Higher Universal Algebra in Type Theory

Eric Finster

Joint with M. Sozeau, Y. Guiraud, A. Allioux

November 8, 2018

Univalent Type Theory

- Interpret types as *homotopy types*

Univalent Type Theory

- Interpret types as *homotopy types*
- Interpret elements of Id_X as paths, paths of paths, etc.

Univalent Type Theory

- Interpret types as *homotopy types*
- Interpret elements of Id_X as paths, paths of paths, etc.
- Univalence axiom:

$$\mathrm{Id}_U(X, Y) \xrightarrow{\simeq} \mathrm{Equiv}(X, Y)$$

Univalent Type Theory

- Interpret types as *homotopy types*
- Interpret elements of Id_X as paths, paths of paths, etc.
- Univalence axiom:

$$\mathrm{Id}_U(X, Y) \xrightarrow{\simeq} \mathrm{Equiv}(X, Y)$$

- Synthetic homotopy theory:

Univalent Type Theory

- Interpret types as *homotopy types*
- Interpret elements of Id_X as paths, paths of paths, etc.
- Univalence axiom:

$$\mathrm{Id}_U(X, Y) \xrightarrow{\simeq} \mathrm{Equiv}(X, Y)$$

- Synthetic homotopy theory:
  - Eilenberg-MacLane spaces

Univalent Type Theory

- Interpret types as *homotopy types*
- Interpret elements of Id_X as paths, paths of paths, etc.
- Univalence axiom:

$$\mathrm{Id}_U(X, Y) \xrightarrow{\simeq} \mathrm{Equiv}(X, Y)$$

- Synthetic homotopy theory:
  - Eilenberg-MacLane spaces
  - Blakers-Massey Theorem

Univalent Type Theory

- Interpret types as *homotopy types*
- Interpret elements of Id_X as paths, paths of paths, etc.
- Univalence axiom:

$$\mathrm{Id}_U(X, Y) \xrightarrow{\simeq} \mathrm{Equiv}(X, Y)$$

- Synthetic homotopy theory:
  - Eilenberg-MacLane spaces
  - Blakers-Massey Theorem
  - Calculations of Homotopy Groups

Univalent Type Theory

- Interpret types as *homotopy types*
- Interpret elements of Id_X as paths, paths of paths, etc.
- Univalence axiom:

$$\mathrm{Id}_U(X, Y) \xrightarrow{\simeq} \mathrm{Equiv}(X, Y)$$

- Synthetic homotopy theory:
  - Eilenberg-MacLane spaces
  - Blakers-Massey Theorem
  - Calculations of Homotopy Groups
  - Steenrod Operations

Univalent Type Theory

- Interpret types as *homotopy types*
- Interpret elements of Id_X as paths, paths of paths, etc.
- Univalence axiom:

$$\mathrm{Id}_U(X, Y) \xrightarrow{\simeq} \mathrm{Equiv}(X, Y)$$

- Synthetic homotopy theory:
  - Eilenberg-MacLane spaces
  - Blakers-Massey Theorem
  - Calculations of Homotopy Groups
  - Steenrod Operations
  - Serre Spectral Sequence

H-Level and the Stratification of Types

- Definition of contractibility

H-Level and the Stratification of Types

- Definition of contractibility

$$\text{is-contr} := \sum_{x:X} \prod_{y:X} (x =_X y)$$

H-Level and the Stratification of Types

- Definition of contractibility

$$\text{is-contr} := \sum_{x:X} \prod_{y:X} (x =_X y)$$

- Remark : (is-contr X) $\iff$ $(X \simeq \top)$

H-Level and the Stratification of Types

- Definition of contractibility

$$\text{is-contr} := \sum_{x:X} \prod_{y:X} (x =_X y)$$

- Remark : $(\text{is-contr } X) \iff (X \simeq \top)$
- Definition of h-level

H-Level and the Stratification of Types

- Definition of contractibility

$$\text{is-contr} := \sum_{x:X} \prod_{y:X} (x =_X y)$$

- Remark : $(\text{is-contr } X) \iff (X \simeq \top)$
- Definition of h-level

$$\text{is-of-level } (-2)\ X := \text{is-contr } X$$
$$\text{is-of-level } (S\ n)\ X := \prod_{x,y:X} \text{is-of-level } n\ (x =_X y)$$

