A Type Theory for Strictly Associative Infinity Categories

Eric Finster   Alex Rice   Jamie Vicary

22nd October 2021

Outline

1. Strict Infinity Categories
2. CaTT
3. Strict Associators

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

$$\begin{tikzcd} C_1 \arrow[d, shift right, "s"'] \arrow[d, shift left, "t"] \\ C_0 \end{tikzcd}$$

Globular Sets

Globular sets are one natural shape of higher categories:

Ordinary 1-categories

$$\begin{tikzcd} C_1 \arrow[d, shift right, "s"'] \arrow[d, shift left, "t"] \\ C_0 \end{tikzcd}$$

Globular sets

$$\begin{tikzcd} \vdots \\ G_2 \arrow[d, shift right, "s_1"'] \arrow[d, shift left, "t_1"] \\ G_1 \arrow[d, shift right, "s_0"'] \arrow[d, shift left, "t_0"] \\ G_0 \end{tikzcd}$$

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

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

Compostition in Globular Sets

Composition of 1 cells

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

Composition of 2 cells

Composition along a 1-boundary:
$$\begin{tikzcd}
\bullet \arrow[r, bend left=70, ""{name=T}] \arrow[r, ""{name=M}] \arrow[r, bend right=70, ""{name=B}] & \bullet
\arrow[from=M, to=T, phantom, "\beta\Uparrow"]
\arrow[from=B, to=M, phantom, "\alpha\Uparrow"]
\end{tikzcd}$$

Compostition in Globular Sets

Composition of 1 cells

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

Composition of 2 cells

Composition along a 1-boundary:
$$\begin{tikzcd} \bullet \arrow[r, bend left=70] \arrow[r] \arrow[r, bend right=70] \arrow[r, phantom, shift left=3, "\beta\Uparrow"] \arrow[r, phantom, shift right=3, "\alpha\Uparrow"] & \bullet \end{tikzcd}$$

Composition along a 0-boundary:
$$\begin{tikzcd} \bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\alpha\Uparrow"] & \bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\beta\Uparrow"] & \bullet \end{tikzcd}$$

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

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

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

Strict Infinity Categories - Identities

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

$$\mathrm{id}(x) \circ_k f = f = f \circ_k \mathrm{id}(y)$$

Strict Infinity Categories - Interchange

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

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

$$
\begin{array}{c}
\begin{tikzcd}
\bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Uparrow c"] & \bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Uparrow d"] & \bullet
\end{tikzcd} \\
\circ \\
\begin{tikzcd}
\bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Uparrow a"] & \bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Uparrow b"] & \bullet
\end{tikzcd}
\end{array}
=
\begin{tikzcd}
\bullet \arrow[r, bend left=60] \arrow[r] \arrow[r, bend right=60] \arrow[r, phantom, shift left=3, "\Uparrow c"] \arrow[r, phantom, shift right=3, "\Uparrow a"] & \bullet \arrow[r, bend left=60] \arrow[r] \arrow[r, bend right=60] \arrow[r, phantom, shift left=3, "\Uparrow d"] \arrow[r, phantom, shift right=3, "\Uparrow b"] & \bullet
\end{tikzcd}
$$

Strict Infinity Categories - Interchange

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

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

$$\begin{array}{c} \begin{tikzcd} \bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Uparrow c"] & \bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Uparrow d"] & \bullet \end{tikzcd} \\ \circ \\ \begin{tikzcd} \bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Uparrow a"] & \bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Uparrow b"] & \bullet \end{tikzcd} \end{array} = \begin{tikzcd} \bullet \arrow[r, bend left=60] \arrow[r] \arrow[r, bend right=60] \arrow[r, phantom, shift left=2, "\Uparrow c"] \arrow[r, phantom, shift right=2, "\Uparrow a"] & \bullet \arrow[r, bend left=60] \arrow[r] \arrow[r, bend right=60] \arrow[r, phantom, shift left=2, "\Uparrow d"] \arrow[r, phantom, shift right=2, "\Uparrow b"] & \bullet \end{tikzcd}$$

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.

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

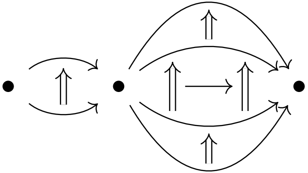
A pasting diagram represents a composition that can be done in an infinity category. More precisely it is an object of the free strict infinity category on the terminal globular set.

