Opetopic Methods in Homotopy Type Theory

Antoine Allioux
Institut de Recherche en Informatique Fondamentale, Paris, France

HoTT 2023

Joint work with Eric Finster

Introduction

HoTT as a foundation of mathematics based on types regarded as *spaces* (∞-groupoids).

Introduction

HoTT as a foundation of mathematics based on types regarded as *spaces* (∞-groupoids).

Question: how to define algebraic structures when equalities behave like homotopies?

Introduction

HoTT as a foundation of mathematics based on types regarded as *spaces* (∞-groupoids).

Question: how to define algebraic structures when equalities behave like homotopies?

In set-level mathematics, one makes use of (set-level) algebraic structures to organise this higher dimensional data (operad, presheaves over a category, ...).

Introduction

HoTT as a foundation of mathematics based on types regarded as *spaces* (∞-groupoids).

Question: how to define algebraic structures when equalities behave like homotopies?

In set-level mathematics, one makes use of (set-level) algebraic structures to organise this higher dimensional data (operad, presheaves over a category, …).

Problem: leads to a situation of circular dependency in HoTT where one would have to define these structures coherently in the first place.

Our approach

We extend type theory with a universe of polynomial monads whose laws are *definitional* equalities.

Our approach

We extend type theory with a universe of polynomial monads whose laws are *definitional* equalities.

Presentation of types and their higher structures as *opetopic types*.

Our approach

We extend type theory with a universe of polynomial monads whose laws are *definitional* equalities.

Presentation of types and their higher structures as *opetopic types*.

Allows the definition of fully coherent higher algebraic structures (∞-groupoids, $(\infty, 1)$-categories).

Our approach

We extend type theory with a universe of polynomial monads whose laws are *definitional* equalities.

Presentation of types and their higher structures as *opetopic types*.

Allows the definition of fully coherent higher algebraic structures (∞-groupoids, $(\infty, 1)$-categories).

Examples: the universe of types, the opetopic type associated to a type, adjunctions, joins, …

Our type theory

Our base type theory is book HoTT with Agda's features (coinductive records, inductive-recursive types, …).

Our type theory

Our base type theory is book HoTT with Agda's features (coinductive records, inductive-recursive types, …).

Most of our work has been formalised in Agda using postulates and rewrite rules to define the universe of polynomial monads.

Polynomial monads

We extend type theory with a universe of polynomial monads $\mathcal{M} : \mathcal{U}$.

Polynomial monads

We extend type theory with a universe of polynomial monads $\mathcal{M} : \mathcal{U}$.

Elements $M : \mathcal{M}$ are *codes* for our monads. They each define a *polynomial* whose data is given by the decoding functions:

Polynomial monads

We extend type theory with a universe of polynomial monads $\mathcal{M} : \mathcal{U}$.

Elements $M : \mathcal{M}$ are *codes* for our monads. They each define a *polynomial* whose data is given by the decoding functions:

- $\mathsf{Idx}_M : \mathcal{U}$

Polynomial monads

We extend type theory with a universe of polynomial monads $\mathcal{M} : \mathcal{U}$.

Elements $M : \mathcal{M}$ are *codes* for our monads. They each define a *polynomial* whose data is given by the decoding functions:

- $\mathsf{Idx}_M : \mathcal{U}$
- $\mathsf{Cns}_M : \mathsf{Idx}_M \to \mathcal{U}$

Polynomial monads

We extend type theory with a universe of polynomial monads $\mathcal{M} : \mathcal{U}$.

Elements $M : \mathcal{M}$ are *codes* for our monads. They each define a *polynomial* whose data is given by the decoding functions:

- $\mathsf{Idx}_M : \mathcal{U}$
- $\mathsf{Cns}_M : \mathsf{Idx}_M \to \mathcal{U}$
- $\mathsf{Pos}_M : \{i : \mathsf{Idx}_M\} \to \mathsf{Cns}_M(i) \to \mathcal{U}$

Polynomial monads

We extend type theory with a universe of polynomial monads $\mathcal{M} : \mathcal{U}$.

Elements $M : \mathcal{M}$ are *codes* for our monads. They each define a *polynomial* whose data is given by the decoding functions:

- $\mathsf{Idx}_M : \mathcal{U}$
- $\mathsf{Cns}_M : \mathsf{Idx}_M \to \mathcal{U}$
- $\mathsf{Pos}_M : \{i : \mathsf{Idx}_M\} \to \mathsf{Cns}_M(i) \to \mathcal{U}$
- $\mathsf{Typ}_M : \{i : \mathsf{Idx}_M\}\ \{c : \mathsf{Cns}_M(i)\} \to \mathsf{Pos}_M(c) \to \mathsf{Idx}_M$

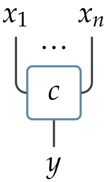
Polynomial monads

We extend type theory with a universe of polynomial monads $\mathcal{M} : \mathcal{U}$.