H-Level and the Stratification of Types

- Definition of contractibility

$$\text{is-contr} := \sum_{x:X} \prod_{y:X} (x =_X y)$$

- Remark : $(\text{is-contr } X) \iff (X \simeq \top)$
- Definition of h-level

$$\text{is-of-level } (-2)\ X := \text{is-contr } X$$

$$\text{is-of-level } (S\ n)\ X := \prod_{x,y:X} \text{is-of-level } n\ (x =_X y)$$

- Remark : There are types which are not of *any* finite *h*-level, i.e., this filtration is not exhaustive

Voevodsky's Vision

- The Mathematics of Cantor:
  - Sets and structured sets (h-level 0)

Voevodsky's Vision

- The Mathematics of Cantor:
  - Sets and structured sets (h-level 0)
- Then Mathematics of Grothendieck
  - The mathematics of categories and structured categories (h-level 1)

Voevodsky's Vision

- The Mathematics of Cantor:
  - Sets and structured sets (h-level 0)
- Then Mathematics of Grothendieck
  - The mathematics of categories and structured categories (h-level 1)
- 21st Century Mathematics
  - The mathematics of strutures on types of higher h-level

Category Theory for Types?

Definition

A *category* consists of the data ...

Category Theory for Types?

Definition
A *category* consists of the data ...

1. Objects:

$$Ob : Type$$

Category Theory for Types?

Definition

A *category* consists of the data ...

1. Objects:

$$Ob : Type$$

2. Morphisms:

$$Hom : Ob \to Ob \to Type$$

Category Theory for Types?

Definition

A *category* consists of the data ...

1. Objects:

$$Ob : Type$$

2. Morphisms:

$$Hom : Ob \to Ob \to Type$$

3. Identity:

$$\mathrm{id} : (x : Ob) \to Hom\, x\, x$$

Category Theory for Types?

Definition

A *category* consists of the data ...

1. Objects:
$$Ob : Type$$
2. Morphisms:
$$Hom : Ob \to Ob \to Type$$
3. Identity:
$$\mathrm{id} : (x : Ob) \to Hom\, x\, x$$
4. Composition:
$$\circ : (x, y, z : Ob)(f : Hom\, y\, z)(g : Hom\, x\, y) \to Hom\, x\, z$$

Laws for Categories

Definition (Cont'd)

… satisfying the laws

Laws for Categories

Definition (Cont'd)

... satisfying the laws

5. Unit Laws:

$$\text{unit-l} : (xy : Ob)(f : Hom\, x\, y) \to f = id_x \circ f$$
$$\text{unit-r} : (xy : Ob)(f : Hom\, x\, y) \to id_y \circ f = f$$

Laws for Categories

Definition (Cont'd)

... satisfying the laws

5. Unit Laws:

$$\text{unit-l} : (xy : Ob)(f : Hom\,x\,y) \to f = id_x \circ f$$
$$\text{unit-r} : (xy : Ob)(f : Hom\,x\,y) \to id_y \circ f = f$$

6. Associative Law:

$$\text{assoc} : (xyzw : Ob)(f : Hom\,z\,w)(g : Hom\,y\,z)(h : Hom\,x\,y) \to ((f \circ g) \circ h) = (f \circ (g \circ h))$$

Slice Category?

For $Z \in C$, we would like to define the slice category C/Z.

Slice Category?

For $Z \in C$, we would like to define the slice category C/Z.

1. Objects:

$$f : X \to Z$$

Slice Category?

For $Z \in C$, we would like to define the slice category C/Z.

1. Objects:
$$f : X \to Z$$
2. Morphisms:
$$\begin{tikzcd} X \arrow[rr, "h"] \arrow[dr, "f"'] & & Y \arrow[dl, "g"] \\ & Z & \end{tikzcd}$$

Slice Category?

For $Z \in C$, we would like to define the slice category C/Z.

1. Objects:

$$f : X \to Z$$

2. Morphisms:

$$\begin{tikzcd} X \arrow[rr, "h"] \arrow[dr, "f"'] & & Y \arrow[dl, "g"] \\ & Z & \end{tikzcd}$$

But we immediately run into a problem:

Slice Category?

For $Z \in C$, we would like to define the slice category C/Z.

