CaTT: A type theoretic approach to weak ω-categories

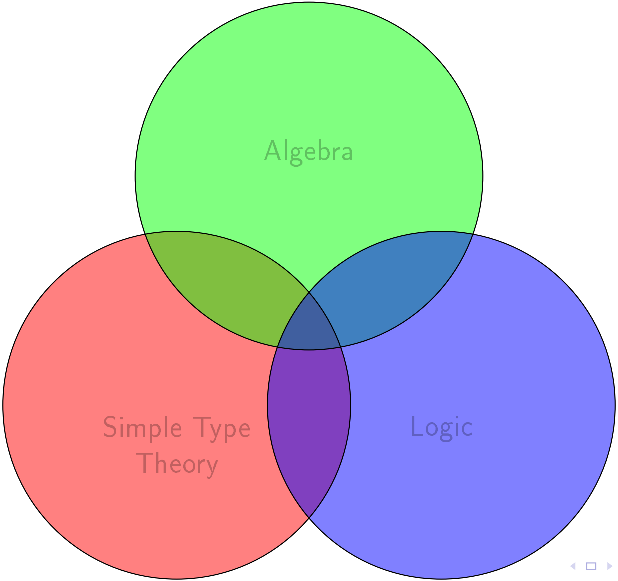
Thibaut Benjamin, Eric Finster, Samuel Mimram

12 February 2021

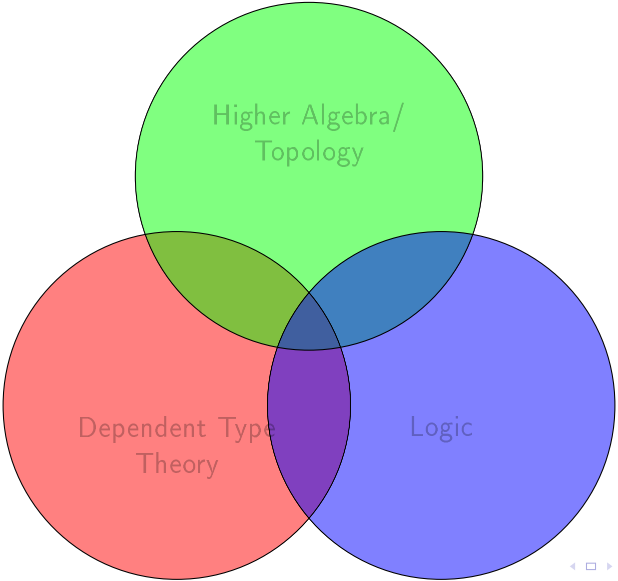
Seminaire polygraphes, categories superieures

Introduction

The panorama



The panorama



Motivations: A bit of category theory

Categories are made of

Motivations: A bit of category theory

Categories are made of

objects

●

Motivations: A bit of category theory

Categories are made of

objects

$\bullet$

arrows

$\bullet \longrightarrow \bullet$

Motivations: A bit of category theory

Categories are made of

objects

$\bullet$

arrows

$\bullet \longrightarrow \bullet$

Equipped with

Motivations: A bit of category theory

Categories are made of

objects

$\bullet$

arrows

$\bullet \longrightarrow \bullet$

Equipped with

identities

$x \rightsquigarrow \quad x \xrightarrow{\mathrm{id}_x} x$

Motivations: A bit of category theory

Categories are made of

objects

$\bullet$

arrows

$\bullet \longrightarrow \bullet$

Equipped with

identities

$x \rightsquigarrow x \xrightarrow{\mathrm{id}_x} x$

compositions

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

Motivations: A bit of category theory

Categories are made of

objects

$$\bullet$$

arrows

$$\bullet \longrightarrow \bullet$$

Equipped with

identities

$$x \rightsquigarrow x \xrightarrow{\mathrm{id}_x} x$$

compositions

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

Satisfying

Motivations: A bit of category theory

Categories are made of

objects

$\bullet$

arrows

$\bullet \longrightarrow \bullet$

Equipped with

identities

$x \rightsquigarrow \quad x \xrightarrow{\mathrm{id}_x} x$

compositions

$x \xrightarrow{f} y \xrightarrow{g} z \quad \rightsquigarrow \quad x \xrightarrow{g \circ f} z$

Satisfying

unitality

$f \circ \mathrm{id}_x = f = \mathrm{id}_y \circ f$

Motivations: A bit of category theory

Categories are made of

objects

$\bullet$

arrows

$\bullet \longrightarrow \bullet$

Equipped with

identities

$x \rightsquigarrow \quad x \xrightarrow{\mathrm{id}_x} x$

compositions

$x \xrightarrow{f} y \xrightarrow{g} z \quad \rightsquigarrow \quad x \xrightarrow{g \circ f} z$

Satisfying

unitality

$f \circ \mathrm{id}_x = f = \mathrm{id}_y \circ f$

associativity

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

A few example of categories

- The category of sets: objects = sets, morphisms = functions

A few example of categories

- The category of sets: objects = sets, morphisms = functions
- The category of groups: objects = groups, morphisms = group homomorphisms

A few example of categories

- The category of sets: objects = sets, morphisms = functions
- The category of groups: objects = groups, morphisms = group homomorphisms
- The category of vector spaces: objects = vector spaces, morphisms = linear maps

A few example of categories

- The category of sets: objects = sets, morphisms = functions
- The category of groups: objects = groups, morphisms = group homomorphisms
- The category of vector spaces: objects = vector spaces, morphisms = linear maps
- The category of metro itineraries: objects = metro stations, morphisms = metro itineraries

A few example of categories

- The category of sets: objects = sets, morphisms = functions
- The category of groups: objects = groups, morphisms = group homomorphisms
- The category of vector spaces: objects = vector spaces, morphisms = linear maps
- The category of metro itineraries: objects = metro stations, morphisms = metro itineraries
- The syntactic category of a type theory : objects = contexts, morphisms = substitutions

A few example of categories

- The category of sets: objects = sets, morphisms = functions
- The category of groups: objects = groups, morphisms = group homomorphisms
- The category of vector spaces: objects = vector spaces, morphisms = linear maps
- The category of metro itineraries: objects = metro stations, morphisms = metro itineraries
- The syntactic category of a type theory : objects = contexts, morphisms = substitutions
- Every monoid M is a category : objects = $\{\bullet\}$, morphisms = M.

Motivations: Why categories?

Categories are ubiquitous, they are notably useful for:

Motivations: Why categories?

Categories are ubiquitous, they are notably useful for:

- Modeling functional programming (objects = types, morphisms = λ-terms)

Motivations: Why categories?

Categories are ubiquitous, they are notably useful for:

- Modeling functional programming (objects = types, morphisms = λ-terms)
- Encoding propositions and implications

Motivations: Why categories?

Categories are ubiquitous, they are notably useful for:

- Modeling functional programming (objects = types, morphisms = λ-terms)
- Encoding propositions and implications
- Defining algebraic structures

Motivations: Why categories?

Categories are ubiquitous, they are notably useful for:

- Modeling functional programming (objects = types, morphisms = λ-terms)
- Encoding propositions and implications
- Defining algebraic structures
- And many other things...

Motivations: Why categories?

