A Type Theory for Strictly Associative ∞-Categories

Alex Rice Eric Finster Jamie Vicary

SYCO 10

UNIVERSITY OF CAMBRIDGE

Outline

1. Weak Globular Infinity Categories

2. Type Theories for Infinity Categories

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. They contain:

- A set of objects or 0-cells G.

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.

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.

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

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.

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

- . . .

Composition in Globular Sets

Composition of 1 cells

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

Composition in Globular Sets

Composition of 1 cells

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

Composition of 2 cells

Composition along a 1-boundary:

$$\begin{tikzcd}\bullet \arrow[r, bend left=70] \arrow[r] \arrow[r, bend right=70] \arrow[r, phantom, shift left=3ex, "\beta\Uparrow"] \arrow[r, phantom, shift right=3ex, "\alpha\Uparrow"] & \bullet\end{tikzcd}$$

Composition in Globular Sets

Composition of 1 cells

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

Composition of 2 cells

Composition along a 1-boundary:
$$\begin{tikzcd}
\bullet \arrow[r, bend left=70, ""{name=U, below}] \arrow[r, ""{name=M, above}, ""{name=N, below}] \arrow[r, bend right=70, ""{name=D, above}] & \bullet
\arrow[Rightarrow, from=N, to=U, "\beta"]
\arrow[Rightarrow, from=D, to=M, "\alpha"]
\end{tikzcd}$$

Composition along a 0-boundary:
$$\begin{tikzcd}
\bullet \arrow[r, bend left=60, ""{name=U1, below}] \arrow[r, bend right=60, ""{name=D1, above}] & \bullet \arrow[r, bend left=60, ""{name=U2, below}] \arrow[r, bend right=60, ""{name=D2, above}] & \bullet
\arrow[Rightarrow, from=D1, to=U1, "\alpha"]
\arrow[Rightarrow, from=D2, to=U2, "\beta"]
\end{tikzcd}$$

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.

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.

Coherence

- For a 1-cell $f : x \to y$, there are unitors $\lambda_f : \mathrm{id}_x \circ f \to f$ and $\rho_f : f \circ \mathrm{id}_y$.
- λ_{id_x} and ρ_{id_x} are both arrows $\mathrm{id}_x \circ \mathrm{id}_x \to \mathrm{id}_x$.
- These should be equivalent.

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:

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	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	$\mathsf{CaTT}_{\mathsf{su}}$[1]
Associators	✓	✓	✓	
Unitors	✓		✓	✓
Interchangers	✓	✓		

---

[1] Finster, Reutter, et al., *A Type Theory for Strictly Unital ∞-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:

	Strict ∞-**Cat**	Simpson	Grey	$\mathsf{CaTT}_{\mathsf{su}}$[1]	$\mathsf{CaTT}_{\mathsf{sa}}$[2]
Associators	✓	✓	✓		✓
Unitors	✓		✓	✓	
Interchangers	✓	✓			

---

[1] Finster, Reutter, et al., *A Type Theory for Strictly Unital ∞-Categories*

[2] Finster, R., and Vicary, *A Type Theory for Strictly Associative Infinity Categories*

CaTT

CaTT is a type theory for *weak infinity categories*[3].

---

[3] Finster and Mimram, *A Type-Theoretical Definition of Weak ω-Categories*.

CaTT

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 ω-Categories*.

CaTT

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.

---

[3] Finster and Mimram, *A Type-Theoretical Definition of Weak ω-Categories*.

CaTT

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 ω-Categories*.

CaTT

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

Types in CaTT have 2 constructors.

Types in CaTT

Types in CaTT have 2 constructors.

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

Types in CaTT

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.

Types in CaTT

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.

Types in CaTT

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.

$$\begin{tikzcd} x \arrow[r, bend left=40, "g"{name=G}] \arrow[r, bend right=40, "f"'{name=F}] & y \arrow[from=F, to=G, Rightarrow, "\alpha"'] \end{tikzcd}$$

Types in CaTT

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.

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

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

Contexts

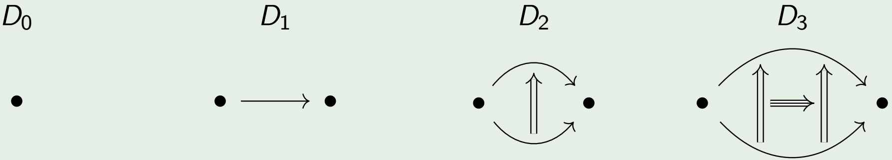
Contexts consist of a list of pairs of variable names and types.