1. Objects:

$$f : X \to Z$$

2. Morphisms:

$$\begin{tikzcd} X \arrow[rr, "h"] \arrow[dr, "f"'] & & Y \arrow[dl, "g"] \\ & Z & \end{tikzcd}$$

But we immediately run into a problem:

- To define the composition in C/Z, we must use that composition in C is associative.

Slice Category?

For $Z \in C$, we would like to define the slice category C/Z.

1. Objects:

$$f : X \to Z$$

2. Morphisms:

$$\begin{tikzcd} X \arrow[rr, "h"] \arrow[dr, "f"'] & & Y \arrow[dl, "g"] \\ & Z & \end{tikzcd}$$

But we immediately run into a problem:

- To define the composition in C/Z, we must use that composition in C is associative.
- We cannot show that the composition in C/Z is *itself* associative without more axioms!

Coherence Conditions

A sufficient condition for composition in C/Z to be associative is the well known pentagon identity:

$$\begin{tikzcd}
 & & (g_1 g_2)(g_3 g_4) \arrow[drr, "a_{g_1, g_2, g_3 g_4}"] & & \\
((g_1 g_2) g_3) g_4 \arrow[urr, "a_{g_1 g_2, g_3, g_4}"] \arrow[dr, "a_{g_1, g_2, g_3} 1_{g_4}"'] & & & & g_1(g_2(g_3 g_4)) \\
 & (g_1(g_2 g_3)) g_4 \arrow[rr, "a_{g_1, g_2 g_3, g_4}"] & & g_1((g_2 g_3) g_4) \arrow[ur, "1_{g_1} a_{g_2, g_3, g_4}"'] &
\end{tikzcd}$$

We can amend the definition of category to included this new law (which lives in a doubly iterated identity type) ...

A Vicious Circle Starts

... But!

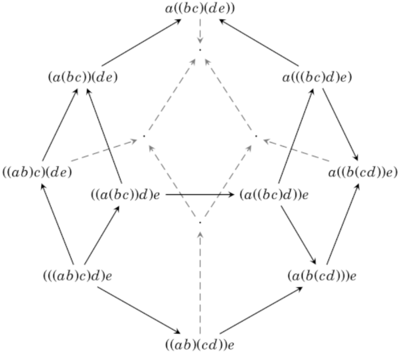
Then we will need to prove the pentagon axiom for C/Z

A Vicious Circle Starts

... But!

Then we will need to prove the pentagon axiom for C/Z

And we quickly find that in order to do so, we need *another* axiom:



The classic “coherence problem”

Now it becomes clear that this completely elementary construction is not well defined unless we add *infinitely many* laws to our definition of category

C	C/Z
assoc	$\circ$
assoc -2	assoc
assoc -3	assoc -2
assoc -4	assoc -3
$\vdots$	$\vdots$

The classic "coherence problem"

Now it becomes clear that this completely elementary construction is not well defined unless we add *infinitely many* laws to our definition of category

C	C/Z
assoc	$\circ$
assoc -2	assoc
assoc -3	assoc -2
assoc -4	assoc -3
$\vdots$	$\vdots$

- We can avoid this problem by supposing that our types are h-sets. (or, for example, that they have decidable equality)

The classic "coherence problem"

Now it becomes clear that this completely elementary construction is not well defined unless we add *infinitely many* laws to our definition of category

C	C/Z
assoc	$\circ$
assoc -2	assoc
assoc -3	assoc -2
assoc -4	assoc -3
$\vdots$	$\vdots$

- We can avoid this problem by supposing that our types are h-sets. (or, for example, that they have decidable equality)
- But what is the correct notion of category for a general type?

Higher Universal Algebra

- Categories are not the only structure we would like to generalize to arbitrary types:

Higher Universal Algebra

- Categories are not the only structure we would like to generalize to arbitrary types:
  1. Monoids

Higher Universal Algebra

- Categories are not the only structure we would like to generalize to arbitrary types:
  1. Monoids
  2. Abelian Groups

Higher Universal Algebra

- Categories are not the only structure we would like to generalize to arbitrary types:
  1. Monoids
  2. Abelian Groups
  3. Commutative Rings

Higher Universal Algebra

- Categories are not the only structure we would like to generalize to arbitrary types:
  1. Monoids
  2. Abelian Groups
  3. Commutative Rings
  4. Modules