Categories are ubiquitous, they are notably useful for:

- ▷ Modeling functional programming (objects = types, morphisms = λ-terms)
- ▷ Encoding propositions and implications
- ▷ Defining algebraic structures
- ▷ And many other things...

Provide a common framework to study programming languages, logic and algebra
↳ Semantics

Foreshadowing: Type theoretic formulation of categories

Categories are made of

objects

$\bullet$

arrows

$\bullet \longrightarrow \bullet$

Equipped with

identities

$x \rightsquigarrow x \xrightarrow{\mathrm{id}_x} x$

compositions

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

Satisfying

unitality

$f \circ \mathrm{id}_x = f = \mathrm{id}_y \circ f$

associativity

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

Foreshadowing: Type theoretic formulation of categories

Categories are made of

objects

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

arrows

$$\bullet \longrightarrow \bullet$$

Equipped with

identities

$$x \rightsquigarrow \quad x \xrightarrow{\mathrm{id}_x} x$$

compositions

$$x \xrightarrow{f} y \xrightarrow{g} z \quad \rightsquigarrow \quad x \xrightarrow{g \circ f} z$$

Satisfying

unitality

$$f \circ \mathrm{id}_x = f = \mathrm{id}_y \circ f$$

associativity

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

Foreshadowing: Type theoretic formulation of categories

Categories are made of

objects

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

arrows

$$\frac{\Gamma \vdash x : \star \qquad \Gamma \vdash y : \star}{\Gamma \vdash x \to y}$$

Equipped with

identities

$$x \rightsquigarrow \quad x \xrightarrow{\mathrm{id}_x} x$$

compositions

$$x \xrightarrow{f} y \xrightarrow{g} z \quad \rightsquigarrow \quad x \xrightarrow{g \circ f} z$$

Satisfying

unitality

$$f \circ \mathrm{id}_x = f = \mathrm{id}_y \circ f$$

associativity

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

Foreshadowing: Type theoretic formulation of categories

Categories are made of

objects

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

arrows

$$\frac{\Gamma \vdash x : \star \qquad \Gamma \vdash y : \star}{\Gamma \vdash x \to y}$$

Equipped with

identities

$$\frac{\Gamma \vdash x : \star}{\Gamma \vdash \mathrm{id}(x) : x \to x}$$

compositions

$$x \xrightarrow{f} y \xrightarrow{g} z \quad \rightsquigarrow \quad x \xrightarrow{g \circ f} z$$

Satisfying

unitality

$$f \circ \mathrm{id}_x = f = \mathrm{id}_y \circ f$$

associativity

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

Foreshadowing: Type theoretic formulation of categories

Categories are made of

objects

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

arrows

$$\frac{\Gamma \vdash x : \star \qquad \Gamma \vdash y : \star}{\Gamma \vdash x \to y}$$

Equipped with

identities

$$\frac{\Gamma \vdash x : \star}{\Gamma \vdash \mathrm{id}(x) : x \to x}$$

compositions

$$\frac{\Gamma \vdash f : x \to y \qquad \Gamma \vdash g : y \to z}{\Gamma \vdash g \circ f : x \to z}$$

Satisfying

unitality

$$f \circ \mathrm{id}_x = f = \mathrm{id}_y \circ f$$

associativity

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

Foreshadowing: Type theoretic formulation of categories

Categories are made of

objects

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

arrows

$$\frac{\Gamma \vdash x : \star \qquad \Gamma \vdash y : \star}{\Gamma \vdash x \longrightarrow y}$$

Equipped with

identities

$$\frac{\Gamma \vdash x : \star}{\Gamma \vdash \mathrm{id}(x) : x \longrightarrow x}$$

compositions

$$\frac{\Gamma \vdash f : x \longrightarrow y \qquad \Gamma \vdash g : y \longrightarrow z}{\Gamma \vdash g \circ f : x \longrightarrow z}$$

Satisfying

unitality

$$\frac{\Gamma \vdash f : x \longrightarrow y}{\Gamma \vdash f \circ \mathrm{id}(x) \equiv f \equiv \mathrm{id}(y) \circ f}$$

associativity

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

Foreshadowing: Type theoretic formulation of categories

Categories are made of

objects

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

arrows

$$\frac{\Gamma \vdash x : \star \qquad \Gamma \vdash y : \star}{\Gamma \vdash x \to y}$$

Equipped with

identities

$$\frac{\Gamma \vdash x : \star}{\Gamma \vdash \mathrm{id}(x) : x \to x}$$

compositions

$$\frac{\Gamma \vdash f : x \to y \qquad \Gamma \vdash g : y \to z}{\Gamma \vdash g \circ f : x \to z}$$

Satisfying

unitality

$$\frac{\Gamma \vdash f : x \to y}{\Gamma \vdash f \circ \mathrm{id}(x) \equiv f \equiv \mathrm{id}(y) \circ f}$$

associativity

$$\frac{\Gamma \vdash f : x \to y \qquad \Gamma \vdash g : y \to z \qquad \Gamma \vdash h : z \to w}{\Gamma \vdash h \circ (g \circ f) \equiv (h \circ g) \circ f}$$

The usefulness of categories for semantics

- One can encode the syntax together with derivability using categories
  The syntactic category: objects = contexts, morphisms = substitutions

The usefulness of categories for semantics

- One can encode the syntax together with derivability using categories
  The syntactic category: objects = contexts, morphisms = substitutions
- How to encode the type and terms of the theory?
  Use an elaborate categorical construction: Categories with families (CwF)

The usefulness of categories for semantics

- One can encode the syntax together with derivability using categories
  The syntactic category: objects = contexts, morphisms = substitutions
- How to encode the type and terms of the theory?
  Use an elaborate categorical construction: Categories with families (CwF)
- The syntactic category of the theory is a CwF
  Encodes all the rules into a categorical device, comes with a notion of morphism

The usefulness of categories for semantics

- One can encode the syntax together with derivability using categories
  The syntactic category: objects = contexts, morphisms = substitutions
- How to encode the type and terms of the theory?
  Use an elaborate categorical construction: Categories with families (CwF)
- The syntactic category of the theory is a CwF
  Encodes all the rules into a categorical device, comes with a notion of morphism
- Models of the theory : morphisms of CwF from the syntactic category to sets
  Interprets the rules of the theory as axioms of an algebraic theory

The usefulness of categories for semantics

- One can encode the syntax together with derivability using categories
  The syntactic category: objects = contexts, morphisms = substitutions
- How to encode the type and terms of the theory?
  Use an elaborate categorical construction: Categories with families (CwF)
- The syntactic category of the theory is a CwF
  Encodes all the rules into a categorical device, comes with a notion of morphism
- Models of the theory : morphisms of CwF from the syntactic category to sets
  Interprets the rules of the theory as axioms of an algebraic theory

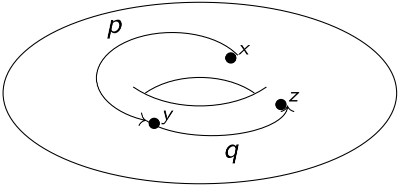
Theorem

The models of the type-theoretic formulation of categories are the categories.

