A Type Theory for Strictly Associative $\infty$-Categories

Alex Rice    Eric Finster    Jamie Vicary

SYCO 10

UNIVERSITY OF CAMBRIDGE
1. Weak Globular Infinity Categories

2. Type Theories for Infinity Categories

3. Strict Associators
Globular sets are one natural shape of higher categories.
*Globular sets* are one natural shape of higher categories. They contain:

- A set of objects or 0-cells $G$. 

Globular sets are one natural shape of higher categories. They contain:

- A set of objects or 0-cells $G$.
- For each pair of objects $x, y \in G$, a set of arrows or 1-cells with source $x$ and target $y$. 
**Globular Sets**

*Globular sets* are one natural shape of higher categories. They contain:

- A set of objects or 0-cells $G$.
- For each pair of objects $x, y \in G$, a set of arrows or 1-cells with source $x$ and target $y$.
- For each pair of parallel arrows $f, g$, a set of 2-arrows (or 2-cells) from $f$ to $g$.

![Diagram of a globular set](attachment:globular_set_diagram.png)
Globular sets are one natural shape of higher categories. They contain:

- A set of objects or 0-cells $G$.
- For each pair of objects $x, y \in G$, a set of arrows or 1-cells with source $x$ and target $y$.
- For each pair of parallel arrows $f, g$, a set of 2-arrows (or 2-cells) from $f$ to $g$. 

...
Composition in Globular Sets

Composition of 1 cells

\[ \bullet \xrightarrow{f} \bullet \xrightarrow{g} \bullet \]
Composition in Globular Sets

Composition of 1 cells

Composition of 2 cells

Composition along a 1-boundary:
Composition in Globular Sets

Composition of 1 cells

\[ \bullet \xrightarrow{f} \bullet \xrightarrow{g} \bullet \]

Composition of 2 cells

Composition along a 1-boundary:

\[ \bullet \xrightarrow{\alpha} \bullet \xrightarrow{\beta} \bullet \]

Composition along a 0-boundary:

\[ \bullet \xrightarrow{\alpha} \bullet \xrightarrow{\beta} \bullet \]
Weak Infinity Categories

In strict category theory, we add equalities between certain arrows.

In higher category theory we can instead require that equivalences exist between certain arrows.
In strict category theory, we add equalities between certain arrows.

In higher category theory we can instead require that equivalences exist between certain arrows.

Coherence

- For a 1-cell $f : x \to y$, there are unitors $\lambda_f : \text{id}_x \circ f \to f$ and $\rho_f : f \circ \text{id}_y$.
- $\lambda_{\text{id}_x}$ and $\rho_{\text{id}_x}$ are both arrows $\text{id}_x \circ \text{id}_x \to \text{id}_x$.
- These should be equivalent.
Strict categories are easier to work with while there are more examples of weak categories.
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.
- *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.
Since full strictification is not possible, we want to do the best possible.
Since full strictification is not possible, we want to do the best possible.

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

<table>
<thead>
<tr>
<th></th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Strict ∞-Cat</strong></td>
<td></td>
</tr>
<tr>
<td>Associators</td>
<td>✓</td>
</tr>
<tr>
<td>Unitsors</td>
<td>✓</td>
</tr>
<tr>
<td>Interchangers</td>
<td>✓</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th></th>
<th>Strict $\infty$-Cat</th>
<th>Simpson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associators</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Unitors</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Interchangers</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Since full strictification is not possible, we want to do the best possible.

Therefore, we look for \textit{semistrict} definitions of infinity categories.

We can strictify:

<table>
<thead>
<tr>
<th></th>
<th>Strict (\infty\text{-Cat})</th>
<th>Simpson</th>
<th>Grey</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associators</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Unitors</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Interchangers</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Since full strictification is not possible, we want to do the best possible.

Therefore, we look for \textit{semistrict} definitions of infinity categories.