Higher Universal Algebra

- Categories are not the only structure we would like to generalize to arbitrary types:
  1. Monoids
  2. Abelian Groups
  3. Commutative Rings
  4. Modules
  5. Lie Algebras

Higher Universal Algebra

- Categories are not the only structure we would like to generalize to arbitrary types:
  1. Monoids
  2. Abelian Groups
  3. Commutative Rings
  4. Modules
  5. Lie Algebras
  6. Operads

Higher Universal Algebra

- Categories are not the only structure we would like to generalize to arbitrary types:
  1. Monoids
  2. Abelian Groups
  3. Commutative Rings
  4. Modules
  5. Lie Algebras
  6. Operads
  7. n-categories

Higher Universal Algebra

- Categories are not the only structure we would like to generalize to arbitrary types:
  1. Monoids
  2. Abelian Groups
  3. Commutative Rings
  4. Modules
  5. Lie Algebras
  6. Operads
  7. n-categories
  8. etc ...

Higher Universal Algebra

- Categories are not the only structure we would like to generalize to arbitrary types:
  1. Monoids
  2. Abelian Groups
  3. Commutative Rings
  4. Modules
  5. Lie Algebras
  6. Operads
  7. n-categories
  8. etc ...
- A number of these "higher structures" already exist (and are *useful*) in modern algebraic topology and algebraic geometry

Higher Universal Algebra

- Categories are not the only structure we would like to generalize to arbitrary types:
  1. Monoids
  2. Abelian Groups
  3. Commutative Rings
  4. Modules
  5. Lie Algebras
  6. Operads
  7. *n*-categories
  8. etc ...
- A number of these "higher structures" already exist (and are *useful*) in modern algebraic topology and algebraic geometry
- We need a *general* theory of algebra on types

Polynomials as Multi-sorted Signatures

Definition

Fix a type I of *sorts*. A *polynomial* over I is the data of

Polynomials as Multi-sorted Signatures

Definition

Fix a type I of *sorts*. A *polynomial* over I is the data of

1. A family of *operations*

$$\mathrm{Op} : I \to \mathit{Type}$$

Polynomials as Multi-sorted Signatures

Definition

Fix a type I of *sorts*. A *polynomial* over I is the data of

1. A family of *operations*

$$\mathrm{Op} : I \to \mathit{Type}$$

2. For each operation, a family of sorted *parameters*

$$\mathrm{Param} : (i : I)(f : \mathrm{Op}\, i) \to I \to \mathit{Type}$$

Polynomials as Multi-sorted Signatures

Definition

Fix a type I of *sorts*. A *polynomial* over I is the data of

1. A family of *operations*

$$\mathrm{Op} : I \to \mathit{Type}$$

2. For each operation, a family of sorted *parameters*

$$\mathrm{Param} : (i : I)(f : \mathrm{Op}\, i) \to I \to \mathit{Type}$$

- For $i : I$, an element $f : \mathrm{Op}\, i$ represents an operation whose *output* sort is i.

Polynomials as Multi-sorted Signatures

Definition

Fix a type I of *sorts*. A *polynomial* over I is the data of

1. A family of *operations*

$$\mathrm{Op} : I \to \mathit{Type}$$

2. For each operation, a family of sorted *parameters*

$$\mathrm{Param} : (i : I)(f : \mathrm{Op}\, i) \to I \to \mathit{Type}$$

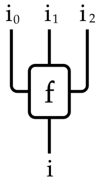
- For $i : I$, an element $f : \mathrm{Op}\, i$ represents an operation whose *output* sort is i.
- For $i : I$, $f : \mathrm{Op}\, i$ and $j : I$, an element $p : \mathrm{Param}\, i\, f\, j$ represents and *input* parameter of sort j.

Graphical Representation

- It is helpful to think of our signature as having "elements" consisting of the operations
- We can think of them as typed symbols:

$$f(i_0, i_1, i_2) : i$$

- Or we can depict them graphically:



Trees

Associated to any polynomial $P : \mathrm{Poly}\, I$ is its W-Type, that is, the type of "well-typed terms" generated by the signature.

