A Type Theory for Strictly Associative Infinity Categories

Eric Finster Alex Rice Jamie Vicary

22nd October 2021

Outline

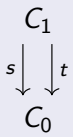
Globular Sets

Globular sets are one natural shape of higher categories:

Globular Sets

Globular sets are one natural shape of higher categories:

Ordinary 1-categories

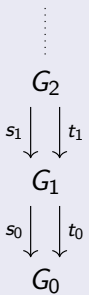


Globular Sets

Globular sets are one natural shape of higher categories:

Ordinary 1-categories

Globular sets



Globular Sets

Definition

A *globular set* $\mathcal{G}$ consists of sets G_n for each n and maps $s_n, t_n : G_{n+1} \to G_n$ for each n such that the following *globularity conditions* hold:

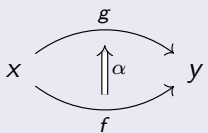
$$s_n \circ s_{n+1} = s_n \circ t_{n+1}$$
$$t_n \circ s_{n+1} = t_n \circ t_{n+1}$$

Globular Sets

Definition

A *globular set* $\mathcal{G}$ consists of sets G_n for each n and maps $s_n, t_n : G_{n+1} \to G_n$ for each n such that the following *globularity conditions* hold:

$$s_n \circ s_{n+1} = s_n \circ t_{n+1}$$
$$t_n \circ s_{n+1} = t_n \circ t_{n+1}$$

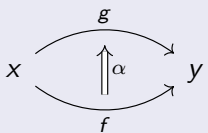


Globular Sets

Definition

A *globular set* $\mathcal{G}$ consists of sets G_n for each n and maps $s_n, t_n : G_{n+1} \to G_n$ for each n such that the following *globularity conditions* hold:

$$s_n \circ s_{n+1} = s_n \circ t_{n+1}$$
$$t_n \circ s_{n+1} = t_n \circ t_{n+1}$$



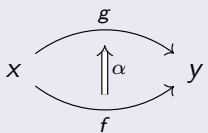
We will write $f : x \to y$ for an object f in G_{n+1} with $s(f) = x$ and $t(f) = y$.

Globular Sets

Definition

A *globular set* $\mathcal{G}$ consists of sets G_n for each n and maps $s_n, t_n : G_{n+1} \to G_n$ for each n such that the following *globularity conditions* hold:

$$s_n \circ s_{n+1} = s_n \circ t_{n+1}$$
$$t_n \circ s_{n+1} = t_n \circ t_{n+1}$$



We will write $f : x \to y$ for an object f in G_{n+1} with $s(f) = x$ and $t(f) = y$.

Terminal globular set

The *terminal globular set* has one cell at each dimension and all source and target maps are uniquely defined.

Compostition in Globular Sets

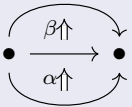
Composition of 1 cells

Compostition in Globular Sets

Composition of 1 cells

Composition of 2 cells

Composition along a 1-boundary:

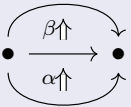


Compostition in Globular Sets

Composition of 1 cells

Composition of 2 cells

Composition along a 1-boundary:



Composition along a 0-boundary:

Strict Infinity Categories - Composition

In a *strict infinity category* we have binary composition of n-cells for along a k boundary for all $k < n$.

Composition

If f and g are n-cells with the k-target of f equalling the k-source of g then there is an n-cell $f \circ_k g$.

Strict Infinity Categories - Composition

In a *strict infinity category* we have binary composition of n-cells for along a k boundary for all $k < n$.

Composition

If f and g are n-cells with the k-target of f equalling the k-source of g then there is an n-cell $f \circ_k g$.

Identities

For each n-cell f there is an $(n+1)$-cell $\mathrm{id}_f : f \to f$.

Strict Infinity Categories - Associativity

If $0 \leq k < n$ and f, g, and h are n-cells then:

$$f \circ_k (g \circ_k h) = (f \circ_k g) \circ_k h$$

If $0 \leq k < n$ and f, g, and h are n-cells then:

$$f \circ_k (g \circ_k h) = (f \circ_k g) \circ_k h$$

Associativity of 1-cells

Given $\bullet \xrightarrow{f} \bullet \xrightarrow{g} \bullet \xrightarrow{h} \bullet$ we have:

$$f \circ_0 (g \circ_0 h) = (f \circ_0 g) \circ_0 h$$

Strict Infinity Categories - Identities

If $0 \leq k < n$ and f is an n-cell with k-source x and k-target y then:

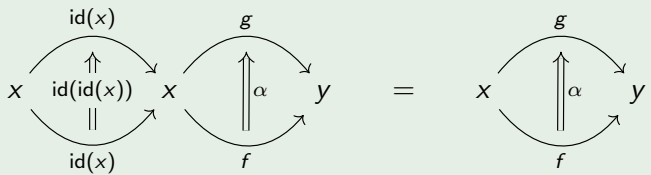
$$\mathrm{id}^{n-k}(x) \circ_k f = f = f \circ_k \mathrm{id}(y)$$

If $0 \leq k < n$ and f is an n-cell with k-source x and k-target y then:

$$\mathrm{id}^{n-k}(x) \circ_k f = f = f \circ_k \mathrm{id}(y)$$

Identity on 2-cell

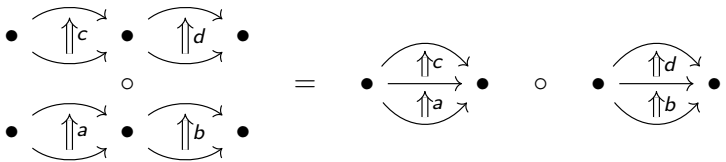
Given $f, g : x \to y$ and $\alpha : f \to g$ we have:



Strict Infinity Categories - Interchange

If $0 \leq q < p < n$ and a, b, c, d are n-cells then:

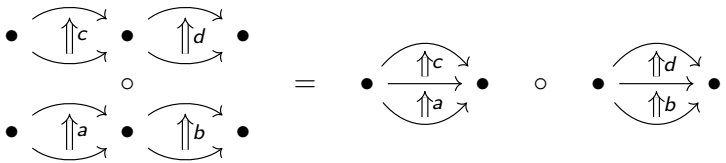
$$(a \circ_p b) \circ_q (c \circ_p d) = (a \circ_q c) \circ_p (b \circ_q d)$$



Strict Infinity Categories - Interchange

If $0 \leq q < p < n$ and a, b, c, d are n-cells then:

$$(a \circ_p b) \circ_q (c \circ_p d) = (a \circ_q c) \circ_p (b \circ_q d)$$



Further if $f \circ_k g$ is well defined then:

$$\mathrm{id}_f \circ_k \mathrm{id}(g) = \mathrm{id}(f \circ_k g)$$

Monoidal Categories

Monoidal categories are instances of infinity categories.

Definition (Monoidal category)

A *monoidal category* is a category $\mathcal{C}$ equipped with a functor $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ and a unit object I satisfying some conditions.

Monoidal Categories

Monoidal categories are instances of infinity categories.

Definition (Monoidal category)

A *monoidal category* is a category $\mathcal{C}$ equipped with a functor $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ and a unit object I satisfying some conditions.

A strict infinity category with one object and no non-identity n-cells for n higher than 2 is a strict monoidal category.

Weakness

If a category has all products and a terminal object, then it can be given the structure of a monoidal category.

Weakness

If a category has all products and a terminal object, then it can be given the structure of a monoidal category.

Set

The category **Set** is a monoidal category with $\otimes$ given by cartesian product and unit object given by the singleton set.

Weakness

If a category has all products and a terminal object, then it can be given the structure of a monoidal category.

Set

The category **Set** is a monoidal category with $\otimes$ given by cartesian product and unit object given by the singleton set.

The monoidal product in **Set** is *not* strict.

Strictification

Strict categories are easier to work with while there are more examples of *weak* categories.

Strictification

Strict categories are easier to work with while there are more examples of *weak* categories.

All weak monoidal categories and all weak 2-categories are equivalent to a strict version of themselves.

Strictification

Strict categories are easier to work with while there are more examples of *weak* categories.

All weak monoidal categories and all weak 2-categories are equivalent to a strict version of themselves.

However this is no longer possible at dimensions 3 and higher.

Semistrictness

Since full strictification is not possible, we want to do the best possible.

Associators
Unitors
Interchangers

Semistrictness

Since full strictification is not possible, we want to do the best possible.

Therefore, we look for *semistrict* definitions of infinity categories.

Associators
Unitors
Interchangers

Semistrictness

Since full strictification is not possible, we want to do the best possible.

Therefore, we look for *semistrict* definitions of infinity categories.

We can strictify:

Associators
Unitors
Interchangers

Semistrictness

Since full strictification is not possible, we want to do the best possible.

Therefore, we look for *semistrict* definitions of infinity categories.

We can strictify:

	Strict ∞-**Cat**
Associators	✓
Unitors	✓
Interchangers	✓

Semistrictness

Since full strictification is not possible, we want to do the best possible.

Therefore, we look for *semistrict* definitions of infinity categories.

We can strictify:

	Strict ∞-**Cat**	Simpson
Associators	✓	✓
Unitors	✓	
Interchangers	✓	✓

Semistrictness

Since full strictification is not possible, we want to do the best possible.

Therefore, we look for *semistrict* definitions of infinity categories.