Elements $M : \mathcal{M}$ are *codes* for our monads. They each define a *polynomial* whose data is given by the decoding functions:

- $\mathsf{Idx}_M : \mathcal{U}$
- $\mathsf{Cns}_M : \mathsf{Idx}_M \to \mathcal{U}$
- $\mathsf{Pos}_M : \{i : \mathsf{Idx}_M\} \to \mathsf{Cns}_M(i) \to \mathcal{U}$
- $\mathsf{Typ}_M : \{i : \mathsf{Idx}_M\}\ \{c : \mathsf{Cns}_M(i)\} \to \mathsf{Pos}_M(c) \to \mathsf{Idx}_M$

Elements depicted as corollas:



Polynomial monads

A cartesian polynomial monad is a polynomial functor along with a unit η and a multiplication μ:

$$\eta_M : (i : \mathsf{Idx}_M) \to \mathsf{Cns}_M(i)$$

$$\mu_M : \{i : \mathsf{Idx}_M\}\ (c : \mathsf{Cns}_M(i)) \to \overrightarrow{\mathsf{Cns}_M}(c) \to \mathsf{Cns}_M(i)$$

Notation. For any monad $M : \mathcal{M}$,

$$\mathsf{Fam}_M :\equiv \mathsf{Idx}_M \to \mathcal{U}$$

For any type family $X : \mathsf{Fam}_M$,

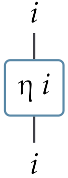
$$\overrightarrow{X}(c) :\equiv (p : \mathsf{Pos}_M(c)) \to X(\mathsf{Typ}_M(c, p))$$

Polynomial monads

The unit

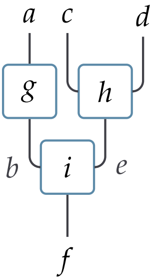
$$\eta_M : (i : \mathsf{Idx}_M) \to \mathsf{Cns}_M(i)$$

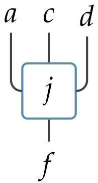
Units $\eta(i)$ are *unary* constructors whose source and target have the same sort:



$$\mu_M : \{i : \mathsf{Idx}_M\}\ (c : \mathsf{Cns}_M(i)) \to \overrightarrow{\mathsf{Cns}_M}(c) \to \mathsf{Cns}_M(i)$$

The multiplication "contracts" a tree of constructors while preserving the type of positions and their sort.





The operation μ_M is associative and unital with units η_M:

$$\mu_M(c, \lambda\, p \to \eta_M(\mathsf{Typ}_M(c, p))) \equiv c$$
$$\mu_M(\eta_M(i), d) \equiv d(\eta\text{-}\mathsf{pos}\,(i))$$
$$\mu_M(\mu_M(c, d), e) \equiv \mu_M(c, (\lambda\, p \to \mu_M(d(p), (\lambda\, q \to e(\mathsf{pair}^{\mu}(p, q))))))$$

Identity monad

We populate the universe by introducing codes for our monads and by defining the relevant decoding functions.
The identity monad $\mathsf{Id} : \mathcal{M}$ has a single unary constructor.

$$
\begin{aligned}
\mathsf{Idx}_{\mathsf{Id}} &:\equiv \mathbf{1} \\
\mathsf{Cns}_{\mathsf{Id}}(i) &:\equiv \mathbf{1} \\
\mathsf{Pos}_{\mathsf{Id}}(c) &:\equiv \mathbf{1} \\
\mathsf{Typ}_{\mathsf{Id}}(c, p) &:\equiv *
\end{aligned}
$$

The monad structure is trivial.

Baez-Dolan slice construction

For any monad $M : \mathcal{M}$ and family $X : \mathsf{Fam}_M$, their slice construction is the monad $\boxed{M/X : \mathcal{M}}$.

Baez-Dolan slice construction

For any monad $M : \mathcal{M}$ and family $X : \mathsf{Fam}_M$, their slice construction is the monad $\boxed{M/X : \mathcal{M}}$.

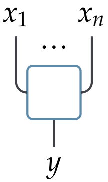
$$\mathsf{Idx}_{M/X} :\equiv (i : \mathsf{Idx}_M)\ (y : X(i))\ (c : \mathsf{Cns}_M(i))\ (x : \overrightarrow{X}(c))$$

Baez-Dolan slice construction

For any monad $M : \mathcal{M}$ and family $X : \mathsf{Fam}_M$, their slice construction is the monad $\boxed{M/X : \mathcal{M}}$.

$$\mathsf{Idx}_{M/X} :\equiv (i : \mathsf{Idx}_M)\ (y : X(i))\ (c : \mathsf{Cns}_M(i))\ (x : \overrightarrow{X}(c))$$

Indices are *frames*: quadruplets $(i, y) \triangleleft (c, x)$ representing a constructor of M whose sources and target are decorated with elements in X.