Motivations: When categories are insufficient

Some situation that ought to be categories

- paths in topological spaces



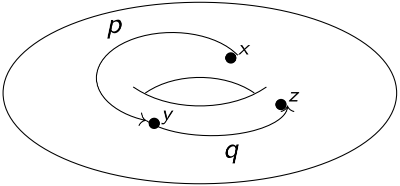
But: The composition is not associative

$$(p * q) * r \neq p * (q * r)$$

Motivations: When categories are insufficient

Some situation that ought to be categories

- paths in topological spaces



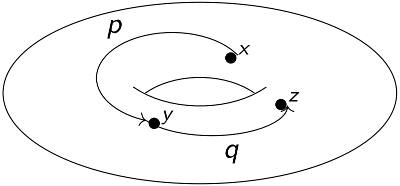
The composition is associative up to homotopy

$$(p * q) * r \Rightarrow p * (q * r)$$

Motivations: When categories are insufficient

Some situation that ought to be categories

- paths in topological spaces



The composition is associative up to homotopy

$$(p * q) * r \Rightarrow p * (q * r)$$

- Identity types in Martin-Löf type theory

$$\mathtt{trans} : \prod A : \mathcal{U},\ \prod x\ y\ z : A,$$
$$x = y \to y = z \to x = z$$
$$\mathtt{trans}\ \ A\ x\ x\ x\ \mathtt{refl}\ \ \mathtt{refl}\ \ \mathtt{:=}\ \mathtt{refl}$$

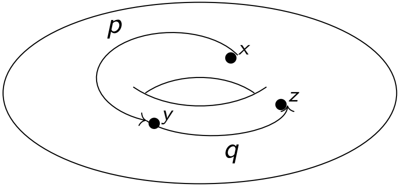
But: Transitivity is not associative

$$\mathtt{trans}(\mathtt{trans}\ p\ q)\ r \not\equiv \mathtt{trans}\ p\ (\mathtt{trans}\ q\ r)$$

Motivations: When categories are insufficient

Some situation that ought to be categories

- paths in topological spaces



The composition is associative up to homotopy

$$(p * q) * r \Rightarrow p * (q * r)$$

- Identity types in Martin-Löf type theory

$$\mathtt{trans} : \prod A : \mathcal{U},\ \prod x\ y\ z : A,$$
$$x = y \to y = z \to x = z$$
$$\mathtt{trans}\ A\ x\ x\ x\ \mathtt{refl}\ \mathtt{refl}\ \mathtt{:=}\ \mathtt{refl}$$

There is a proof term for associativity

$$\mathtt{assoc} : \prod A : \mathcal{U},\ \prod\ x\ y\ z\ w : A,$$
$$\prod p : x = y,\ \prod q : y = z,\ \prod r : z = w,$$
$$\mathtt{trans}(\mathtt{trans}\ p\ q)\ r = \mathtt{trans}\ p\ (\mathtt{trans}\ q\ r)$$

Higher categories

We can encode this defect into cells of higher dimension

Weak ω-categories

Globular sets

Weak ω-categories = globular sets + compositions

Globular sets

A globular set is composed of

- ▷ *points* (or *objects*) : ●

Globular sets

A globular set is composed of

- *points* (or *objects*) : $\bullet$
- *arrows* (or 1-*cells*) : $\bullet \longrightarrow \bullet$

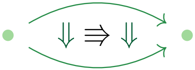
Globular sets

A globular set is composed of

- *points* (or *objects*) : $\bullet$
- *arrows* (or 1-*cells*) : $\bullet \longrightarrow \bullet$
- 2-*cells* :
  $$\begin{tikzcd} \bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Downarrow"] & \bullet \end{tikzcd}$$

Globular sets

A globular set is composed of

- *points* (or *objects*) : $\bullet$
- *arrows* (or 1-*cells*) : $\bullet \longrightarrow \bullet$
- 2-*cells* : $$\begin{tikzcd} \bullet \arrow[r, bend left, ""{name=U, below}] \arrow[r, bend right, ""{name=D}] & \bullet \arrow[Rightarrow, from=U, to=D] \end{tikzcd}$$
- 3-*cells* : 

Globular sets

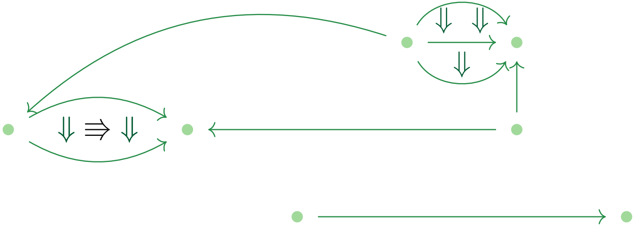
A globular set is composed of

- *points* (or *objects*) : $\bullet$
- *arrows* (or 1-*cells*) : $\bullet \longrightarrow \bullet$
- 2-*cells* : $$\begin{tikzcd} \bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Downarrow"] & \bullet \end{tikzcd}$$
- 3-*cells* : $$\begin{tikzcd} \bullet \arrow[r, bend left=40] \arrow[r, bend right=40] \arrow[r, phantom, "\Downarrow \Rrightarrow \Downarrow"] & \bullet \end{tikzcd}$$
- etc

Globular sets

A globular set is composed of

- *points* (or *objects*) : $\bullet$
- *arrows* (or 1-*cells*) : $\bullet \longrightarrow \bullet$
- 2-*cells* : $\begin{tikzcd} \bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Downarrow"] & \bullet \end{tikzcd}$
- 3-*cells* : $\begin{tikzcd} \bullet \arrow[r, bend left] \arrow[r, bend right] \arrow[r, phantom, "\Downarrow \Rrightarrow \Downarrow"] & \bullet \end{tikzcd}$
- etc



With compositions

We require that these cells compose

With compositions

We require that these cells compose

- Arrows:

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

With compositions

We require that these cells compose

- Arrows:

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

With compositions

We require that these cells compose

- Arrows:

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

- 2-cells:

Vertical composition

$$\begin{tikzcd} x \arrow[r, bend left=60, "f", ""{name=F, below}] \arrow[r, "g" description, ""{name=G1, above}, ""{name=G2, below}] \arrow[r, bend right=60, "h"', ""{name=H, above}] & y \arrow[Rightarrow, from=F, to=G1, "\alpha"] \arrow[Rightarrow, from=G2, to=H, "\beta"] \end{tikzcd}$$

With compositions

We require that these cells compose

- Arrows:

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

- 2-cells:

Vertical composition

$$\begin{tikzcd}[row sep=large] x \arrow[r, bend left=60, "f", ""{name=F, below}] \arrow[r, "g" description, ""{name=G}] \arrow[r, bend right=60, "h"', ""{name=H, above}] \arrow[Rightarrow, from=F, to=G, "\alpha"] \arrow[Rightarrow, from=G, to=H, "\beta"] \arrow[d, squiggly] & y \\ x \arrow[r, bend left=60, "f", ""{name=F2, below}] \arrow[r, bend right=60, "h"', ""{name=H2, above}] \arrow[Rightarrow, from=F2, to=H2, "\beta\circ_1\alpha"] & y \end{tikzcd}$$

With compositions

We require that these cells compose

- Arrows:

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

- 2-cells:

Vertical composition

$$\begin{tikzcd}[column sep=huge] x \arrow[r, bend left=60, "f", ""{name=F, below}] \arrow[r, "g" description, ""{name=G}] \arrow[r, bend right=60, "h"', ""{name=H, above}] & y \arrow[Rightarrow, from=F, to=G, "\alpha"] \arrow[Rightarrow, from=G, to=H, "\beta"] \end{tikzcd}$$

$$\begin{tikzcd} {} \arrow[d, rightsquigarrow] \\ {} \end{tikzcd}$$

$$\begin{tikzcd}[column sep=huge] x \arrow[r, bend left=50, "f", ""{name=F, below}] \arrow[r, bend right=50, "h"', ""{name=H, above}] & y \arrow[Rightarrow, from=F, to=H, "\beta\circ_{\mathbf{1}}\alpha"] \end{tikzcd}$$

Horizontal composition

$$\begin{tikzcd}[column sep=huge] x \arrow[r, bend left=40, "f", ""{name=F, below}] \arrow[r, bend right=40, "g"', ""{name=G, above}] & y \arrow[r, bend left=40, "f'", ""{name=F2, below}] \arrow[r, bend right=40, "g'"', ""{name=G2, above}] & z \arrow[Rightarrow, from=F, to=G, "\alpha"] \arrow[Rightarrow, from=F2, to=G2, "\beta"] \end{tikzcd}$$

With compositions

We require that these cells compose

- Arrows:

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

- 2-cells:

Vertical composition

$$\begin{tikzcd} x \arrow[r, bend left=60, "f"{name=F}] \arrow[r, "g"{description, name=G}] \arrow[r, bend right=60, "h"'{name=H}] & y \arrow[from=F, to=G, Rightarrow, "\alpha"] \arrow[from=G, to=H, Rightarrow, "\beta"] \end{tikzcd}$$

$$\begin{tikzcd} {} \arrow[d, squiggly] \\ {} \end{tikzcd}$$

$$\begin{tikzcd} x \arrow[r, bend left=60, "f"{name=F}] \arrow[r, bend right=60, "h"'{name=H}] & y \arrow[from=F, to=H, Rightarrow, "\beta\circ_1\alpha"] \end{tikzcd}$$

Horizontal composition

$$\begin{tikzcd} x \arrow[r, bend left=40, "f"{name=F}] \arrow[r, bend right=40, "g"'{name=G}] & y \arrow[r, bend left=40, "f'"{name=F2}] \arrow[r, bend right=40, "g'"'{name=G2}] & z \arrow[from=F, to=G, Rightarrow, "\alpha"] \arrow[from=F2, to=G2, Rightarrow, "\beta"] \end{tikzcd}$$

$$\begin{tikzcd} {} \arrow[d, squiggly] \\ {} \end{tikzcd}$$

$$\begin{tikzcd} x \arrow[r, bend left=40, "f'\circ f"{name=F}] \arrow[r, bend right=40, "g'\circ g"'{name=G}] & z \arrow[from=F, to=G, Rightarrow, "\beta\circ_0\alpha"] \end{tikzcd}$$

With compositions

We require that these cells compose

- Arrows:

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

- 2-cells:

Vertical composition

$$\begin{tikzcd}[column sep=large] x \arrow[r, bend left=60, "f", ""{name=F, below}] \arrow[r, "g" description, ""{name=G1, above}, ""{name=G2, below}] \arrow[r, bend right=60, "h"', ""{name=H, above}] & y \arrow[Rightarrow, from=F, to=G1, "\alpha"] \arrow[Rightarrow, from=G2, to=H, "\beta"] \\ \arrow[d, squiggly] \\ x \arrow[r, bend left=60, "f", ""{name=F2, below}] \arrow[r, bend right=60, "h"', ""{name=H2, above}] & y \arrow[Rightarrow, from=F2, to=H2, "\beta\circ_1\alpha"] \end{tikzcd}$$

Horizontal composition

$$\begin{tikzcd}[column sep=large] x \arrow[r, bend left=40, "f", ""{name=A, below}] \arrow[r, bend right=40, "g"', ""{name=B, above}] & y \arrow[r, bend left=40, "f'", ""{name=C, below}] \arrow[r, bend right=40, "g'"', ""{name=D, above}] \arrow[d, squiggly] & z \arrow[Rightarrow, from=A, to=B, "\alpha"] \arrow[Rightarrow, from=C, to=D, "\beta"] \\ x \arrow[rr, bend left=40, "f'\circ f", ""{name=E, below}] \arrow[rr, bend right=40, "g'\circ g"', ""{name=G, above}] & & z \arrow[Rightarrow, from=E, to=G, "\beta\circ_0\alpha"] \end{tikzcd}$$

- etc

Satisfying associativity

The compositions are required to be associative

Satisfying associativity

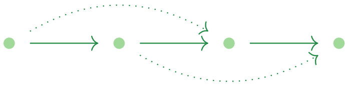
The compositions are required to be associative

- Arrows:

Satisfying associativity

The compositions are required to be associative

- Arrows:



Satisfying associativity

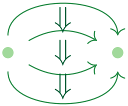
The compositions are required to be associative

- Arrows:

$$\begin{tikzcd}\bullet \arrow[r] & \bullet \arrow[r] & \bullet \arrow[r] & \bullet\end{tikzcd}$$

- 2-cells:

Vertical composition



Satisfying associativity

The compositions are required to be associative

- Arrows:

$$\begin{tikzcd}
\bullet \arrow[r] & \bullet \arrow[r] & \bullet \arrow[r] & \bullet
\end{tikzcd}$$

- 2-cells:

Vertical composition

$$\begin{tikzcd}[column sep=huge]
\bullet \arrow[r, bend left=70, ""{name=A, below}] \arrow[r, bend left=20, ""{name=B, above}, ""{name=B2, below}] \arrow[r, bend right=20, ""{name=C, above}, ""{name=C2, below}] \arrow[r, bend right=70, ""{name=D, above}] & \bullet
\arrow[Rightarrow, from=A, to=B] \arrow[Rightarrow, from=B2, to=C] \arrow[Rightarrow, from=C2, to=D]
\end{tikzcd}$$

Horizontal composition

$$\begin{tikzcd}
\bullet \arrow[r, bend left, ""{name=A1, below}] \arrow[r, bend right, ""{name=B1, above}] & \bullet \arrow[r, bend left, ""{name=A2, below}] \arrow[r, bend right, ""{name=B2, above}] & \bullet \arrow[r, bend left, ""{name=A3, below}] \arrow[r, bend right, ""{name=B3, above}] & \bullet
\arrow[Rightarrow, from=A1, to=B1] \arrow[Rightarrow, from=A2, to=B2] \arrow[Rightarrow, from=A3, to=B3]
\end{tikzcd}$$

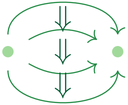
Satisfying associativity

The compositions are required to be associative

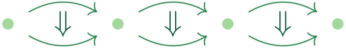
- Arrows:

- 2-cells:

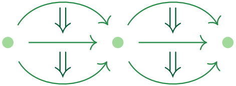
Vertical composition



Horizontal composition



Exchange law

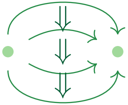


Satisfying associativity

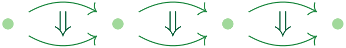
The compositions are required to be associative

- Arrows:

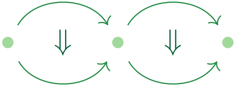
- 2-cells:



Vertical composition



Horizontal composition



Exchange law

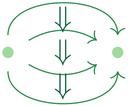
Satisfying associativity

The compositions are required to be associative

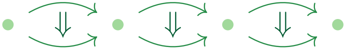
- Arrows:

- 2-cells:

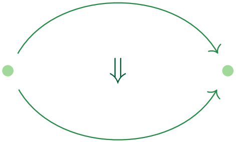
Vertical composition



Horizontal composition



Exchange law



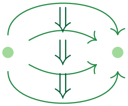
Satisfying associativity

The compositions are required to be associative

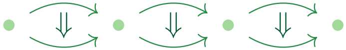
- Arrows:

- 2-cells:

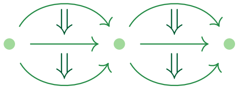
Vertical composition



Horizontal composition



Exchange law



Satisfying associativity

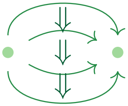
The compositions are required to be associative

- Arrows:

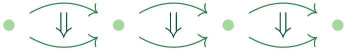
$$\begin{tikzcd} \bullet \arrow[r] & \bullet \arrow[r] & \bullet \arrow[r] & \bullet \end{tikzcd}$$

- 2-cells:

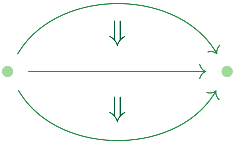
Vertical composition



Horizontal composition



Exchange law



Satisfying associativity

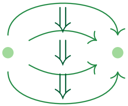
The compositions are required to be associative

- Arrows:

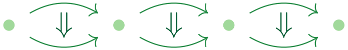
$$\begin{tikzcd} \bullet \arrow[r] & \bullet \arrow[r] & \bullet \arrow[r] & \bullet \end{tikzcd}$$

- 2-cells:

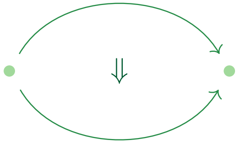
Vertical composition



Horizontal composition



Exchange law



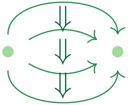
Satisfying associativity

The compositions are required to be associative

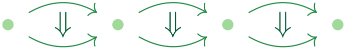
- Arrows:

- 2-cells:

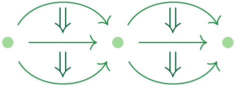
Vertical composition



Horizontal composition



Exchange law



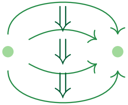
Satisfying associativity

The compositions are required to be associative

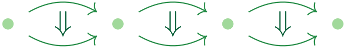
- Arrows:

- 2-cells:

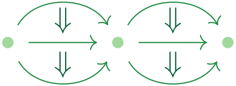
Vertical composition



Horizontal composition



Exchange law



- etc

Everything is weak!

All the above-mentioned equalities are weak: they are equivalences

Everything is weak!

All the above-mentioned equalities are weak: they are equivalences

$$\begin{tikzcd}
\bullet \arrow[r, bend left=40, "h\circ(g\circ f)"{name=U}] \arrow[r, bend right=40, "(h\circ g)\circ f"'{name=D}] & \bullet
\arrow[Rightarrow, from=U, to=D, shift right=3, "\alpha"'] \arrow[Rightarrow, from=D, to=U, shift right=3, "\beta"']
\end{tikzcd}$$

Pasting schemes

The pasting schemes are the globular set that have one unambiguous way of being fully composed

They are well-ordered and do not have any hole

Pasting schemes

The pasting schemes are the globular set that have one unambiguous way of being fully composed

They are well-ordered and do not have any hole

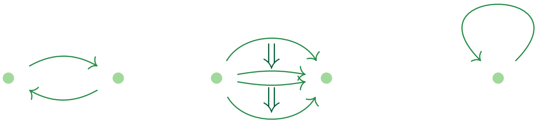
Examples

Pasting schemes

The pasting schemes are the globular set that have one unambiguous way of being fully composed

They are well-ordered and do not have any hole

Examples



Counter-Examples

The Grothendieck-Maltsiniotis definition [?]

- Existence of compositions:
  Every pasting scheme can be composed

The Grothendieck-Maltsiniotis definition [?]

- Existence of compositions:
  Every pasting scheme can be composed

- Generalized “Associativities”:
  Any two ways of composing a same pasting scheme are related by a higher cell

The Grothendieck-Maltsiniotis definition [?]

- Existence of compositions:
  Every pasting scheme can be composed

- Generalized "Associativities":
  Any two ways of composing a same pasting scheme are related by a higher cell

Formally:
These axioms are encoded in a category Θ_∞ defined by a universal property of formally adding lifts for well-chosen pairs of morphisms.

The Grothendieck-Maltsiniotis definition [?]

- Existence of compositions:
  Every pasting scheme can be composed

- Generalized “Associativities”:
  Any two ways of composing a same pasting scheme are related by a higher cell

Formally:
These axioms are encoded in a category Θ_∞ defined by a universal property of formally adding lifts for well-chosen pairs of morphisms.

The weak ω-categories are the presheaves over Θ_∞ that preserve a class of colimits.

The type theory CaTT

First Examples

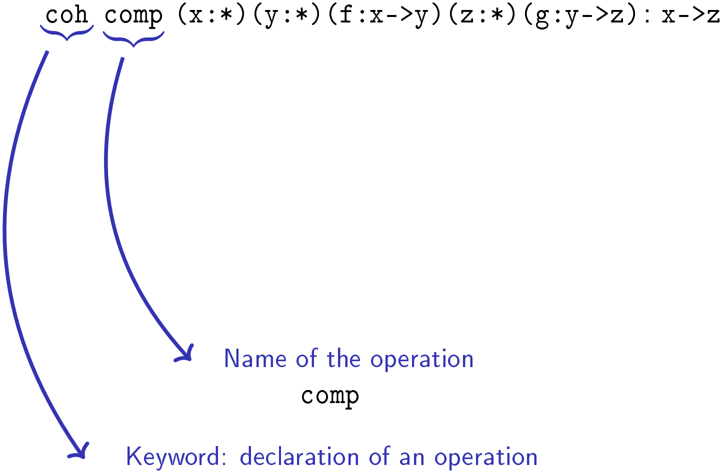
```
coh comp (x:*)(y:*)(f:x->y)(z:*)(g:y->z): x->z
```

