: signature}
   (support: UU) (nm: names σ): UU :=
    Vector support (arity nm).

Definition cod {σ: signature}
   (support: UU) (nm: names σ): UU :=
    support.

Definition algebra (σ: signature): UU :=
   ∑ (support: hSet), ∏ (nm: names σ),
    dom support nm → cod support nm.
```

General Idea

$$\frac{\text{Signature}}{\text{Syntax}} \quad \text{and} \quad \frac{\text{Algebra}}{\text{Semantics}}$$

Algebras and Signatures

A (single-sorted) signature σ consists of an *set* of *symbols* each one having a specific *arity*.

An algebraic structure over σ is given by an *set* A together with a *collection* of *operations* on A, corresponding to symbols of σ.

```
Definition Arity: UU := nat.

Definition signature: UU :=
 ∑ (names: hSet), names → Arity.

Definition names (σ: signature): hSet :=
 pr1 σ.

Definition arity {σ: signature}
  (nm: names σ): Arity :=
 pr2 σ nm.

Definition dom {σ: signature}
  (support: UU) (nm: names σ): UU :=
   Vector support (arity nm).

Definition cod {σ: signature}
  (support: UU) (nm: names σ): UU :=
   support.

Definition algebra (σ: signature): UU :=
   ∑ (support: hSet), ∏ (nm: names σ),
    dom support nm → cod support nm.
```

General Idea

$$\frac{\text{Signature}}{\text{Syntax}} \quad \text{and} \quad \frac{\text{Algebra}}{\text{Semantics}}$$

Algebras and Signatures

A (single-sorted) signature σ consists of an *set* of *symbols* each one having a specific *arity*.

An algebraic structure over σ is given by an *set* A together with a *collection* of *operations* on A, corresponding to symbols of σ.

```
Definition Arity: UU := nat.

Definition signature: UU :=
 ∑ (names: hSet), names → Arity.

Definition names (σ: signature): hSet :=
 pr1 σ.

Definition arity {σ: signature}
  (nm: names σ): Arity :=
 pr2 σ nm.

Definition dom {σ: signature}
  (support: UU) (nm: names σ): UU :=
   Vector support (arity nm).

Definition cod {σ: signature}
  (support: UU) (nm: names σ): UU :=
   support.

Definition algebra (σ: signature): UU :=
   ∑ (support: hSet), ∏ (nm: names σ),
    dom support nm → cod support nm.
```

General Idea

$$\frac{\text{Signature}}{\text{Syntax}} \quad \text{and} \quad \frac{\text{Algebra}}{\text{Semantics}}$$

Algebras and Signatures

A (single-sorted) signature σ consists of an *set* of *symbols* each one having a specific *arity*.

An algebraic structure over σ is given by an *set* A together with a *collection* of *operations* on A, corresponding to symbols of σ.

```
Definition Arity: UU := nat.

Definition signature: UU :=
 ∑ (names: hSet), names → Arity.

Definition names (σ: signature): hSet :=
 pr1 σ.

Definition arity {σ: signature}
  (nm: names σ): Arity :=
 pr2 σ nm.

Definition dom {σ: signature}
  (support: UU) (nm: names σ): UU :=
   Vector support (arity nm).

Definition cod {σ: signature}
  (support: UU) (nm: names σ): UU :=
   support.

Definition algebra (σ: signature): UU :=
   ∑ (support: hSet), ∏ (nm: names σ),
    dom support nm → cod support nm.
```

General Idea

$$\frac{\text{Signature}}{\text{Syntax}} \quad \text{and} \quad \frac{\text{Algebra}}{\text{Semantics}}$$

Univalent Category of σ-Algebras

A homomorphism between algebras over the same σ is a map of underlying carriers which respects operations.

```
Definition ishom {a1 a2: algebra σ}
  (f: a1 → a2): UU :=
   ∏ (nm: names σ) (x: dom a1 nm),
f (op a1 nm x) = (op a2 nm (vector_map f x)).
```

Lemma

σ-algebras and homomorphisms form a *univalent* category

Proof.

We use structure identity principle as implemented by displayed categories:

- $(A : \mathrm{hSet}) \mapsto (\mathrm{isAlgebra}\ A) : \mathcal{U}$
- $(f : A \to B) \mapsto (\texttt{ishom}\ f) : \mathrm{hProp}$
- `ishomid` and `ishomcomp`
- `is_univalent_disp_from_SIP_data`

See `CatAlgebras.v`. □

Univalent Category of σ-Algebras

A homomorphism between algebras over the same σ is a map of underlying carriers which respects operations.