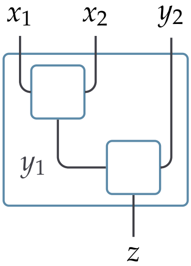


Baez-Dolan slice construction

Constructors are well-founded trees of frames which multiply to their indexing frame under the operation μ_M. Defined as an inductive types whose constructors are:

$\mathsf{lf}(i, x)$

$\mathsf{nd}((i, z) \triangleleft (c, y), t)$



Baez-Dolan slice construction

The positions of a constructor are *paths* in the tree it represents from its root to its different nodes:

$$\begin{aligned}\mathsf{Pos}_{M/X}(\mathsf{lf}(i,x)) &:\equiv \mathbf{0}\\ \mathsf{Pos}_{M/X}(\mathsf{nd}((i,z) \triangleleft (c,y),t)) &:\equiv \mathbf{1} + (p : \mathsf{Pos}_M(c)) \times \mathsf{Pos}_{M/X}(t_p)\end{aligned}$$

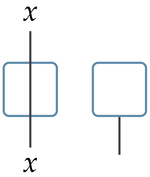
The typing function projects out the constructor associated to a node at a specified position:

$$\begin{aligned}\mathsf{Typ}_{M/X}(\mathsf{nd}((i,z) \triangleleft (c,y),t),\mathsf{inl}(*)) &:\equiv (i,z) \triangleleft (c,y)\\ \mathsf{Typ}_{M/X}(\mathsf{nd}((i,z) \triangleleft (c,y),t),\mathsf{inr}(p,q)) &:\equiv \mathsf{Typ}_{M/X}(t_p,q)\end{aligned}$$

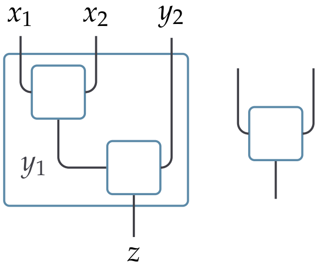
Baez-Dolan slice construction

The constructors of M/X illustrated.

$\mathsf{lf}(i, x)$



$\mathsf{nd}((i, z) \triangleleft (c, y), t)$



Algebras

An algebra for a monad M is

- a family $X_0 : \mathsf{Fam}_M$,
- a family $X_1 : \mathsf{Fam}_{M/X_0}$.

such that the type

$$(y : X_0(i)) \times X_1((i, y) \triangleleft (c, x))$$

is contractible for any constructor $c : \mathsf{Cns}_M(i)$ and values $x : \overrightarrow{X_0}(c)$.

Algebras

An algebra for a monad M is

- a family $X_0 : \mathsf{Fam}_M$,
- a family $X_1 : \mathsf{Fam}_{M/X_0}$.

such that the type

$$(y : X_0(i)) \times X_1((i, y) \triangleleft (c, x))$$

is contractible for any constructor $c : \mathsf{Cns}_M(i)$ and values $x : \overrightarrow{X_0}(c)$.

X_1 is an *entire* and *functional* relation.

Algebras

An algebra for a monad M is

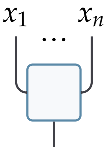
- a family $X_0 : \mathsf{Fam}_M$,
- a family $X_1 : \mathsf{Fam}_{M/X_0}$.

such that the type

$$(y : X_0(i)) \times X_1((i, y) \triangleleft (c, x))$$

is contractible for any constructor $c : \mathsf{Cns}_M(i)$ and values $x : \overrightarrow{X_0}(c)$.

X_1 is an *entire* and *functional* relation.



Algebras

An algebra for a monad M is

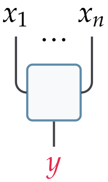
- a family $X_0 : \mathsf{Fam}_M$,
- a family $X_1 : \mathsf{Fam}_{M/X_0}$.

such that the type

$$(y : X_0(i)) \times X_1((i, y) \triangleleft (c, x))$$

is contractible for any constructor $c : \mathsf{Cns}_M(i)$ and values $x : \overrightarrow{X_0}(c)$.

X_1 is an *entire* and *functional* relation.



ALGEBRAS

An algebra for a monad M is

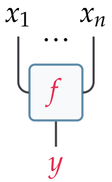
- a family $X_0 : \mathsf{Fam}_M$,
- a family $X_1 : \mathsf{Fam}_{M/X_0}$.

such that the type

$$(y : X_0(i)) \times X_1((i, y) \triangleleft (c, x))$$

is contractible for any constructor $c : \mathsf{Cns}_M(i)$ and values $x : \overrightarrow{X_0}(c)$.