First Examples

```
coh comp (x:*)(y:*)(f:x->y)(z:*)(g:y->z): x->z
```

Keyword: declaration of an operation

First Examples



```
coh comp (x:*)(y:*)(f:x->y)(z:*)(g:y->z): x->z
```

Name of the operation

`comp`

Keyword: declaration of an operation

First Examples

```
coh comp (x:*)(y:*)(f:x->y)(z:*)(g:y->z): x->z
```

Globular set

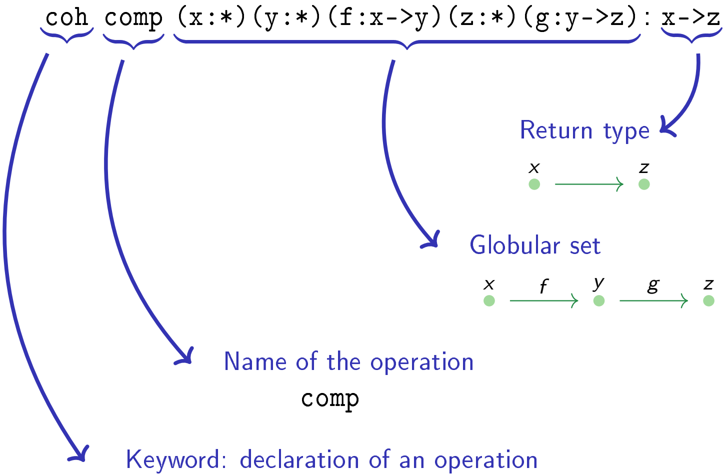
$$\begin{tikzcd} x \arrow[r, "f"] & y \arrow[r, "g"] & z \end{tikzcd}$$

Name of the operation

`comp`

Keyword: declaration of an operation

First Examples



```
coh comp (x:*)(y:*)(f:x->y)(z:*)(g:y->z): x->z
```

Return type

$$\begin{tikzcd} x \arrow[r] & z \end{tikzcd}$$

Globular set

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

Name of the operation

`comp`

Keyword: declaration of an operation

First Examples

```
coh comp (x:*)(y:*)(f:x->y)(z:*)(g:y->z): x->z
```

$$\begin{tikzcd} x \arrow[r, "f"] & y \arrow[r, "g"] & z \arrow[r, phantom, "\rightsquigarrow"] & x \arrow[r, "\texttt{comp f g}"] & z \end{tikzcd}$$

Composition of two arrows

First Examples

```
coh comp (x:*)(y:*)(f:x->y)(z:*)(g:y->z): x->z
```

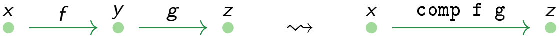
$$\begin{tikzcd} x \arrow[r, "f"] & y \arrow[r, "g"] & z \arrow[r, phantom, "\rightsquigarrow"] & x \arrow[r, "\texttt{comp f g}"] & z \end{tikzcd}$$

Composition of two arrows

```
coh id (x:*) :  x->x
```

First Examples

```
coh comp (x:*)(y:*)(f:x->y)(z:*)(g:y->z): x->z
```



Composition of two arrows

```
coh id (x:*) :  x->x
```

Identity of an object

Types for cells

In order to manipulate ω-categories, we need:

Types for cells

In order to manipulate ω-categories, we need:

- a type $\star$ for objects:

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

Types for cells

In order to manipulate ω-categories, we need:

- a type $\star$ for objects:

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

- a type $\to$ for arrows:

$$\frac{\Gamma \vdash t : \star \qquad \Gamma \vdash u : \star}{\Gamma \vdash t \to u}$$

Types for cells

In order to manipulate ω-categories, we need:

- a type $\star$ for objects:

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

- a type $\rightarrow$ for arrows:

$$\frac{\Gamma \vdash t : \star \qquad \Gamma \vdash u : \star}{\Gamma \vdash t \rightarrow u}$$

- a type $\Rightarrow$ for 2-cells:

$$\frac{\Gamma \vdash f : t \rightarrow u \qquad \Gamma \vdash g : t \rightarrow u}{\Gamma \vdash f \Rightarrow g}$$

Types for cells

In order to manipulate ω-categories, we need:

- a type $\star$ for objects:

$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

- a type $\rightarrow$ for arrows:

$$\frac{\Gamma \vdash t : \star \qquad \Gamma \vdash u : \star}{\Gamma \vdash t \rightarrow u}$$

- a type $\Rightarrow$ for 2-cells:

$$\frac{\Gamma \vdash f : t \rightarrow u \qquad \Gamma \vdash g : t \rightarrow u}{\Gamma \vdash f \Rightarrow g}$$

- etc

Types for cells

In order to manipulate ω-categories, we need:

- a type $\star$ for objects:

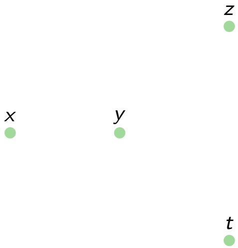
$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

- a type $\rightarrow$ for all $\geq$ 1-cells

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash t \underset{A}{\longrightarrow} u}$$

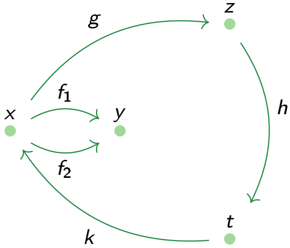
Contexts are diagrams!

$$\Gamma \left\{ x : \star, \qquad y : \star, \qquad z : \star, \qquad t : \star \right.$$



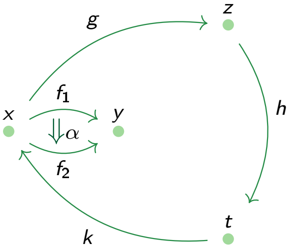
Contexts are diagrams!

$$\Gamma \left\{ \begin{array}{lllll} x : \star, & y : \star, & z : \star, & t : \star & \\ f_1 : x \xrightarrow[\star]{} y, & f_2 : x \xrightarrow[\star]{} y, & g : x \xrightarrow[\star]{} z, & h : z \xrightarrow[\star]{} t, & k : t \xrightarrow[\star]{} x \\ & & & & \\ & & & & \end{array} \right.$$



Contexts are diagrams!

$$\Gamma \left\{ \begin{array}{lllll} x : \star, & y : \star, & z : \star, & t : \star & \\ f_1 : x \underset{\star}{\to} y, & f_2 : x \underset{\star}{\to} y, & g : x \underset{\star}{\to} z, & h : z \underset{\star}{\to} t, & k : t \underset{\star}{\to} x \\ \alpha : f_1 \xrightarrow[x \underset{\star}{\to} y]{} f_2 & & & & \end{array} \right.$$



PS-contexts

- The ps-contexts are the contexts corresponding to pasting schemes.

They are recognized by a judgment $\Gamma \vdash_{\mathsf{ps}}$

PS-contexts

- ▷ The ps-contexts are the contexts corresponding to pasting schemes.

They are recognized by a judgment $\Gamma \vdash_{\mathrm{ps}}$

- ▷ This judgment is decidable with an algorithm

$$\frac{}{x : \star \vdash_{\mathrm{ps}} x : \star} \qquad \frac{\Gamma \vdash_{\mathrm{ps}} x : A}{\Gamma, y : A, f : x \xrightarrow[A]{} y \vdash_{\mathrm{ps}} f : x \xrightarrow[A]{} y}$$

$$\frac{\Gamma \vdash_{\mathrm{ps}} f : x \xrightarrow[A]{} y}{\Gamma \vdash_{\mathrm{ps}} y : A} \qquad \frac{\Gamma \vdash_{\mathrm{ps}} x : \star}{\Gamma \vdash_{\mathrm{ps}}}$$

Source and target

Every ps-context Γ has a source $\partial^{-}(\Gamma)$ and a target $\partial^{+}(\Gamma)$ (which are ps-contexts themselves)

Intuitively, composing fully the ps-context yields a cell from its source to its target