The compositions we have already seen form pasting diagrams.

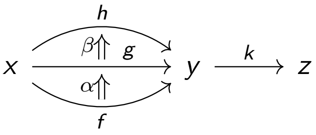
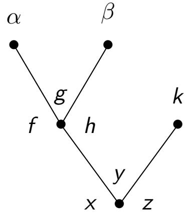
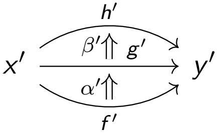
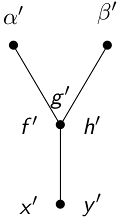
$$\begin{tikzcd} \bullet \arrow[r, "f"] & \bullet \arrow[r, "g"] & \bullet \end{tikzcd}$$

$$\begin{tikzcd} \bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\alpha\Uparrow"] & \bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\beta\Uparrow"] & \bullet \end{tikzcd}$$

We can also form more complicated compositions as pasting diagrams.



Trees

<table>
<tr><th>Pasting diagram</th><th>Batanin tree</th></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td>
</td><td>
</td></tr>
</table>

Weakness

If a category has all products and a terminal object, then it is a monoidal category.

Weakness

If a category has all products and a terminal object, then it is 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 is 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.

Semistrictness

Since full strictification is not possible, we want to do the best possible.
Therefore, we look for semistrict definitions of infinity categories.

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:

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

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

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:

- 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:

- 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

Overview

CaTT is a type theory for weak infinity categories.

Overview

CaTT is a type theory for weak infinity categories. There are 4 pieces of syntax, all defined by mutual induction:

- Contexts: Represent finitely generated infinity categories.

Overview

CaTT is a type theory for weak infinity categories. There are 4 pieces of syntax, all defined by mutual induction:

- Contexts: Represent finitely generated infinity categories.
- Terms: A term in a context Γ represents a cell in the infinity category generated from Γ.

Overview

CaTT is a type theory for weak infinity categories. There are 4 pieces of syntax, all defined by mutual induction:

- Contexts: Represent finitely generated infinity categories.
- Terms: A term in a context Γ represents a cell in the infinity category generated from 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. There are 4 pieces of syntax, all defined by mutual induction:

- Contexts: Represent finitely generated infinity categories.
- Terms: A term in a context Γ represents a cell in the infinity category generated from 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.

Types and Variables

Variables of a context Γ represent the generators of the infinity category generated by Γ.

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.

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.

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.

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 $s \rightarrow_A t$ has source s, target t and lower dimensional sources and targets given by A.

$$
\begin{tikzcd}
x \arrow[r, bend left=50, "g"{name=G}] \arrow[r, bend right=50, "f"'{name=F}] & y
\arrow[from=F, to=G, Rightarrow, "\alpha"']
\end{tikzcd}
$$

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

$$\begin{aligned} &x : \star, \\ &y : \star, \\ &f : x \to_\star y, \\ &z : \star, \\ &g : y \to_\star z \end{aligned}$$

Contexts can represent non-pasting diagrams.

$$\begin{tikzcd} x & y \arrow[r, bend left, "g"] \arrow[r, bend right, "f"'] & z \end{tikzcd}$$

can be represented by the context:

$$\begin{aligned} &x : \star, y : \star, z : \star, \\ &f : y \to_\star z \\ &g : y \to_\star z \end{aligned}$$

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

$$\begin{aligned} &x : \star, \\ &y : \star, \\ &f : x \to_\star y, \\ &z : \star, \\ &g : y \to_\star z \end{aligned}$$

Contexts can represent non-pasting diagrams.

$$\begin{tikzcd} x & y \arrow[r, bend left, "g"] \arrow[r, bend right, "f"'] & z \end{tikzcd}$$

can be represented by the context:

$$\begin{aligned} &x : \star, y : \star, z : \star, \\ &f : y \to_\star z \\ &g : y \to_\star z \end{aligned}$$

Substitutions $\sigma : \Gamma \to \Delta$ map variables of Γ to terms of Δ, this

Disc contexts

We can define disc contexts by mutual induction as follows:

$$\begin{aligned} D_0 &= (d_0^- : \star) & A_0 &= \star \\ D_{n+1} &= D_n, d_n^+ : A_n, d_{n+1} & A_{n+1} &= d_n^- \to_{A_n} d_n^+ \end{aligned}$$

Disc contexts

We can define disc contexts by mutual induction as follows:

$$\begin{aligned} D_0 &= (d_0^- : \star) & A_0 &= \star \\ D_{n+1} &= D_n, d_n^+ : A_n, d_{n+1} & A_{n+1} &= d_n^- \to_{A_n} d_n^+ \end{aligned}$$

These are all pasting diagrams.