Definition

Let $P : \mathrm{Poly}\, I$ be a polynomial. Define

$$
\begin{aligned}
W\ P &: I \to \mathit{Type} \\
\mathrm{lf} &: (i : I) \to W\ P\ i \\
\mathrm{nd} &: (i : I) \to (f : \mathrm{Op}\, P i) \\
&\quad \to (\delta : (j : J)(p : \mathrm{Param}\, f\, j) \to \mathrm{W}\ P\ j) \\
&\quad \to \mathrm{W}\ P\ i
\end{aligned}
$$

Representations of Trees

- We can represent elements of $\mathrm{W}\,P$ as terms

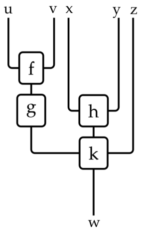
$$k(g(f(u,v)),h(x,y),z) : w$$

Representations of Trees

- We can represent elements of W P as terms

$$k(g(f(u,v)),h(x,y),z):w$$

- Or graphically as actual trees:

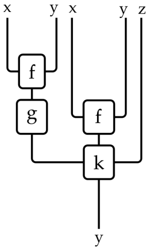


Leaves and Nodes

It is not hard to define the "type of leaves" and "type of nodes" of a give tree. These have types:

$$\text{Leaf} : \{i : I\}(w : \mathrm{W}\, P\, i)(j : I) \to \mathit{Type}$$

$$\text{Node} : \{i : I\}(w : \mathrm{W}\, P\, i)\{j : I\}(g : \mathrm{Op}\, P\, j) \to \mathit{Type}$$



$$\text{Leaf}\ w\, x = \{\bullet, \bullet\}$$

$$\text{Node}\ w\, f = \{\bullet, \bullet\}$$

Adding Relations to our Signature

- The next step is to add some axioms/relations to our structure

Adding Relations to our Signature

- The next step is to add some axioms/relations to our structure
- In order to single out a “tractable” class of structures we will consider relations which

Adding Relations to our Signature

- The next step is to add some axioms/relations to our structure
- In order to single out a "tractable" class of structures we will consider relations which
  1. Relate some term with a single operation

Adding Relations to our Signature

- The next step is to add some axioms/relations to our structure
- In order to single out a "tractable" class of structures we will consider relations which
  1. Relate some term with a single operation
  2. Preserve the number of variables

Adding Relations to our Signature

- The next step is to add some axioms/relations to our structure
- In order to single out a "tractable" class of structures we will consider relations which
    1. Relate some term with a single operation
    2. Preserve the number of variables

$$\begin{aligned} \text{Good:} \quad & f(g(x,y),z) = k(x,y,z) \\ \text{Bad:} \quad & f(g(x,y),z) = h(x,k(y)) \end{aligned}$$

Adding Relations to our Signature

- The next step is to add some axioms/relations to our structure
- In order to single out a "tractable" class of structures we will consider relations which
  1. Relate some term with a single operation
  2. Preserve the number of variables

$$\begin{aligned} \text{Good:} \quad & f(g(x,y),z) = k(x,y,z) \\ \text{Bad:} \quad & f(g(x,y),z) = h(x,k(y)) \end{aligned}$$

- The reason for these restrictions is we can then encode such relations as a multiplication operator on the signature itself.

Frames and Magmas

Definition

Let $P : \mathrm{Poly}\, I$ be a polynomial $w : \mathrm{W}\, P\, i$ a tree and $f : \mathrm{Op}\, P\, i$ and operation. A *frame* from w to f is

$$(j : I) \to \mathrm{Leaf}\, w\, j \simeq \mathrm{Param}\, P\, f\, j$$

Frames and Magmas

Definition

Let $P : \mathrm{Poly}\, I$ be a polynomial $w : \mathrm{W}\, P\, i$ a tree and $f : \mathrm{Op}\, P\, i$ and operation. A *frame* from w to f is

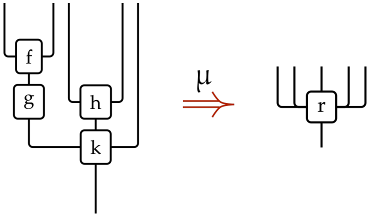
$$(j : I) \to \mathrm{Leaf}\, w\, j \simeq \mathrm{Param}\, P\, f\, j$$

Definition

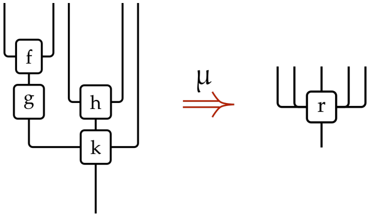
Let $P : \mathrm{Poly}\, I$ be a polynomial. A *polynomial magma* M over P is