Source and target

Every ps-context Γ has a source $\partial^{-}(\Gamma)$ and a target $\partial^{+}(\Gamma)$ (which are ps-contexts themselves)

Intuitively, composing fully the ps-context yields a cell from its source to its target

Γ

$$\begin{tikzcd}
x \arrow[r, bend left=60, "f_1"] \arrow[r, "f_2" description] \arrow[r, bend right=60, "f_3"'] \arrow[r, phantom, shift left=3, "\Downarrow \alpha"] \arrow[r, phantom, shift right=3, "\Downarrow \beta"] & y \arrow[r, "g"] & z
\end{tikzcd}$$

Example

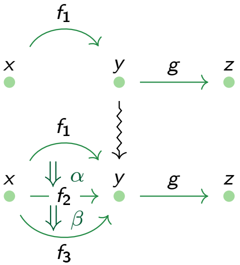
Source and target

Every ps-context Γ has a source $\partial^-(\Gamma)$ and a target $\partial^+(\Gamma)$ (which are ps-contexts themselves)

Intuitively, composing fully the ps-context yields a cell from its source to its target

$\partial^-(\Gamma)$

Γ

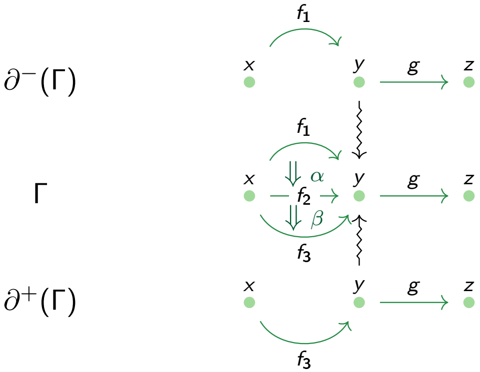


Example

Source and target

Every ps-context Γ has a source $\partial^-(\Gamma)$ and a target $\partial^+(\Gamma)$ (which are ps-contexts themselves)

Intuitively, composing fully the ps-context yields a cell from its source to its target



Example

Interpretation

- A term $\Gamma \vdash t : A$ corresponds to a composition of certain cells of Γ.
  More precisely: a cell in the weak ω-catégory freely generated by the globular set corresponding to Γ.

Interpretation

- A term $\Gamma \vdash t : A$ corresponds to a composition of certain cells of Γ.
  More precisely: a cell in the weak ω-catégory freely generated by the globular set corresponding to Γ.

- Case of ps-contexts $\Gamma \vdash_{\mathsf{ps}}$:

$$\left.\begin{array}{l} \Gamma \vdash t : A \\ \mathrm{Var}(t : A) = \mathrm{Var}(\Gamma) \end{array}\right\}$$

t is a way of composing completely the ps-context Γ.

Operations and coherences

- Existence of compositions:
  Every pasting scheme can be composed

Operations and coherences

- Existence of compositions:
  Every pasting scheme can be composed

Rule for generating these compositions:

$$\frac{\Gamma \vdash_{\mathsf{ps}} \qquad \partial^{-}(\Gamma) \vdash t : A \qquad \partial^{+}(\Gamma) \vdash u : A}{\Gamma \vdash \mathsf{op}_{\Gamma, t \underset{A}{\longrightarrow} u} : t \underset{A}{\longrightarrow} u}$$

$$\mathsf{Var}(t : A) = \mathsf{Var}(\partial^{-}(\Gamma))$$
$$\mathsf{Var}(u : A) = \mathsf{Var}(\partial^{+}(\Gamma))$$

Operations and coherences

- Existence of compositions:
  Every pasting scheme can be composed

Rule for generating these compositions:

$$\frac{\Gamma \vdash_{\mathsf{ps}} \qquad \partial^{-}(\Gamma) \vdash t : A \qquad \partial^{+}(\Gamma) \vdash u : A}{\Gamma \vdash \mathsf{op}_{\Gamma, t \underset{A}{\longrightarrow} u} : t \underset{A}{\longrightarrow} u}$$

$$\begin{aligned} \mathrm{Var}(t : A) &= \mathrm{Var}(\partial^{-}(\Gamma)) \\ \mathrm{Var}(u : A) &= \mathrm{Var}(\partial^{+}(\Gamma)) \end{aligned}$$

$$\Gamma = x : \star, y : \star, f : x \underset{\star}{\longrightarrow} y, z : \star, g : y \underset{\star}{\longrightarrow} z$$

$$\Gamma \vdash \mathtt{comp} : x \longrightarrow z$$

Example

Operations and coherences

- Existence of compositions:
  Every pasting scheme can be composed

- general "Associativities": **Any two ways of composing the same pasting scheme are related by a higher cell**

Operations and coherences

- Existence of compositions:
  Every pasting scheme can be composed
- general "Associativities": **Any two ways of composing the same pasting scheme are related by a higher cell**

Rule for generating these associativities:

$$\frac{\Gamma \vdash_{\mathrm{ps}} \qquad \Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash \mathrm{coh}_{\Gamma, t \underset{A}{\longrightarrow} u} : t \underset{A}{\longrightarrow} u} \qquad \begin{array}{l} \mathrm{Var}(t : A) = \mathrm{Var}(\Gamma) \\ \mathrm{Var}(u : A) = \mathrm{Var}(\Gamma) \end{array}$$

Operations and coherences

- Existence of compositions:
  Every pasting scheme can be composed
- general "Associativities": **Any two ways of composing the same pasting scheme are related by a higher cell**

Rule for generating these associativities:

$$\frac{\Gamma \vdash_{\mathrm{ps}} \qquad \Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash \mathrm{coh}_{\Gamma, t \underset{A}{\longrightarrow} u} : t \underset{A}{\longrightarrow} u} \qquad \begin{array}{l} \mathrm{Var}(t : A) = \mathrm{Var}(\Gamma) \\ \mathrm{Var}(u : A) = \mathrm{Var}(\Gamma) \end{array}$$

$$\Gamma = x : \star, y : \star, f : x \underset{\star}{\longrightarrow} y, z : \star, g : y \underset{\star}{\longrightarrow} z, w : \star, h : z \underset{\star}{\longrightarrow} w$$

$$\Gamma \vdash \mathtt{assoc} : \mathtt{comp\ f\ (comp\ g\ h)} \longrightarrow \mathtt{comp\ (comp\ f\ g)\ h}$$

Example

Encoding the theory

Recall the syntactic category: $\mathcal{S}_{\mathsf{CaTT}}$

- objects = contexts of the theory CaTT
- morphisms = substitutions of the theory

This category carries a structure of CwF

Suspension in CaTT

A problem with CaTT

Writing in CaTT may be very tedious :

$$x \rightsquigarrow \quad x \xrightarrow{\mathrm{id}_x} x$$

```
coh id (x:*):x->x
```

$$\begin{tikzcd} x \arrow[r, "f"] & y \end{tikzcd} \quad \rightsquigarrow \quad \begin{tikzcd} x \arrow[r, bend left, "f"{name=U}] \arrow[r, bend right, "f"'{name=D}] \arrow[from=U, to=D, Rightarrow, phantom, "\Downarrow \mathrm{id}_f"] & y \end{tikzcd}$$

```
coh id2 (x:*)(y:*)(f:x->y):f->f
```

A problem with CaTT

Writing in CaTT may be very tedious :