We can strictify:

	Strict ∞-**Cat**	Simpson
Associators	✓	✓
Unitors	✓	
Interchangers	✓	✓

Semistrictness

Since full strictification is not possible, we want to do the best possible.

Therefore, we look for *semistrict* definitions of infinity categories.

We can strictify:

	Strict ∞-**Cat**	Simpson	Grey
Associators	✓	✓	✓
Unitors	✓		✓
Interchangers	✓	✓	

Semistrictness

Since full strictification is not possible, we want to do the best possible.

Therefore, we look for *semistrict* definitions of infinity categories.

We can strictify:

	Strict ∞-**Cat**	Simpson	Grey	$\mathrm{CaTT}_{\mathrm{su}}$
Associators	✓	✓	✓	
Unitors	✓		✓	✓
Interchangers	✓	✓		

Semistrictness

Since full strictification is not possible, we want to do the best possible.

Therefore, we look for *semistrict* definitions of infinity categories.

We can strictify:

	Strict ∞-**Cat**	Simpson	Grey	$\mathrm{CaTT}_{\mathrm{su}}$	$\mathrm{CaTT}_{\mathrm{sa}}$
Associators	✓	✓	✓		✓
Unitors	✓		✓	✓	
Interchangers	✓	✓			

Pasting Diagrams

A *pasting diagram* represents a composition that can be done in an infinity category.

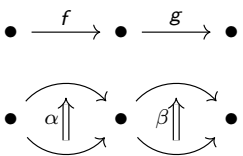
More precisely it is an object of the *free strict infinity category* on the terminal globular set.

Pasting Diagrams

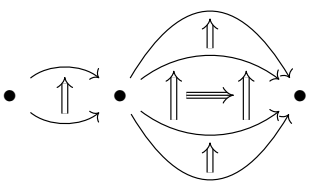
A *pasting diagram* represents a composition that can be done in an infinity category.

More precisely it is an object of the *free strict infinity category* on the terminal globular set.

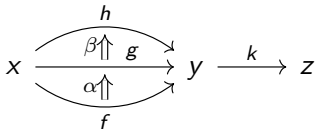
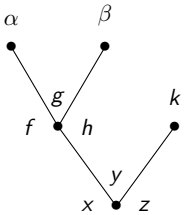
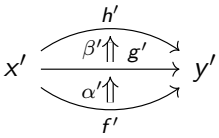
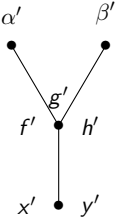
The compositions we have already seen form pasting diagrams.



We can also form more complicated compositions as pasting diagrams.



Trees

Pasting diagram	Batanin tree
	
	

Overview

CaTT is a type theory for *weak infinity categories*. It allows us to build and describe the free generated words in an infinity category.

Overview

CaTT is a type theory for *weak infinity categories*. It allows us to build and describe the free generated words in an infinity category.

There are 4 pieces of syntax, all defined by mutual induction:

- Contexts: Represent the *generating* data of an infinity category.

Overview

CaTT is a type theory for *weak infinity categories*. It allows us to build and describe the free generated words in an infinity category.

There are 4 pieces of syntax, all defined by mutual induction:

- Contexts: Represent the *generating* data of an infinity category.
- Terms: A term in a context Γ represents a *word* that can be built from the generators in Γ.

Overview

CaTT is a type theory for *weak infinity categories*. It allows us to build and describe the free generated words in an infinity category.

There are 4 pieces of syntax, all defined by mutual induction:

- Contexts: Represent the *generating* data of an infinity category.
- Terms: A term in a context Γ represents a *word* that can be built from the generators in Γ.
- Types: A type contains all the information of the *sources* and *targets* for a term.

Overview

CaTT is a type theory for *weak infinity categories*. It allows us to build and describe the free generated words in an infinity category.

There are 4 pieces of syntax, all defined by mutual induction:

- Contexts: Represent the *generating* data of an infinity category.
- Terms: A term in a context Γ represents a *word* that can be built from the generators in Γ.
- Types: A type contains all the information of the *sources* and *targets* for a term.
- Substitutions: A substitution is a *morphism* between contexts.

Types and Variables

Variables of a context Γ represent the *generators* of the infinity category generated by Γ. Their type is given by the context.

Types and Variables

Variables of a context Γ represent the *generators* of the infinity category generated by Γ. Their type is given by the context.

Types have 2 constructors, and model the source and target operations of globular sets.

Types and Variables

Variables of a context Γ represent the *generators* of the infinity category generated by Γ. Their type is given by the context.

Types have 2 constructors, and model the source and target operations of globular sets.

- The $\star$ constructor takes no arguments.
  A variable of type $\star$ represents a 0-cell, and so has no sources or targets.