X_1 is an *entire* and *functional* relation.



Opetopic types

A M-opetopic type is the data of

- a family $X : \mathsf{Fam}_M$
- a M/X-opetopic type

Opetopic types

A M-opetopic type is the data of

- a family $X : \mathsf{Fam}_M$
- a M/X-opetopic type

A M-opetopic type X is *fibrant* if it satisfies the following coinductive property:

- (X_0, X_1) is an algebra.
- $X_{>0}$ is a fibrant opetopic type.

Opetopic types

A M-opetopic type is the data of

- a family $X : \mathsf{Fam}_M$
- a M/X-opetopic type

A M-opetopic type X is *fibrant* if it satisfies the following coinductive property:

- (X_0, X_1) is an algebra.
- $X_{>0}$ is a fibrant opetopic type.

$\mathcal{O}_M$ is the type of M-opetopic types.

Opetopic types

Examples of opetopic types:

- $\infty\text{-Grp} = (X : \mathcal{O}_{\mathsf{Id}}) \times \text{is-fibrant}(X)$
- $(\infty, 1)\text{-Cat} = (X : \mathcal{O}_{\mathsf{Id}}) \times \text{is-fibrant}(X_{>0})$

The universe

0-cells

Types and their fibrant relations assemble into the $(\infty, 1)$-category

$$\mathcal{U}^o : \mathcal{O}_{\mathsf{Id}}$$

The universe

0-cells

Types and their fibrant relations assemble into the $(\infty, 1)$-category

$$\mathcal{U}^o : \mathcal{O}_{\mathsf{Id}}$$

Its family of objects $\mathcal{U}^o_0$ is the universe of types $\mathcal{U}$:

$$\mathcal{U}^o_0(*) \equiv \mathcal{U}$$

The universe

1-cells

The family of 1-cells

$$\mathcal{U}_1^o : \mathsf{Id} \times_{\mathsf{Id}/\mathcal{U}_0^o} \to \mathcal{U}$$

is a binary relation on $\mathcal{U}$.

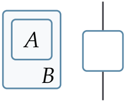
The universe
1-cells

The family of 1-cells

$$\mathcal{U}_1^o : \mathsf{Idx}_{\,\mathsf{Id}/\mathcal{U}_0^o} \to \mathcal{U}$$

is a binary relation on $\mathcal{U}$.

For example,

$\mathcal{U}_1^o($ 

$) \simeq (R : (a : A)\,(b : B) \to \mathcal{U}) \times \mathsf{is\text{-}fibrant}(R)$

The family of 2-cells

$$\mathcal{U}_2^o : \mathsf{Idx}_{\,\mathsf{Id}/\mathcal{U}_0^o/\mathcal{U}_1^o} \to \mathcal{U}$$

relates a *source* pasting diagram of 1-cells to a *target* 1-cell.

The universe

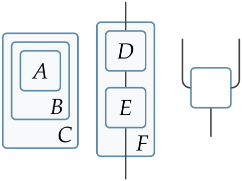
2-cells

The family of 2-cells

$$\mathcal{U}_2^o : \mathsf{Idx}_{\,\mathsf{Id}/\mathcal{U}_0^o/\mathcal{U}_1^o} \to \mathcal{U}$$

relates a *source* pasting diagram of 1-cells to a *target* 1-cell.

For example,

$\mathcal{U}_2^o($ 

$)$

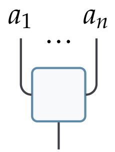
$$\simeq (R : (a : A)\,(b : B)\,(c : C) \to (d : D(a,b))\,(e : E(b,c))\,(f : F(a,c)) \to \mathcal{U}) \times \text{is-fibrant}(R)$$

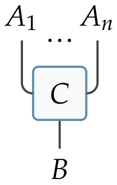
Formally, the domain of our relations are frames of the universal fibration $\mathcal{U}^o_\bullet \to \mathcal{U}^o$.

The universe

Fibrant relations

Formally, the domain of our relations are frames of the universal fibration $\mathcal{U}^o_\bullet \to \mathcal{U}^o$.

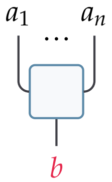


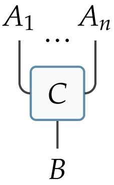


The universe

Fibrant relations

Formally, the domain of our relations are frames of the universal fibration $\mathcal{U}^o_\bullet \to \mathcal{U}^o$.

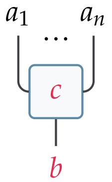


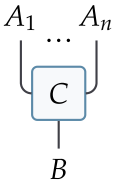


The universe

Fibrant relations

Formally, the domain of our relations are frames of the universal fibration $\mathcal{U}^o_\bullet \to \mathcal{U}^o$.





Thank you for your attention.