Contexts

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.



$$D_2 := x : \star, y : \star, f : x \to_\star y, g : x \to_\star y, \alpha : f \to_{x \to_\star y} g$$

Composition in CaTT

Composition can be done with the coh constructor.

coh constructor

Given:

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

we get a term in Δ:

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

The contexts for which the coh constructor is well typed are called *pasting contexts*

Example composition

Suppose we have:

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

Example composition

Suppose we have:

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

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

$$\begin{aligned} f \cdot g := \mathsf{coh}\ (\Gamma : x \to z)[a \mapsto f, \\ b \mapsto g] \end{aligned}$$

Example composition

Suppose we have:

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

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

$$\begin{aligned} f \cdot g := \mathsf{coh}\ (\Gamma : x \to z)[a \mapsto f, \\ b \mapsto g] \end{aligned}$$

$$\begin{aligned} (f \cdot g) \cdot h := \mathsf{coh}\ (\Gamma : x \to z)[a \mapsto f \cdot g, \\ b \mapsto h] \end{aligned}$$

Type theories for semistrict languages

- 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.

Insertion

$\mathrm{CaTT}_{\mathrm{sa}}$ has a definitional equality based on an operation we call insertion.

1-associator

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

is sent to:

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

Components of insertion

$$\Delta = \begin{tikzcd} x \arrow[r, bend left=50, "h"] \arrow[r, "g"] \arrow[r, bend right=50, "f"'] \arrow[r, phantom, shift left=2.5ex, "\beta\Uparrow"] \arrow[r, phantom, shift right=2.5ex, "\alpha\Uparrow"] & y \arrow[r, "k"] & z \end{tikzcd}$$

$$\Theta = \begin{tikzcd} x' \arrow[r, bend left=50, "h'"] \arrow[r, "g'"] \arrow[r, bend right=50, "f'"'] \arrow[r, phantom, shift left=2.5ex, "\beta'\Uparrow"] \arrow[r, phantom, shift right=2.5ex, "\alpha'\Uparrow"] & y' \end{tikzcd}$$

Components of insertion

$$\Delta = \begin{tikzcd} x \arrow[r, bend left=60, "h"] \arrow[r, "g"] \arrow[r, bend right=60, "f"'] \arrow[r, phantom, bend left=30, "\beta\Uparrow"] \arrow[r, phantom, bend right=30, "\alpha\Uparrow"] & y \arrow[r, "k"] & z \end{tikzcd} \qquad \Theta = \begin{tikzcd} x' \arrow[r, bend left=60, "h'"] \arrow[r, "g'"] \arrow[r, bend right=60, "f'"'] \arrow[r, phantom, bend left=30, "\beta'\Uparrow"] \arrow[r, phantom, bend right=30, "\alpha'\Uparrow"] & y' \end{tikzcd}$$

$$\Delta \ll\alpha\ \Theta = \begin{tikzcd} x' \arrow[r, bend left=70, "h"] \arrow[r, bend left=20, "h'"] \arrow[r, bend right=20, "g'"] \arrow[r, bend right=70, "f'"'] \arrow[r, phantom, bend left=45, "\beta\Uparrow"] \arrow[r, phantom, "\beta'\Uparrow"] \arrow[r, phantom, bend right=45, "\alpha'\Uparrow"] & y' \arrow[r, "k"] & z \end{tikzcd}$$

Components of insertion

$$\begin{tikzcd}[column sep=large] \Delta = x \arrow[r, bend left=60, "h"] \arrow[r, "g"] \arrow[r, bend right=60, "f"'] \arrow[r, phantom, shift left=3, "\beta\Uparrow"] \arrow[r, phantom, shift right=3, "\alpha\Uparrow"] & y \arrow[r, "k"] & z \end{tikzcd}$$

$$\begin{tikzcd}[column sep=large] \Theta = x' \arrow[r, bend left=60, "h'"] \arrow[r, "g'"] \arrow[r, bend right=60, "f'"'] \arrow[r, phantom, shift left=3, "\beta'\Uparrow"] \arrow[r, phantom, shift right=3, "\alpha'\Uparrow"] & y' \end{tikzcd}$$

$$\begin{tikzcd}[column sep=large] \Delta \lll\alpha\ \Theta = x' \arrow[r, bend left=70, "h"] \arrow[r, bend left=20, "h'"] \arrow[r, bend right=20, "g'"] \arrow[r, bend right=70, "f'"'] \arrow[r, phantom, shift left=5, "\beta\Uparrow"] \arrow[r, phantom, "\beta'\Uparrow"] \arrow[r, phantom, shift right=5, "\alpha'\Uparrow"] & y' \arrow[r, "k"] & z \end{tikzcd}$$

$$\iota : \Theta \to \Delta \lll\alpha\ \Theta$$

Components of insertion

$$\Delta = \begin{tikzcd} x \arrow[r, bend left=50, "h"{name=H}] \arrow[r, "g"{near end, name=G}] \arrow[r, bend right=50, "f"'{name=F}] \arrow[from=G, to=H, phantom, "\beta\Uparrow"] \arrow[from=F, to=G, phantom, "\alpha\Uparrow"] & y \arrow[r, "k"] & z \end{tikzcd}$$