```
Definition ishom {a1 a2: algebra σ}
  (f: a1 → a2): UU :=
   ∏ (nm: names σ) (x: dom a1 nm),
f (op a1 nm x) = (op a2 nm (vector_map f x)).
```

Lemma

σ-algebras and homomorphisms form a *univalent* category

Proof.

We use structure identity principle as implemented by displayed categories:

- $(A : \mathrm{hSet}) \mapsto (\mathrm{isAlgebra}\ A) : \mathcal{U}$
- $(f : A \to B) \mapsto$ (`ishom` f) : hProp
- `ishomid` and `ishomcomp`
- `is_univalent_disp_from_SIP_data`

See `CatAlgebras.v`. □

Univalent Category of σ-Algebras

A homomorphism between algebras over the same σ is a map of underlying carriers which respects operations.

```
Definition ishom {a1 a2: algebra σ}
  (f: a1 → a2): UU :=
   ∏ (nm: names σ) (x: dom a1 nm),
f (op a1 nm x) = (op a2 nm (vector_map f x)).
```

Lemma

σ-algebras and homomorphisms form a *univalent* category

Proof.

We use structure identity principle as implemented by displayed categories:

- $(A : \mathrm{hSet}) \mapsto (\mathrm{isAlgebra}\ A) : \mathcal{U}$
- $(f : A \to B) \mapsto (\texttt{ishom}\ f) : \mathrm{hProp}$
- `ishomid` and `ishomcomp`
- `is_univalent_disp_from_SIP_data`

See `CatAlgebras.v`. $\square$

Term Algebra

Let σ be a signature and V a set of variables (disjoint from σ). The term algebra $T(\sigma, V)$ has

- as carrier, the least set including V and closed under application of symbols of σ
- as operations, the "symbols themselves"

Lemma (`iscontrhomsfromterm`)

Given a signature σ, $T(\sigma, \emptyset)$ is the initial object of σ-algebras.

Lemma (`iscontruniversalmap`)

For any signature σ, any set V of variables, any σ-algebra A, and any $\alpha : V \to |A|$, there exists a unique homomorphism $\alpha^* : T(\sigma, V) \to A$ that extends α.

We want UniMath to evaluate and handle easily the terms, but we *do not* have inductive types!

Term Algebra

Let σ be a signature and V a set of variables (disjoint from σ). The term algebra $T(\sigma, V)$ has

- as carrier, the least set including V and closed under application of symbols of σ
- as operations, the "symbols themselves"

Lemma (`iscontrhomsfromterm`)

Given a signature σ, $T(\sigma, \varnothing)$ is the initial object of σ-algebras.

Lemma (`iscontruniversalmap`)

For any signature σ, any set V of variables, any σ-algebra A, and any $\alpha : V \to |A|$, there exists a unique homomorphism $\alpha^* : T(\sigma, V) \to A$ that extends α.

We want UniMath to evaluate and handle easily the terms, but we *do not* have inductive types!

Terms Implementation

Sketch of implementation

- $t \in T(sigma, V)$ $\rightsquigarrow$ list of function symbols (and variables)
- Lists are executed by a stack machine (`status` monad on natural numbers)
  - Status `n` $\rightsquigarrow$ remaining elements after execution
  - Status `error` $\rightsquigarrow$ stack underflow
- At the end of execution, a w.f. term always has status `1`
- Induction principle in order to reason on terms

```
Theorem term_ind (P: term sigma → UU)
  (R: term_ind_HP P) (t: term sigma): P t.
```

`R` states that for any symbol `nm` of σ, if `P` holds for any elements of the list corresponding to `nm`, then `P` holds for the whole term.

$\rightsquigarrow$ Key ingredients of the implementation, but several lemmas required to make it work!

(See `Terms.v` for the details)

Terms Implementation

Sketch of implementation

- $t \in T(sigma, V)$ $\leadsto$ list of function symbols (and variables)
- Lists are executed by a stack machine (`status` monad on natural numbers)
  - Status `n` $\leadsto$ remaining elements after execution
  - Status `error` $\leadsto$ stack underflow
- At the end of execution, a w.f. term always has status `1`
- Induction principle in order to reason on terms