Types and Variables

Variables of a context Γ represent the *generators* of the infinity category generated by Γ. Their type is given by the context.

Types have 2 constructors, and model the source and target operations of globular sets.

- The $\star$ constructor takes no arguments.
  A variable of type $\star$ represents a 0-cell, and so has no sources or targets.
- The *arrow* constructor takes 2 terms and a type as arguments.
  A variable of type $s \to_A t$ has source s, target t and lower dimensional sources and targets given by A.

Types and Variables

Variables of a context Γ represent the *generators* of the infinity category generated by Γ. Their type is given by the context.

Types have 2 constructors, and model the source and target operations of globular sets.

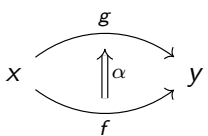
- The $\star$ constructor takes no arguments.
  A variable of type $\star$ represents a 0-cell, and so has no sources or targets.
- The *arrow* constructor takes 2 terms and a type as arguments.
  A variable of type $s \to_A t$ has source s, target t and lower dimensional sources and targets given by A.

Types and Variables

Variables of a context Γ represent the *generators* of the infinity category generated by Γ. Their type is given by the context.

Types have 2 constructors, and model the source and target operations of globular sets.

- The $\star$ constructor takes no arguments.
  A variable of type $\star$ represents a 0-cell, and so has no sources or targets.
- The *arrow* constructor takes 2 terms and a type as arguments.
  A variable of type $s \to_A t$ has source s, target t and lower dimensional sources and targets given by A.

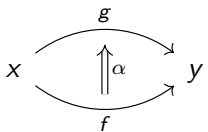


Types and Variables

Variables of a context Γ represent the *generators* of the infinity category generated by Γ. Their type is given by the context.

Types have 2 constructors, and model the source and target operations of globular sets.

- The $\star$ constructor takes no arguments.
  A variable of type $\star$ represents a 0-cell, and so has no sources or targets.
- The *arrow* constructor takes 2 terms and a type as arguments.
  A variable of type $s \to_A t$ has source s, target t and lower dimensional sources and targets given by A.



$$\alpha : f \to_{x \to_\star y} g$$

Contexts and Substitutions

Contexts consist of a list of pairs of variable names and types.

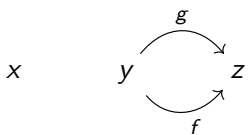
All *pasting diagrams* describe a context.

$$x \xrightarrow{f} y \xrightarrow{g} z$$

gives the context

$$x : \star,$$
$$y : \star,$$
$$f : x \to_\star y,$$
$$z : \star,$$
$$g : y \to_\star z$$

Contexts can represent non-pasting diagrams.



can be represented by the context:

$$x : \star, y : \star, z : \star,$$
$$f : y \to_\star z$$
$$g : y \to_\star z$$

Contexts and Substitutions

Contexts consist of a list of pairs of variable names and types.

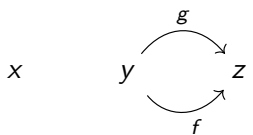
All *pasting diagrams* describe a context.

$$x \xrightarrow{f} y \xrightarrow{g} z$$

gives the context

$$x : \star,$$
$$y : \star,$$
$$f : x \to_\star y,$$
$$z : \star,$$
$$g : y \to_\star z$$

Contexts can represent non-pasting diagrams.



can be represented by the context:

$$x : \star, y : \star, z : \star,$$
$$f : y \to_\star z$$
$$g : y \to_\star z$$

Substitutions $\sigma : \Gamma \to \Delta$ map *variables* of Γ to *terms* of Δ, this operation can be extended to all terms, types, and substitutions.

Disc contexts

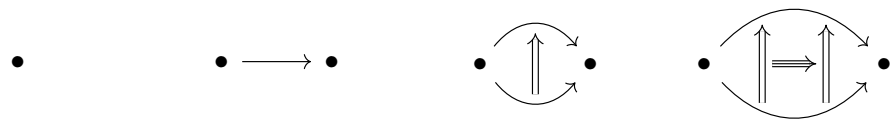
For each natural number we can define the *disc context* D_n.

D_0 D_1 D_2 D_3

Disc contexts

For each natural number we can define the *disc context* D_n.

D_0 D_1 D_2 D_3

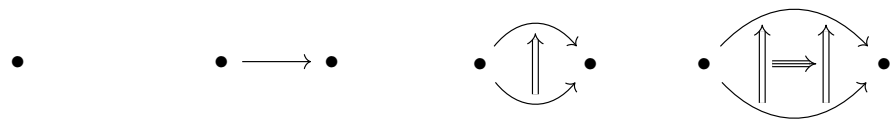


- These are all *pasting diagrams*.

Disc contexts

For each natural number we can define the *disc context* D_n.