$$\Theta = \begin{tikzcd} x' \arrow[r, bend left=50, "h'"{name=H}] \arrow[r, "g'"{near end, name=G}] \arrow[r, bend right=50, "f'"'{name=F}] \arrow[from=G, to=H, phantom, "\beta'\Uparrow"] \arrow[from=F, to=G, phantom, "\alpha'\Uparrow"] & y' \end{tikzcd}$$

$$\Delta \ll\alpha\ \Theta = \begin{tikzcd} x' \arrow[r, bend left=60, "h"{name=H}] \arrow[r, bend left=20, "h'"{near end, name=H'}] \arrow[r, bend right=20, "g'"{near end, name=G'}] \arrow[r, bend right=60, "f'"'{name=F'}] \arrow[from=H', to=H, phantom, "\beta\Uparrow"] \arrow[from=G', to=H', phantom, "\beta'\Uparrow"] \arrow[from=F', to=G', phantom, "\alpha'\Uparrow"] & y' \arrow[r, "k"] & z \end{tikzcd}$$

$$\iota : \Theta \to \Delta \ll\alpha\ \Theta$$

$$\kappa : \Delta \to \Delta \ll\alpha\ \Theta$$

Components of insertion

$$\Delta = \begin{tikzcd} x \arrow[r, bend left=50, "h"] \arrow[r, "g"] \arrow[r, bend right=50, "f"'] \arrow[r, phantom, shift left=3, "\beta\Uparrow"] \arrow[r, phantom, shift right=3, red, "\alpha\Uparrow"] & y \arrow[r, "k"] & z \end{tikzcd} \qquad \Theta = \begin{tikzcd}[red] x' \arrow[r, bend left=50, "h'"] \arrow[r, "g'"] \arrow[r, bend right=50, "f'"'] \arrow[r, phantom, shift left=3, "\beta'\Uparrow"] \arrow[r, phantom, shift right=3, "\alpha'\Uparrow"] & y' \end{tikzcd}$$

$$\Delta \ll_\alpha \Theta = \begin{tikzcd} \textcolor{red}{x'} \arrow[r, bend left=70, "h"] \arrow[r, red, bend left=25, "h'"] \arrow[r, red, bend right=10, "g'"] \arrow[r, red, bend right=60, "f'"'] \arrow[r, phantom, shift left=5, "\beta\Uparrow"] \arrow[r, phantom, red, shift left=1, "\beta'\Uparrow"] \arrow[r, phantom, red, shift right=4, "\alpha'\Uparrow"] & \textcolor{red}{y'} \arrow[r, "k"] & z \end{tikzcd}$$

$$\iota : \Theta \to \Delta \ll_\alpha \Theta$$

$$\kappa : \Delta \to \Delta \ll_\alpha \Theta$$

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

$$\sigma \ll_\alpha \tau : \Delta \ll_\alpha \Theta \to \Gamma$$

Universal Property of Insertion

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

Universal Property of Insertion

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

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

Universal Property of Insertion

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

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

Universal Property of Insertion

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

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

Universal Property of Insertion

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

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

Universal Property of Insertion

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

$$\begin{tikzcd}
 & \Delta \arrow[d, "\kappa"] \arrow[rd, bend left, "\sigma"] & \\
\Theta \arrow[r, "\iota"'] \arrow[rrd, bend right, "\tau"'] & \Delta \ll\alpha \Theta \arrow[rd, "\sigma \ll\alpha \tau" description] & \\
 & & \Gamma
\end{tikzcd}$$

Universal Property of Insertion

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

$$\begin{tikzcd}
D_n \arrow[r, "\alpha"] \arrow[d, "\text{coh}\ (\Theta:B)[\text{id}]"'] & \Delta \arrow[d, "\kappa"] \arrow[rdd, bend left, "\sigma"] & \\
\Theta \arrow[r, "\iota"'] \arrow[rrd, bend right, "\tau"'] & \Delta \ll\alpha \Theta \arrow[rd, "\sigma \ll\alpha \tau" description] & \\
 & & \Gamma
\end{tikzcd}$$

Universal Property of Insertion

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

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

Properties of Insertion

Insertion generates a reduction relation for $\mathsf{Catt}_{\mathsf{sa}}$:

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

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

Properties of Insertion

Insertion generates a reduction relation for $\mathsf{Catt}_{\mathsf{sa}}$:

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

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

This reduction has been proven to have the following properties:

- Subject reduction
- Termination
- Confluence

References

- Finster, Eric and Samuel Mimram. *A Type-Theoretical Definition of Weak ω-Categories*. 2017. DOI: 10.1109/lics.2017.8005124. eprint: 1706.02866.
- Finster, Eric, Alex R., and Jamie Vicary. *A Type Theory for Strictly Associative Infinity Categories*. 2021. arXiv: 2109.01513.
- Finster, Eric, David Reutter, et al. *A Type Theory for Strictly Unital ∞-Categories*. Proceedings of the Thirty-Seventh Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2022). 2020. DOI: 10.1145/3531130.3533363. arXiv: 2007.08307.