We can strictify:

<table>
<thead>
<tr>
<th></th>
<th>Strict $\infty$-\textbf{Cat}</th>
<th>Simpson</th>
<th>Grey</th>
<th>CaTT$_{su}$ $^1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associators</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Unitors</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Interchangers</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$^1$Finster, Reutter, et al., \textit{A Type Theory for Strictly Unital $\infty$-Categories}
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:

<table>
<thead>
<tr>
<th></th>
<th>Strict $\infty$-Cat</th>
<th>Simpson</th>
<th>Grey</th>
<th>CaTT$_{su}$</th>
<th>CaTT$_{sa}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Associators</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Unitors</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Interchangers</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

1. Finster, Reutter, et al., *A Type Theory for Strictly Unital $\infty$-Categories*
2. Finster, R., and Vicary, *A Type Theory for Strictly Associative Infinity Categories*
CaTT is a type theory for weak infinity categories\(^3\).

\(^3\)Finster and Mimram, *A Type-Theoretical Definition of Weak \(\omega\)-Categories.*
CaTT is a type theory for *weak infinity categories*\(^3\).

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

- **Contexts**: Generating data of an infinity category.

---

\(^3\)Finster and Mimram, *A Type-Theoretical Definition of Weak \(\omega\)-Categories.*
CaTT is a type theory for \textit{weak infinity categories}\textsuperscript{3}.

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

- \textbf{Contexts}: Generating data of an infinity category.
- \textbf{Terms}: Operations in an infinity category.

\textsuperscript{3}Finster and Mimram, \textit{A Type-Theoretical Definition of Weak $\omega$-Categories}. 
CaTT is a type theory for *weak infinity categories*\(^3\).

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

- **Contexts**: Generating data of an infinity category.
- **Terms**: Operations in an infinity category.
- **Types**: Source and Target for a term.

---

\(^3\)Finster and Mimram, *A Type-Theoretical Definition of Weak \(\omega\)-Categories.*
CaTT is a type theory for *weak infinity categories*\(^3\).

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

- **Contexts**: Generating data of an infinity category.
- **Terms**: Operations in an infinity category.
- **Types**: Source and Target for a term.
- **Substitutions**: A mapping from variables of one context to terms of another.

\(^3\)Finster and Mimram, *A Type-Theoretical Definition of Weak ω-Categories*. 
Types in CaTT have 2 constructors.

\[ \text{The } \star \text{ constructor takes no arguments. A term of type } \star \text{ represents a 0-cell.} \]

\[ \text{The } \text{arrow} \text{ constructor takes 2 terms and a type as arguments. A term of type } s \rightarrow A \text{ has source } s, \text{ target } t \text{ and lower dimensional sources and targets given by } A. \]
Types in CaTT have 2 constructors.

- The \( \star \) constructor takes no arguments.
  
  A term of type \( \star \) represents a 0-cell.
Types in CaTT have 2 constructors.

- The $\star$ constructor takes no arguments.  
  A term of type $\star$ represents a 0-cell.

- The \textit{arrow} constructor takes 2 terms and a type as arguments.  
  A term of type $s \to_A t$ has source $s$, target $t$ and lower dimensional sources and targets given by $A$. 

Types in CaTT have 2 constructors.

- The $\star$ constructor takes no arguments.
  A term of type $\star$ represents a 0-cell.

- The *arrow* constructor takes 2 terms and a type as arguments.
  A term of type $s \rightarrow_A t$ has source $s$, target $t$ and lower dimensional sources and targets given by $A$. 
Types in CaTT have 2 constructors.

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

\[
x \xrightarrow{\alpha} y \xleftarrow{\beta}
\]
Types in CaTT have 2 constructors.

- The \( \star \) constructor takes no arguments.
  A term of type \( \star \) represents a 0-cell.