1. A function $\mu : (i : I) \to \mathrm{W}\, P\, i \to \mathrm{Op}\, P\, i$
2. A function $\mu_f : (i : I)(w : \mathrm{W}\, P\, i) \to \mathrm{Frame}\, P\, w\, (\mu\, w)$

Visualization of Relations

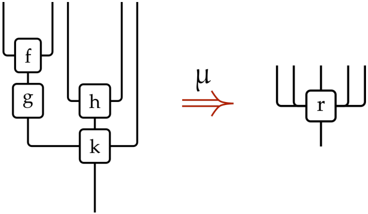


Visualization of Relations



$$k(g(f(u,v)),h(x,y),z) = r(u,v,x,y,z)$$

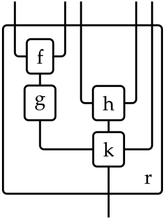
Visualization of Relations



$$k(g(f(u, v)), h(x, y), z) = r(u, v, x, y, z)$$

$$\mu(f, g, h, k) = r$$

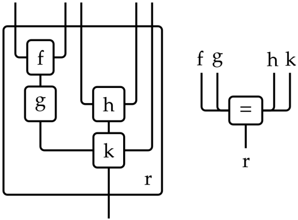
Visualization of Relations



$$k(g(f(u,v)),h(x,y),z) = r(u,v,x,y,z)$$

$$\mu(f,g,h,k) = r$$

Visualization of Relations



$$k(g(f(u,v)),h(x,y),z) = r(u,v,x,y,z)$$

$$\mu(f,g,h,k) = r$$

The Slice of a Polynomial

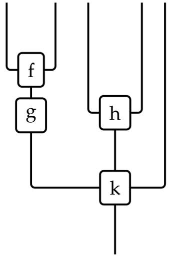
Given a polynomial $P : \mathsf{Poly}\, I$ and a magma $M : \mathsf{PolyMagma}\, P$, we can define a new polynomial $P//M : \mathsf{Poly}(\Sigma\, I\ \mathsf{Op})$ as follows:

$$\mathsf{Op}(P//M)\,(i, f) := \sum_{w : \mathsf{W}\, P\, i} \mu w = f$$

$$\mathsf{Param}(P//M)\,(i, f)(w, e)(j, g) := \mathsf{Node}\, w\, g$$

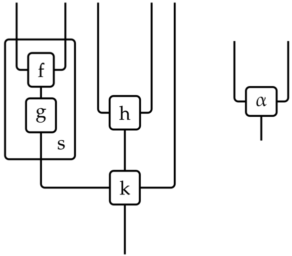
Visualizing Iterated Compositions

A tree in the slice polynomial $P//M$ can be visualized as representing a *sequence* of applications of the multiplication μ.



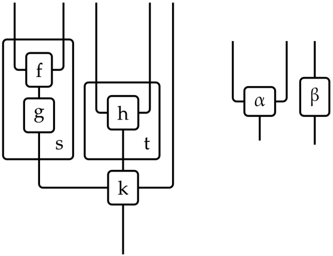
Visualizing Iterated Compositions

A tree in the slice polynomial $P//M$ can be visualized as representing a *sequence* of applications of the multiplication μ.



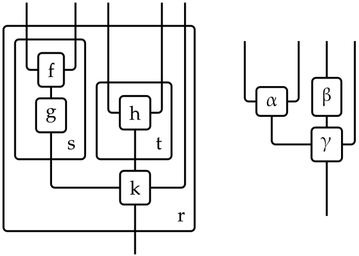
Visualizing Iterated Compositions

A tree in the slice polynomial $P//M$ can be visualized as representing a *sequence* of applications of the multiplication μ.



Visualizing Iterated Compositions

A tree in the slice polynomial $P//M$ can be visualized as representing a *sequence* of applications of the multiplication μ.