D_0 D_1 D_2 D_3

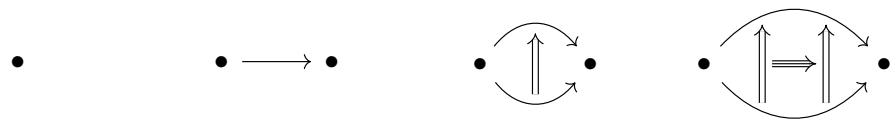


- These are all *pasting diagrams*.
- Their trees are all *linear*.

Disc contexts

For each natural number we can define the *disc context* D_n.

D_0 D_1 D_2 D_3



- These are all *pasting diagrams*.
- Their trees are all *linear*.
- A well-typed substitution from a disc context has the same data as a term.

Weak Composition

At the moment we can model globular sets but have no composition or coherence terms.

Weak Composition

At the moment we can model globular sets but have no composition or coherence terms.

Motto

There is a composite of every pasting diagram and for any two terms s, t over a pasting diagram (with some fullness conditions) there is a term with source s and target t.

Weak Composition

At the moment we can model globular sets but have no composition or coherence terms.

Motto

There is a composite of every pasting diagram and for any two terms s, t over a pasting diagram (with some fullness conditions) there is a term with source s and target t.

Taking the composite of the diagram:

$$\bullet \xrightarrow{f} \bullet \xrightarrow{g} \bullet$$

gives the composite $f \circ g$.

Over the singleton pasting diagram

$$x$$

and taking $s = x$ and $t = x$ we get a term from x to x representing the identity on x.

Coherence term constructor

The coherence term constructor allows us to represent these compositions and coherences. We need the following arguments.

Coherence term constructor

The coherence term constructor allows us to represent these compositions and coherences. We need the following arguments.

- A pasting diagram, which we can represent as a context satisfying a certain typing judgment.

Coherence term constructor

The coherence term constructor allows us to represent these compositions and coherences. We need the following arguments.

- A pasting diagram, which we can represent as a context satisfying a certain typing judgment.
- A pair of terms over the pasting diagram, which can be represented by a type.

Coherence term constructor

The coherence term constructor allows us to represent these compositions and coherences. We need the following arguments.

- A pasting diagram, which we can represent as a context satisfying a certain typing judgment.
- A pair of terms over the pasting diagram, which can be represented by a type.
- A labelling of the pasting diagram with the terms we want to compose/form a coherence over, which can be represented by a substitution.

Coherence term constructor

The coherence term constructor allows us to represent these compositions and coherences. We need the following arguments.

- A pasting diagram, which we can represent as a context satisfying a certain typing judgment.
- A pair of terms over the pasting diagram, which can be represented by a type.
- A labelling of the pasting diagram with the terms we want to compose/form a coherence over, which can be represented by a substitution.

This gives us the coherence constructor

$$\text{coh}\ (\Gamma : A)[\sigma]$$

which takes a pasting context Γ, a type A over Γ and $\sigma : \Gamma \to \Delta$ to form a term in Δ.

Examples

Identity

Let t be a 1 dimensional term. The identity on t is:

$$\text{coh}\ (x \xrightarrow{f} y : f \to_{x \to_\star y} f)[\sigma]$$

where σ maps f to t.

Examples

Identity

Let t be a 1 dimensional term. The identity on t is:

$$\text{coh}\ (x \xrightarrow{f} y : f \to_{x \to_\star y} f)[\sigma]$$

where σ maps f to t.

1-composition

Let $s : x \to_\star y$ and $t : y \to_\star z$ be terms. Their composite is given by:

$$\text{coh}\ (x \xrightarrow{f} y \xrightarrow{g} z : x \to_\star z)[\sigma]$$

where $\sigma(x) = x$, $\sigma(y) = y$, $\sigma(z) = z$, $\sigma(f) = s$, $\sigma(g), = t$.

Examples

Take the context $\Gamma = w \xrightarrow{f} x \xrightarrow{g} y \xrightarrow{h} z$.

The *associator* is given by:

$$\mathrm{coh}\ (\Gamma : (f \circ g) \circ h \to_{w \to_\star z} f \circ (g \circ h))[\mathrm{id}]$$

Strategy

CaTT as we have presented it has no non-trivial equality and no computation.

Strategy

CaTT as we have presented it has no non-trivial equality and no computation.

The idea is to implement a reduction relation that trivialises associatation.

Strategy

CaTT as we have presented it has no non-trivial equality and no computation.

The idea is to implement a reduction relation that trivialises associatation.

By showing this is confluent and terminating, we can create a type theory where both the source target of the associator are the same but retain decidable type checking and equality.

Insertion

Any reduction scheme must act consistently over all compositions and coherences in order to be confluent.