- The *arrow* constructor takes 2 terms and a type as arguments.
  A term of type \( s \rightarrow^A t \) has source \( s \), target \( t \) and lower dimensional sources and targets given by \( A \).

\[\begin{align*}
\alpha : f & \rightarrow_{x \rightarrow \star y} g
\end{align*}\]
Contexts consist of a list of pairs of variable names and types.
Contexts consist of a list of pairs of variable names and types.

**Disc contexts**

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

\[
D_0 \quad D_1 \quad D_2 \quad D_3
\]

\[
D_2 := x : *, y : *, f : x \to^* y, g : x \to^* y, \alpha : f \to_{x \to^* y} g
\]
Composition can be done with the coh constructor.

**coh constructor**

Given:
- A context $\Gamma$ - the shape of the composition,
- A type $A$ in $\Gamma$ - the boundary of the composition,
- A substitution $\sigma : \Gamma \rightarrow \Delta$ - the terms to be composed,

we get a term in $\Delta$:

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

The contexts for which the coh constructor is well typed are called *pasting contexts*.
Suppose we have:

\[
\begin{array}{c}
\bullet \xrightarrow{f} \bullet \xrightarrow{g} \bullet \xrightarrow{h} \bullet
\end{array}
\]

Let \( \Gamma = \bullet \xrightarrow{a} \bullet \xrightarrow{b} \bullet \). \( \Gamma \) is a pasting context. Then:

\[
(f \cdot g) \cdot h = coh (\Gamma : x \rightarrow z)[a \mapsto f, b \mapsto g] (f \cdot g) \cdot h = coh (\Gamma : x \rightarrow z)[a \mapsto f \cdot g, b \mapsto h]
\]
Example composition

Suppose we have:

\[ \bullet \xrightarrow{f} \bullet \xrightarrow{g} \bullet \xrightarrow{h} \bullet \]

Let \( \Gamma = \bullet \xrightarrow{a} \bullet \xrightarrow{b} \bullet \). \( \Gamma \) is a pasting context. Then:

\[
\begin{align*}
  f \cdot g &:= \text{coh} (\Gamma : x \to z)[a \mapsto f, \\
             &\quad b \mapsto g]
\end{align*}
\]
Example composition

Suppose we have:

\[ \bullet \xrightarrow{f} \bullet \xrightarrow{g} \bullet \xrightarrow{h} \bullet \]

Let \( \Gamma = \bullet \xrightarrow{a} \bullet \xrightarrow{b} \bullet \). \( \Gamma \) is a pasting context. Then:

\[
f \cdot g := \text{coh} (\Gamma : x \rightarrow z)[a \mapsto f, \quad b \mapsto g]
\]

\[
(f \cdot g) \cdot h := \text{coh} (\Gamma : x \rightarrow z)[a \mapsto f \cdot g, \quad b \mapsto h]
\]
CaTT as we have presented it has no non-trivial equality and no computation.

The idea is to implement a reduction relation that unifies the operations we want to strictify.

By doing this we obtain a type theory for which the models are semistrict categories.
CaTT$_{sa}$ has a definitional equality based on an operation we call insertion.

1-associator

\[
\begin{array}{c}
\text{x} \xrightarrow{f} \text{y} \xrightarrow{g} \text{z} \\
\text{x}' \xrightarrow{f'} \text{y}' \xrightarrow{g'} \text{z}'
\end{array}
\]

is sent to:

\[
\begin{array}{c}
\text{x} \xrightarrow{f} \text{x}' \xrightarrow{f'} \text{y}' \xrightarrow{g'} \text{z}'
\end{array}
\]
Components of insertion

\[ \Delta = x \xrightarrow{g} y \xrightarrow{k} z \]

\[ \Theta = x' \xrightarrow{g'} y' \]

Given \( \sigma : \Delta \rightarrow \Gamma \) and \( \tau : \Theta \rightarrow \Gamma \) we get:

\( \sigma \ll \alpha \tau : \Delta \ll \alpha \Theta \rightarrow \Gamma \)
Components of insertion

$$\Delta = x \xrightarrow{\beta \uparrow} g \xrightarrow{\alpha \uparrow} y \xrightarrow{k} z$$

$$\Theta = x' \xrightarrow{\beta' \uparrow} g' \xrightarrow{\alpha' \uparrow} y'$$

$$\Delta \ll\alpha\Theta = x' \xrightarrow{\beta' \uparrow} h' \xrightarrow{\alpha' \uparrow} g' \xrightarrow{f'} y' \xrightarrow{k} z$$
Components of insertion

\[ \Delta = x \xrightarrow{\beta \uparrow} g \xrightarrow{\alpha \uparrow} y \xrightarrow{k} z \]

\[ \Theta = x' \xrightarrow{\beta' \uparrow} g' \xrightarrow{\alpha' \uparrow} y' \]

\[ \Delta \ll \alpha \Theta = x' \xrightarrow{\beta' \uparrow} g' \xrightarrow{\alpha' \uparrow} y' \xrightarrow{k} z \]

\[ \iota : \Theta \rightarrow \Delta \ll \alpha \Theta \]
Components of insertion

\[ \Delta = x \xrightarrow{\alpha} y \xrightarrow{k} z \]

\[ \Theta = x' \xrightarrow{\alpha'} y' \xrightarrow{k} z \]

\[ \Delta \ll_{\alpha} \Theta = x' \xrightarrow{\beta'} y' \xrightarrow{h} z \]

\[ \iota : \Theta \rightarrow \Delta \ll_{\alpha} \Theta \]

\[ \kappa : \Delta \rightarrow \Delta \ll_{\alpha} \Theta \]

Given \( \sigma : \Delta \rightarrow \Gamma \) and \( \tau : \Theta \rightarrow \Gamma \) we get:

\[ \sigma \ll_{\alpha} \tau : \Delta \ll_{\alpha} \Theta \rightarrow \Gamma \]
Components of insertion

\[ \Delta = x \xrightarrow{\beta \uparrow} y \xrightarrow{k} z \]

\[ \Theta = x' \xrightarrow{\beta' \uparrow} y' \]

\[ \Delta \ll_{\alpha} \Theta = x' \xrightarrow{\beta' \uparrow} y' \xrightarrow{k} z \]

\[ \iota : \Theta \to \Delta \ll_{\alpha} \Theta \]

\[ \kappa : \Delta \to \Delta \ll_{\alpha} \Theta \]

Given \( \sigma : \Delta \to \Gamma \) and \( \tau : \Theta \to \Gamma \) we get:

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

![Diagram]

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

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

\[
\begin{array}{ccc}
D_n & \xrightarrow{\alpha} & \Delta \\
| & \downarrow{\gamma} & \downarrow{\kappa} \\
\Theta & \xrightarrow{\iota} & \Delta \ll \alpha \Theta \\
| & \downarrow{\tau} & \downarrow{\sigma} \\
& \Theta & \Gamma
\end{array}
\]
Insertion generates a reduction relation for $\text{Catt}_{sa}$:

$$\text{coh} \ (\Delta : A)[\sigma] \rightsquigarrow \text{coh} \ (\Delta \ll_{\alpha} \Theta : A[\kappa])[\sigma \ll_{\alpha} \tau]$$

where $\sigma(\alpha) = \text{coh} \ (\Delta : B)[\tau]$. 
Insertion generates a reduction relation for $\text{Catt}_{sa}$:

$$\text{coh} \left( \Delta : A \right)[\sigma] \rightsquigarrow \text{coh} \left( \Delta \lll \alpha \Theta : A[\kappa] \right)[\sigma \lll \alpha \tau]$$

where $\sigma(\alpha) = \text{coh} \left( \Delta : B \right)[\tau]$.

This reduction has been proven to have the following properties:

- Subject reduction
- Termination
- Confluence