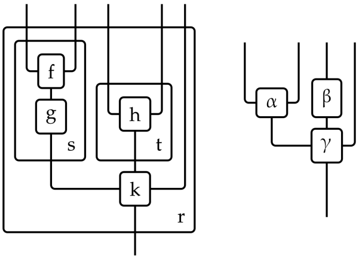


Flattening

There is a function:

$$\text{flatten} : (i : I)(f : \text{Op}\, P\, i) \to \mathsf{W}(P//M)\,(i, f) \to \mathsf{W}\, P\, i$$

which, given a *pasting diagram* extracts its "boundary":

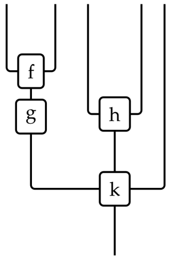


Flattening

There is a function:

$$\mathsf{flatten} : (i : I)(f : \mathsf{Op}\, P\, i) \to \mathsf{W}(P//M)\,(i, f) \to \mathsf{W}\, P\, i$$

which, given a *pasting diagram* extracts its "boundary":



Invariance by Subdivision

A *coherence witness* Ψ for M is proof that the multiplication μ is invariant under all subdivisions.

$$\begin{aligned}\Psi : (i : I)(f : \mathrm{Op}\, P\, i)(pd : \mathrm{W}(P//M)\,(i, f)) \\ \to \mu(\mathit{flatten}\, pd) = f\end{aligned}$$

Invariance by Subdivision

A *coherence witness* Ψ for M is proof that the multiplication μ is invariant under all subdivisions.

$$\begin{aligned}\Psi : (i : I)(f : \mathrm{Op}\, P\, i)(pd : \mathrm{W}(P//M)\,(i, f)) \\ \to \mu(\mathit{flatten}\, pd) = f\end{aligned}$$

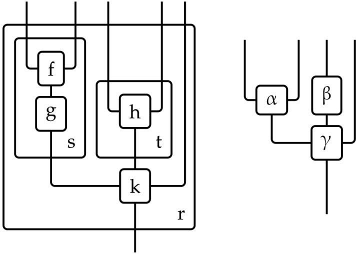
It turns out that, given a coherence witness Ψ, we can define a magma structure on $P//M$ which we will write M_Ψ.

Invariance by Subdivision

A *coherence witness* Ψ for M is proof that the multiplication μ is invariant under all subdivisions.

$$\begin{aligned}\Psi : (i : I)(f : \mathrm{Op}\, P\, i)(pd : \mathrm{W}(P//M)\,(i, f)) \\ \to \mu(\mathit{flatten}\, pd) = f\end{aligned}$$

It turns out that, given a coherence witness Ψ, we can define a magma structure on $P//M$ which we will write M_Ψ.

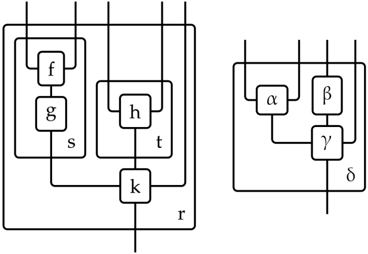


Invariance by Subdivision

A *coherence witness* Ψ for M is proof that the multiplication μ is invariant under all subdivisions.

$$\begin{aligned}\Psi : (i : I)(f : \mathrm{Op}\, P\, i)(pd : \mathrm{W}(P//M)\,(i, f))\\ \to \mu(\mathit{flatten}\, pd) = f\end{aligned}$$

It turns out that, given a coherence witness Ψ, we can define a magma structure on $P//M$ which we will write M_Ψ.

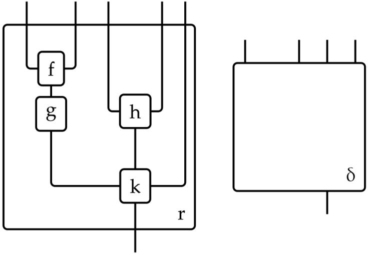


Invariance by Subdivision

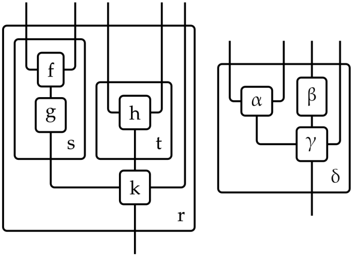
A *coherence witness* Ψ for M is proof that the multiplication μ is invariant under all subdivisions.

$$\Psi : (i : I)(f : \mathsf{Op}\, P\, i)(pd : \mathsf{W}(P//M)\,(i, f)) \\ \to \mu(\mathit{flatten}\, pd) = f$$

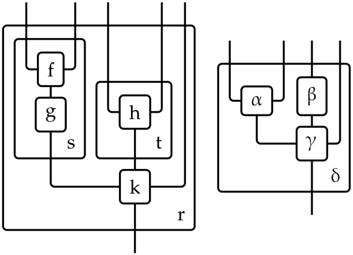
It turns out that, given a coherence witness Ψ, we can define a magma structure on $P//M$ which we will write M_Ψ.