Insertion

Any reduction scheme must act consistently over all compositions and coherences in order to be confluent.

We propose an operation on terms which we call *Insertion*. This collapses certain compound composites into a single composite by "inserting" the inner composite into the outer composite.

1-Associator

is sent to:

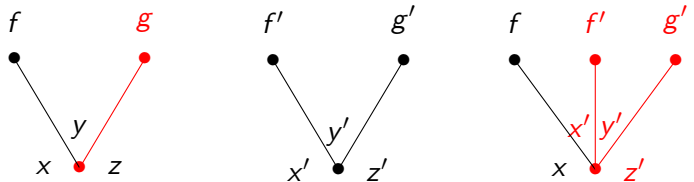
$$x \xrightarrow{f} x' \xrightarrow{f'} y' \xrightarrow{g'} z'$$

Insertion on Trees

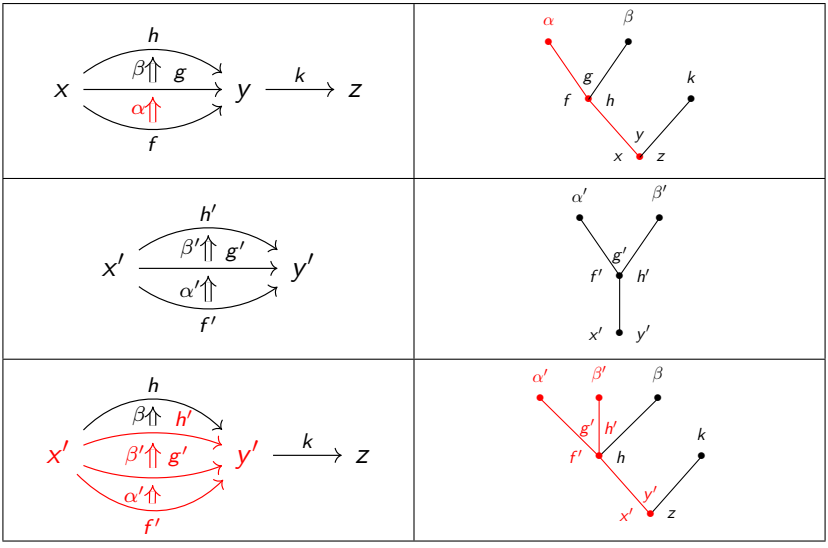
The *insertion* operation on pasting diagrams is more easily described as an operation on trees.

Insertion on Trees

The *insertion* operation on pasting diagrams is more easily described as an operation on trees.



Insertion on Trees



Insertion Rule

Suppose we have a coherence $\mathsf{coh}\ (\Gamma : A)[\sigma]$ where $\sigma(x) \equiv \mathsf{coh}\ (\Delta : B)[\tau]$ for some locally maximal x in Γ. So far we have described the action on pasting diagrams but not on the entire term.

Insertion Rule

Suppose we have a coherence coh $(\Gamma : A)[\sigma]$ where $\sigma(x) \equiv \text{coh } (\Delta : B)[\tau]$ for some locally maximal x in Γ. So far we have described the action on pasting diagrams but not on the entire term.

To make a new coherence term we need:

- A new pasting diagram, which we call $\Gamma \ll_x \Delta$.

Insertion Rule

Suppose we have a coherence coh $(\Gamma : A)[\sigma]$ where $\sigma(x) \equiv \text{coh } (\Delta : B)[\tau]$ for some locally maximal x in Γ. So far we have described the action on pasting diagrams but not on the entire term.

To make a new coherence term we need:

- A new pasting diagram, which we call $\Gamma \ll x\ \Delta$.
- A new type over this pasting diagram. We form a substitution $\kappa : \Gamma \to \Gamma \ll x\ \Delta$ and then use $A[\![\kappa]\!]$.

Insertion Rule

Suppose we have a coherence coh $(\Gamma : A)[\sigma]$ where $\sigma(x) \equiv \text{coh } (\Delta : B)[\tau]$ for some locally maximal x in Γ. So far we have described the action on pasting diagrams but not on the entire term.

To make a new coherence term we need:

- A new pasting diagram, which we call $\Gamma \ll x\ \Delta$.
- A new type over this pasting diagram. We form a substitution $\kappa : \Gamma \to \Gamma \ll x\ \Delta$ and then use $A[\![\kappa]\!]$.
- A new substitution assigning arguments to the new composite, which we call $\sigma \ll x\ \tau$

Insertion Rule

Suppose we have a coherence coh $(\Gamma : A)[\sigma]$ where $\sigma(x) \equiv \text{coh } (\Delta : B)[\tau]$ for some locally maximal x in Γ. So far we have described the action on pasting diagrams but not on the entire term.