```
Theorem term_ind (P: term sigma → UU)
  (R: term_ind_HP P) (t: term sigma): P t.
```

`R` states that for any symbol `nm` of σ, if `P` holds for any elements of the list corresponding to `nm`, then `P` holds for the whole term.

$\leadsto$ Key ingredients of the implementation, but several lemmas required to make it work!

(See `Terms.v` for the details)

Terms Implementation

Sketch of implementation

- $t \in T(sigma, V) \quad \rightsquigarrow \quad$ list of function symbols (and variables)
- Lists are executed by a stack machine (`status` monad on natural numbers)
  - Status `n` $\quad \rightsquigarrow \quad$ remaining elements after execution
  - Status `error` $\quad \rightsquigarrow \quad$ stack underflow
- At the end of execution, a w.f. term always has status `1`
- Induction principle in order to reason on terms

```
Theorem term_ind (P: term sigma → UU)
  (R: term_ind_HP P) (t: term sigma): P t.
```

`R` states that for any symbol `nm` of σ, if `P` holds for any elements of the list corresponding to `nm`, then `P` holds for the whole term.

$\rightsquigarrow$ Key ingredients of the implementation, but several lemmas required to make it work!

(See `Terms.v` for the details)

Terms Implementation

Sketch of implementation

- $t \in T(sigma, V)$ $\rightsquigarrow$ list of function symbols (and variables)
- Lists are executed by a stack machine (`status` monad on natural numbers)
  - Status `n` $\rightsquigarrow$ remaining elements after execution
  - Status `error` $\rightsquigarrow$ stack underflow
- At the end of execution, a w.f. term always has status `1`
- Induction principle in order to reason on terms

```
Theorem term_ind (P: term sigma → UU)
  (R: term_ind_HP P) (t: term sigma): P t.
```

`R` states that for any symbol `nm` of σ, if `P` holds for any elements of the list corresponding to `nm`, then `P` holds for the whole term.

$\rightsquigarrow$ Key ingredients of the implementation, but several lemmas required to make it work!

(See `Terms.v` for the details)

Terms Implementation

Sketch of implementation

- $t \in T(sigma, V) \quad \leadsto \quad$ list of function symbols (and variables)
- Lists are executed by a stack machine (`status` monad on natural numbers)
  - Status `n` $\leadsto$ remaining elements after execution
  - Status `error` $\leadsto$ stack underflow
- At the end of execution, a w.f. term always has status `1`
- Induction principle in order to reason on terms

```
Theorem term_ind (P: term sigma → UU)
  (R: term_ind_HP P) (t: term sigma): P t.
```

`R` states that for any symbol `nm` of σ, if `P` holds for any elements of the list corresponding to `nm`, then `P` holds for the whole term.

$\leadsto$ Key ingredients of the implementation, but several lemmas required to make it work!

(See `Terms.v` for the details)

Varieties

Given a signature σ, an equation has the form $t = s$, for $t, s \in T(\sigma, V)$.

```
Definition equation : UU :=
vterm σ V × vterm σ V.

Definition sysequations: UU
  := ∑ E : hSet, E → equation σ V.
```

```
Definition fromvterm {A:UU}
 (R : (∏ (nm : names σ),
    Vector A (arity nm)→A))(α:V → A)
    : vterm σ V → A.
```

We say that a σ-algebra A satisfies an equation $t = s$ when for any valuation $\alpha : V \to |A|$, $t[\alpha] = s[\alpha]$ holds in A.

```
Definition veval(A:algebra σ)
(α:V→support a):free_algebra σ V
→ support A :=
fromvterm(λ nm rec, op A nm rec) f.

Definition holds (A:algebra σ)
(e:equation σ V) : UU := ∏ α,
veval A α (lhs e) = veval A α(rhs e).