Invariance and Associativity

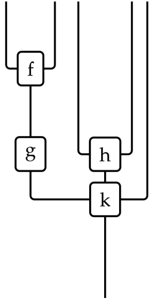


Invariance and Associativity

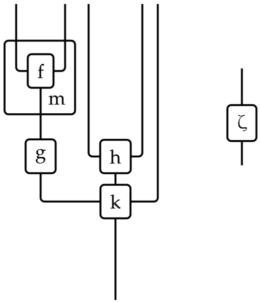


$$\mu(\mu(f, g), \mu(h), k) = r$$
$$\mu(f, g, h, k) = r$$

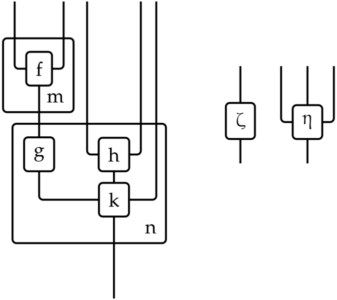
Invariance and Associativity



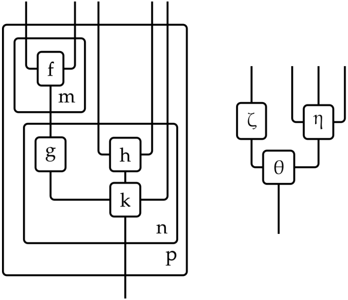
Invariance and Associativity



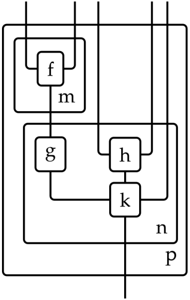
Invariance and Associativity

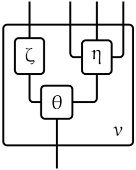


Invariance and Associativity

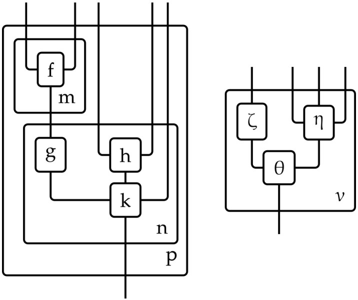


Invariance and Associativity



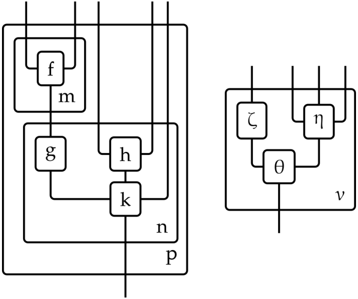


Invariance and Associativity



$$\mu(\mu(f), \mu(g, h, k)) = p$$
$$\mu(f, g, h, k) = p$$

Invariance and Associativity



$$p = \mu(f, g, h, k) = r$$
$$\mu(\mu(f, g), \mu(h), k) = \mu(\mu(f), \mu(g, h, k))$$

Polynomial Monads

The advantage of this formulation is that we can now define a coherent structure as one for which one can find an infinite sequence of such extensions.

Definition

A coherence structure for M consists of

1. A coherence witness Ψ for M
2. Coinductively, a coherence structure for M_{Ψ}

Polynomial Monads

The advantage of this formulation is that we can now define a coherent structure as one for which one can find an infinite sequence of such extensions.

Definition

A coherence structure for M consists of

1. A coherence witness Ψ for M
2. Coinductively, a coherence structure for M_Ψ

Definition

A *polynomial monad* is consists of

1. A polynomial $P : \mathrm{Poly}\, I$
2. A magma $M : \mathrm{PolyMagma}\, P$
3. A coherence structure C for M

Applications and future work

- A special case of this definition gives a complete definition of category structure on a type

Applications and future work

- A special case of this definition gives a complete definition of category structure on a type
- More generally, this definition gives us an internalization of *higher operads*

Applications and future work

- A special case of this definition gives a complete definition of category structure on a type
- More generally, this definition gives us an internalization of *higher operads*
- Remains to explore examples and use these techniques to prove coherence theorems

Thank you!