To make a new coherence term we need:

- A new pasting diagram, which we call $\Gamma \ll x\ \Delta$.
- A new type over this pasting diagram. We form a substitution $\kappa : \Gamma \to \Gamma \ll x\ \Delta$ and then use $A[\![\kappa]\!]$.
- A new substitution assigning arguments to the new composite, which we call $\sigma \ll x\ \tau$

Insertion Rule

Suppose we have a coherence $\mathsf{coh}\ (\Gamma : A)[\sigma]$ where $\sigma(x) \equiv \mathsf{coh}\ (\Delta : B)[\tau]$ for some locally maximal x in Γ. So far we have described the action on pasting diagrams but not on the entire term.

To make a new coherence term we need:

- A new pasting diagram, which we call $\Gamma \ll x\ \Delta$.
- A new type over this pasting diagram. We form a substitution $\kappa : \Gamma \to \Gamma \ll x\ \Delta$ and then use $A[\![\kappa]\!]$.
- A new substitution assigning arguments to the new composite, which we call $\sigma \ll x\ \tau$

We therefore get the following generator for our reduction:

$$\mathsf{coh}\ (\Gamma : A)[\sigma] \rightsquigarrow \mathsf{coh}\ ((\Gamma \ll x\ \Delta) : (A[\![\kappa]\!]))[(\sigma \ll x\ \tau)]$$

Example

Over the context $\Gamma = w \xrightarrow{f} x \xrightarrow{g} y \xrightarrow{h} z$ we have that:

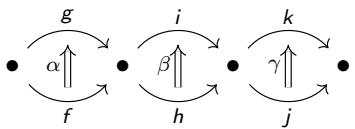
$$f \circ_0 (g \circ_0 h) \rightsquigarrow f \circ_0 g \circ_0 h = \mathrm{coh}\ (\Gamma : w \to_\star z)[\mathrm{id}]$$

Example

Over the context $\Gamma = w \xrightarrow{f} x \xrightarrow{g} y \xrightarrow{h} z$ we have that:

$$f \circ_0 (g \circ_0 h) \rightsquigarrow f \circ_0 g \circ_0 h = \mathrm{coh}\ (\Gamma : w \to_\star z)[\mathrm{id}]$$

Over the context Δ which we let be given by:



We have:

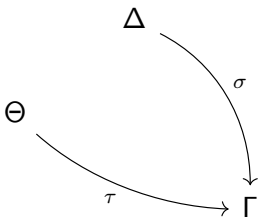
$$\begin{aligned} \alpha \circ_0 (\beta \circ_0 \gamma) &\rightsquigarrow \mathrm{coh}\ (\Delta : f \circ_0 (h \circ_0 j) \to g \circ_0 (i \circ_0 k))[\mathrm{id}] \\ &\rightsquigarrow \mathrm{coh}\ (\Delta : f \circ_0 h \circ_0 j \to g \circ_0 i \circ_0 k)[\mathrm{id}] \end{aligned}$$

Universal Property of Insertion

Insertion also satisfies a *universal property*. Suppose we have coh $(\Delta : A)[\sigma]$ where $\sigma(x) = \text{coh } (\Theta : B)[\tau]$.

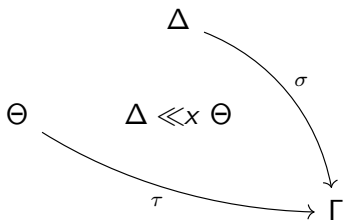
Universal Property of Insertion

Insertion also satisfies a *universal property*. Suppose we have coh $(\Delta : A)[\sigma]$ where $\sigma(x) = \text{coh } (\Theta : B)[\tau]$.



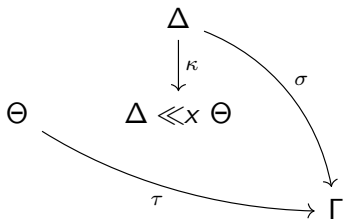
Universal Property of Insertion

Insertion also satisfies a *universal property*. Suppose we have coh $(\Delta : A)[\sigma]$ where $\sigma(x) = \text{coh } (\Theta : B)[\tau]$.



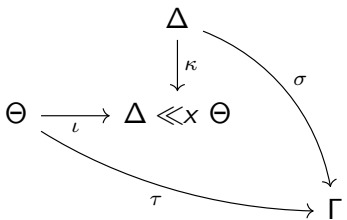
Universal Property of Insertion

Insertion also satisfies a *universal property*. Suppose we have coh $(\Delta : A)[\sigma]$ where $\sigma(x) = \text{coh } (\Theta : B)[\tau]$.