Disc contexts

We can define disc contexts by mutual induction as follows:

$$
\begin{aligned}
D_0 &= (d_0^- : \star) & A_0 &= \star \\
D_{n+1} &= D_n, d_n^+ : A_n, d_{n+1} & A_{n+1} &= d_n^- \to_{A_n} d_n^+
\end{aligned}
$$

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

Weak Compostition

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

Weak Compostition

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 Compostition

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.

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.
- 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.
- 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.
- 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 $\mathsf{coh}\ (\Gamma : A)[\sigma]$ 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 term. The identity on t is:

$$\mathrm{coh}\ (D_1 : d_1^- \to_{A_1} d_1^-)[\sigma]$$

where σ maps d_1^- to t.

Examples

Identity

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

$$\mathsf{coh}\ (D_1 : d_1^- \to_{A_1} d_1^-)[\sigma]$$

where σ maps d_1^- to t.

1-composition

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

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

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

Examples

Take the context
$\Gamma = w : \star, x : \star, f : w \to_\star x, y : \star, g : x \to_\star y, z : \star, h : y \to_\star z.$
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

$$\begin{tikzcd} x \arrow[r, "f"] & y \arrow[r, "g"] & z & & \color{red}{x'} \arrow[r, red, "f'"] & \color{red}{y'} \arrow[r, red, "g'"] \arrow[llll, Rightarrow, bend left] & \color{red}{z'} \end{tikzcd}$$

is sent to:

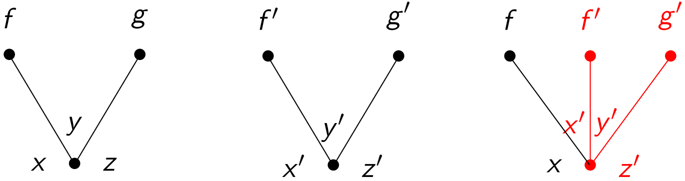
$$\begin{tikzcd} x \arrow[r, "f"] & \color{red}{x'} \arrow[r, red, "f'"] & \color{red}{y'} \arrow[r, red, "g'"] & \color{red}{z'} \end{tikzcd}$$

Insertion on Trees

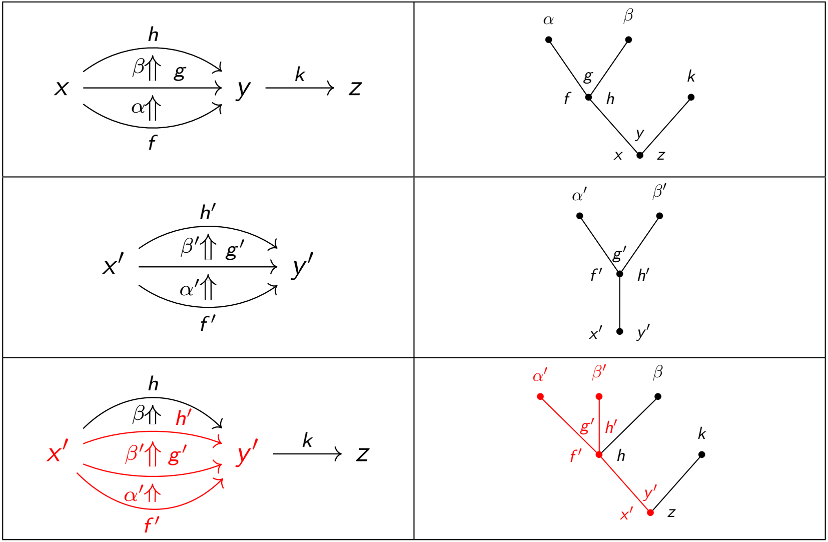
The insertion operation on pasting diagrams is more easily described as an operation on trees.