Definition eqsignature : UU
:= ∑ (σ : signature)(V: hSet),
sysequations σ V.
```

Univalent category of varieties (via displayed categories):

```
Definition is_eqalgebra {σ : eqsignature}(A : algebra σ) : UU
  := ∏ e : eqs σ, holds A (geteq e)
```

Varieties

Given a signature σ, an equation has the form $t = s$, for $t, s \in T(\sigma, V)$.

```
Definition equation : UU :=
vterm σ V × vterm σ V.

Definition sysequations: UU
  := ∑ E : hSet, E → equation σ V.

Definition fromvterm {A:UU}
 (R : (∏ (nm : names σ),
    Vector A (arity nm)→A))(α:V → A)
    : vterm σ V → A.
```

We say that a σ-algebra A satisfies an equation $t = s$ when for any valuation $\alpha : V \to |A|$, $t[\alpha] = s[\alpha]$ holds in A.

```
Definition veval(A:algebra σ)
(α:V→support a):free_algebra σ V
→ support A :=
fromvterm(λ nm rec, op A nm rec) f.

Definition holds (A:algebra σ)
(e:equation σ V) : UU := ∏ α,
veval A α (lhs e) = veval A α(rhs e).

Definition eqsignature : UU
:= ∑ (σ : signature)(V: hSet),
sysequations σ V.
```

Univalent category of varieties (via displayed categories):

```
Definition is_eqalgebra {σ : eqsignature}(A : algebra σ) : UU
  := ∏ e : eqs σ, holds A (geteq e)
```

Varieties

Given a signature σ, an equation has the form $t = s$, for $t, s \in T(\sigma, V)$.

```
Definition equation : UU :=
vterm σ V × vterm σ V.

Definition sysequations: UU
  := ∑ E : hSet, E → equation σ V.

Definition fromvterm {A:UU}
 (R : (∏ (nm : names σ),
    Vector A (arity nm)→A))(α:V → A)
    : vterm σ V → A.
```

We say that a σ-algebra A satisfies an equation $t = s$ when for any valuation $\alpha : V \to |A|$, $t[\alpha] = s[\alpha]$ holds in A.

```
Definition veval(A:algebra σ)
(α:V→support a):free_algebra σ V
→ support A :=
fromvterm(λ nm rec, op A nm rec) f.

Definition holds (A:algebra σ)
(e:equation σ V) : UU := ∏ α,
veval A α (lhs e) = veval A α(rhs e).

Definition eqsignature : UU
:= ∑ (σ : signature)(V: hSet),
sysequations σ V.
```

Univalent category of varieties (via displayed categories):

```
Definition is_eqalgebra {σ : eqsignature}(A : algebra σ) : UU
  := ∏ e : eqs σ, holds A (geteq e)
```

The code is publicly available from GitHub `https://github.com/amato-gianluca/UniMath`, directory `Algebra/Universal` and it is still under development.

Let's see how it works by now!

The code is publicly available from GitHub `https://github.com/amato-gianluca/UniMath`, directory `Algebra/Universal` and it is still under development.

Let's see how it works by now!

Summary and Future Work

What we have:

- A formalization of signatures, algebras, varieties;
- A "bottom-up" implementation of terms over a signature and term algebras that UniMath is able to use computationally;
  - Both proofs *and* computations concerning terms in UniMath environment $\rightsquigarrow$ Poincaré Principle;
  - A general induction principle to handle terms $\rightsquigarrow$ Reflection;
- Univalent categories of algebras and varieties built as displayed structures over HSET.

What we are working on:

- Left universal objects and Birkhoff Theorem;
- Multi-sorted signatures;
- Comparison with different formalizations.

Summary and Future Work

What we have:

- A formalization of signatures, algebras, varieties;
- A "bottom-up" implementation of terms over a signature and term algebras that UniMath is able to use computationally;
  - Both proofs *and* computations concerning terms in UniMath environment $\rightsquigarrow$ Poincaré Principle;
  - A general induction principle to handle terms $\rightsquigarrow$ Reflection;
- Univalent categories of algebras and varieties built as displayed structures over HSET.

What we are working on:

- Left universal objects and Birkhoff Theorem;
- Multi-sorted signatures;
- Comparison with different formalizations.

Many thanks
for your attention!