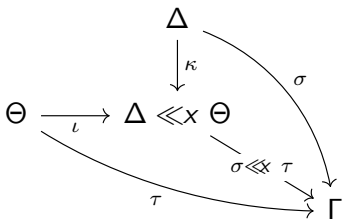
Universal Property of Insertion

Insertion also satisfies a *universal property*. Suppose we have coh $(\Delta : A)[\sigma]$ where $\sigma(x) = \text{coh } (\Theta : B)[\tau]$.



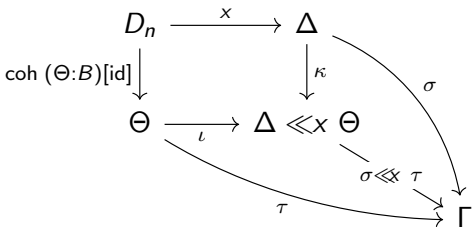
Universal Property of Insertion

Insertion also satisfies a *universal property*. Suppose we have coh $(\Delta : A)[\sigma]$ where $\sigma(x) = \text{coh } (\Theta : B)[\tau]$.



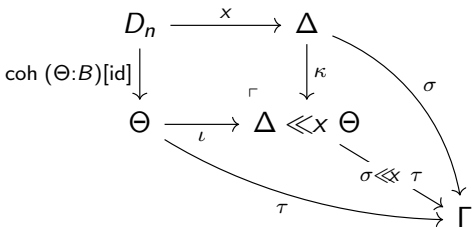
Universal Property of Insertion

Insertion also satisfies a *universal property*. Suppose we have coh $(\Delta : A)[\sigma]$ where $\sigma(x) = \text{coh } (\Theta : B)[\tau]$.



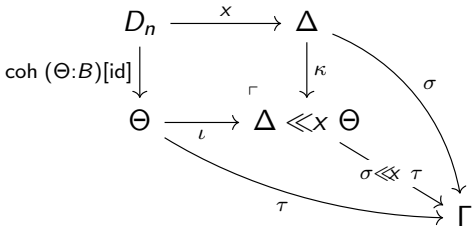
Universal Property of Insertion

Insertion also satisfies a *universal property*. Suppose we have coh $(\Delta : A)[\sigma]$ where $\sigma(x) = \text{coh } (\Theta : B)[\tau]$.



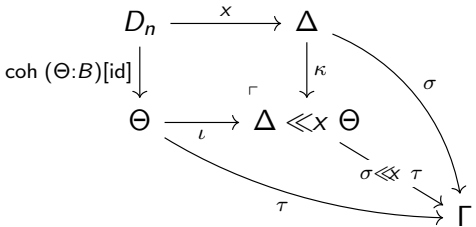
Universal Property of Insertion

Insertion also satisfies a *universal property*. Suppose we have coh $(\Delta : A)[\sigma]$ where $\sigma(x) = \text{coh } (\Theta : B)[\tau]$.



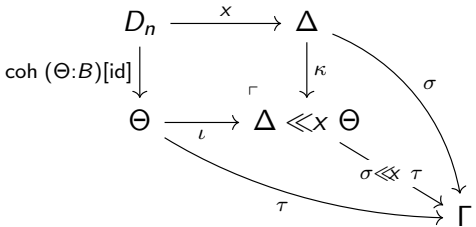
Universal Property of Insertion

Insertion also satisfies a *universal property*. Suppose we have coh $(\Delta : A)[\sigma]$ where $\sigma(x) = \text{coh } (\Theta : B)[\tau]$.



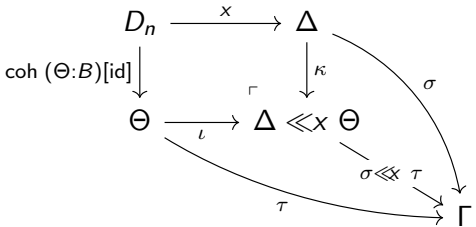
Universal Property of Insertion

Insertion also satisfies a *universal property*. Suppose we have coh $(\Delta : A)[\sigma]$ where $\sigma(x) = \text{coh } (\Theta : B)[\tau]$.



Universal Property of Insertion

Insertion also satisfies a *universal property*. Suppose we have coh $(\Delta : A)[\sigma]$ where $\sigma(x) = \text{coh } (\Theta : B)[\tau]$.



The universal property provides intuition for insertion, and also provides us with another proof technique for working with insertion.

Equality generated from Insertion

Our reduction scheme generates an equality that:

- trivialises all *associativity* equations,
- is *terminating*,
- is *confluent*,
- and has a *decidable* algorithm for type-checking.

Future Work

- Formalise all results in the paper.
- Combine this with the reduction for strict units to get a type theory for strictly unital and associative categories.
- Create a general framework for CaTT based type theories with definitional equality.
- Show that models of the strict versions of CaTT are equivalent to the models of the original version.