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

Insertion Rule

Suppose we have a coherence coh $(\Gamma : A)[\sigma]$ where $\sigma(x) \equiv \mathrm{coh}\ (\Delta : B)[\tau]$ 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 $\sigma(x) \equiv \mathrm{coh}\ (\Delta : B)[\tau]$ 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 $\sigma \ll_x \tau$

We therefore get the following generator for our equality relation:

$$\text{coh } (\Gamma : A)[\sigma] = \text{coh } ((\Gamma \ll_x \Delta) : (A[\![\kappa]\!]))[(\sigma \ll_x \tau)]$$

Example

Take the term $a \circ (b \circ c)$. Written in full this is:

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

with $\sigma(f) = a$ and:

$$\sigma(g) = \mathsf{coh}\ (x' \xrightarrow{f'} y' \xrightarrow{g'} z' : x' \to_\star z')[\tau]$$

with $\tau(f') = b$ and $\tau(g') = c$.

Example

Take the term $a \circ (b \circ c)$. Written in full this is:

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

with $\sigma(f) = a$ and:

$$\sigma(g) = \mathsf{coh}\ (x' \xrightarrow{f'} y' \xrightarrow{g'} z' : x' \to_\star z')[\tau]$$

with $\tau(f') = b$ and $\tau(g') = c$.
In this case the inserted context is $x \xrightarrow{a} x' \xrightarrow{a'} y' \xrightarrow{b'} z'$ with $\kappa(a) = a$ and $\kappa(b) = a' \circ b'$ which gives us final term:

$$\mathsf{coh}\ (x \xrightarrow{a} x' \xrightarrow{a'} y' \xrightarrow{b'} z' : x \to_\star z')[\sigma \ll_x \tau]$$

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 $\sigma(x) = \text{coh } (\Theta : B)[\tau]$.

$$\begin{tikzcd}
 & \Delta \arrow[ddr, bend left, "\sigma"] & \\
\Theta \arrow[drr, bend right, "\tau"'] & & \\
 & & \Gamma
\end{tikzcd}$$

Universal Property of Insertion

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

$$\begin{tikzcd}
 & \Delta \arrow[ddr, bend left, "\sigma"] & \\
\Theta \arrow[drr, bend right, "\tau"'] & \Delta \ll x\ \Theta & \\
 & & \Gamma
\end{tikzcd}$$

Universal Property of Insertion

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

$$\begin{tikzcd}
 & \Delta \arrow[d, "\kappa"] \arrow[dr, bend left, "\sigma"] & \\
\Theta \arrow[rr, bend right, "\tau"'] & \Delta \ll_x \Theta & \\
 & & \Gamma
\end{tikzcd}$$

Universal Property of Insertion

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

$$\begin{tikzcd}
 & \Delta \arrow[d, "\kappa"] \arrow[ddr, bend left, "\sigma"] & \\
\Theta \arrow[r, "\iota"'] \arrow[drr, bend right, "\tau"] & \Delta \ll_x \Theta & \\
 & & \Gamma
\end{tikzcd}$$

Universal Property of Insertion

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

$$
\begin{tikzcd}
 & \Delta \arrow[d, "\kappa"] \arrow[ddr, bend left, "\sigma"] & \\
\Theta \arrow[r, "\iota"'] \arrow[drr, bend right, "\tau"'] & \Delta \ll_x \Theta \arrow[dr, "\sigma \ll_x \tau"] & \\
 & & \Gamma
\end{tikzcd}
$$

Universal Property of Insertion

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

$$\begin{tikzcd}
D_n \arrow[r, "x"] \arrow[d, "\mathsf{coh}\ (\Theta : B)[\mathrm{id}]"'] & \Delta \arrow[d, "\kappa"] \arrow[ddr, bend left, "\sigma"] & \\
\Theta \arrow[r, "\iota"'] \arrow[drr, bend right, "\tau"'] & \Delta \ll_x \Theta \arrow[dr, "\sigma \ll_x \tau"] & \\
 & & \Gamma
\end{tikzcd}$$

Universal Property of Insertion

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

$$\begin{tikzcd}
D_n \arrow[r, "x"] \arrow[d, "\text{coh } (\Theta : B)[\text{id}]"'] \arrow[dr, phantom, "\ulcorner", very near end] & \Delta \arrow[d, "\kappa"] \arrow[ddr, bend left, "\sigma"] & \\
\Theta \arrow[r, "\iota"'] \arrow[drr, bend right, "\tau"'] & \Delta \ll_x \Theta \arrow[dr, dashed, "\sigma \ll_x \tau"] & \\
 & & \Gamma
\end{tikzcd}$$

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.