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

```
coh comp (x:*)(y:*)(f:x->y)
(z:*)(g:y->z):x->z
```

$$\begin{tikzcd} x \arrow[r, bend left=60, "f"{name=F}] \arrow[r, "g"{name=G, description}] \arrow[r, bend right=60, "h"'{name=H}] & y \arrow[Rightarrow, from=F, to=G, "\alpha"] \arrow[Rightarrow, from=G, to=H, "\beta"] \arrow[r, phantom, "\rightsquigarrow"] & x \arrow[r, bend left=60, "f"{name=F2}] \arrow[r, bend right=60, "h"'{name=H2}] & y \arrow[Rightarrow, from=F2, to=H2, "\alpha * \beta"] \end{tikzcd}$$

```
coh vcomp (x:*)(y:*)(f:x->y)(g:x->y)
(a:f->g)(h:x->y)(b:g->h):f->h
```

A problem with CaTT

Writing in CaTT may be very tedious :

$$\begin{tikzcd} x \arrow[r, "f"] & y \end{tikzcd} \rightsquigarrow \begin{tikzcd} x \arrow[r, bend left, "f\circ\mathrm{id}_x"] \arrow[r, bend right, "f"'] \arrow[r, phantom, "\Downarrow"] & y \end{tikzcd}$$

```
coh unitl (x:*)(y:*)(f:x->y)
    :comp (id x) f -> f
```

$$\begin{tikzcd} x \arrow[r, bend left, "f"] \arrow[r, bend right, "g"'] \arrow[r, phantom, "\Downarrow\alpha"] & y \end{tikzcd} \rightsquigarrow \begin{tikzcd}[column sep=huge] x \arrow[r, bend left=50, "f"] \arrow[r, bend right=50, "h"'] \arrow[r, phantom, "\mathrm{id}_f * \alpha \Downarrow \Rrightarrow \Downarrow \alpha"] & y \end{tikzcd}$$

```
coh unitl2 (x:*)(y:*)(f:x->y)(g:x->y)
    (a:f->g):vcomp (id f) a -> a
```

A solution: the suspension

Idea: Write only the left term and let the software generate the right one.

```
coh id (x:*):x->x
```

$\overset{\text{suspension}}{\rightsquigarrow}$

```
coh id2 (x:*)(y:*)(f:x->y):f->f
```

Practice: Generate the terms on the fly using the dimension of the argument.

```
coh id (x:*):x->x
let ex (x:*)(y:*)(f:x->y)=id f
```

$\rightsquigarrow$

```
coh id (x:*):x->x
coh id2 (x:*)(y:*)(f:x->y):f->f
let ex (x:*)(y:*)(f:x->y)=id2 f
```

Definition of the suspension

Induction over the syntax: Define the suspension as a meta-operation on the syntax of the type theory.

Definition of the suspension

Induction over the syntax: Define the suspension as a meta-operation on the syntax of the type theory.

$$\Sigma\varnothing = (x_0 : \star, y_0 : \star) \qquad \Sigma(\Gamma, x : A) = \Sigma\Gamma, x : \Sigma A$$

Definition of the suspension

Induction over the syntax: Define the suspension as a meta-operation on the syntax of the type theory.

$$\Sigma\varnothing = (x_0 : \star, y_0 : \star) \qquad \Sigma(\Gamma, x : A) = \Sigma\Gamma, x : \Sigma A$$

$$\Sigma\star = x_0 \underset{\star}{\to} y_0 \qquad \Sigma(t \underset{A}{\to} u) = \Sigma t \underset{\Sigma A}{\longrightarrow} \Sigma u$$

Definition of the suspension

Induction over the syntax: Define the suspension as a meta-operation on the syntax of the type theory.

$$\Sigma\varnothing = (x_0 : \star, y_0 : \star) \qquad \Sigma(\Gamma, x : A) = \Sigma\Gamma, x : \Sigma A$$

$$\Sigma\star = x_0 \underset{\star}{\to} y_0 \qquad \Sigma(t \underset{A}{\to} u) = \Sigma t \underset{\Sigma A}{\longrightarrow} \Sigma u$$

$$\Sigma x = x \qquad \Sigma(\mathrm{coh}_{\Gamma,A}[\gamma]) = \mathrm{coh}_{\Sigma\Gamma,\Sigma A}[\Sigma\gamma]$$

Definition of the suspension

Induction over the syntax: Define the suspension as a meta-operation on the syntax of the type theory.

$$\Sigma\varnothing = (x_0 : \star, y_0 : \star) \qquad \Sigma(\Gamma, x : A) = \Sigma\Gamma, x : \Sigma A$$

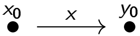
$$\Sigma\star = x_0 \underset{\star}{\rightarrow} y_0 \qquad \Sigma(t \underset{A}{\rightarrow} u) = \Sigma t \underset{\Sigma A}{\longrightarrow} \Sigma u$$

$$\Sigma x = x \qquad \Sigma(\mathsf{coh}_{\Gamma,A}[\gamma]) = \mathsf{coh}_{\Sigma\Gamma,\Sigma A}[\Sigma\gamma]$$

$$\Sigma(\langle\rangle) = \langle x_0 \mapsto x_0, y_0 \mapsto y_0\rangle \qquad \Sigma(\langle\gamma, x \mapsto t\rangle) = \langle\Sigma\gamma, x \mapsto \Sigma t\rangle$$

Illustration

On contexts:

Γ	$\Sigma\Gamma$
	
$(x : \star)$	$(x_0 : \star, y_0 : \star, x : x_0 \underset{\star}{\to} y_0)$

Illustration

On contexts:

Γ	$\Sigma\Gamma$
$\overset{x}{\bullet}$ $(x : \star)$	$\begin{tikzcd} \overset{x_0}{\bullet} \arrow[r, "x"] & \overset{y_0}{\bullet} \end{tikzcd}$ $(x_0 : \star, y_0 : \star, x : x_0 \underset{\star}{\rightarrow} y_0)$
$\begin{tikzcd} \overset{x}{\bullet} \arrow[d, "f"] \\ \underset{y}{\bullet} \end{tikzcd}$ $(x : \star, y : \star, f : x \underset{\star}{\rightarrow} y)$	$\begin{tikzcd} \overset{x_0}{\bullet} \arrow[r, bend left, "x"] \arrow[r, bend right, "y"'] \arrow[r, phantom, "\Downarrow f"] & \overset{y_0}{\bullet} \end{tikzcd}$ $(x_0 : \star, y_0 : \star, x : x_0 \rightarrow y_0, y : x_0 \rightarrow y_0, f : x \rightarrow y)$

Well-definedness of the suspension

$$\Sigma\varnothing = (x_0 : *, y_0 : *) \qquad \Sigma(\Gamma, x : A) = \Sigma\Gamma, x : \Sigma A$$

$$\Sigma\star = x_0 \to y_0 \qquad \Sigma(t \underset{A}{\to} u) = \Sigma t \underset{\Sigma A}{\longrightarrow} \Sigma u$$

$$\Sigma x = x \qquad \Sigma(\mathrm{coh}_{\Gamma,A}[\gamma]) = \mathrm{coh}_{\Sigma\Gamma,\Sigma A}[\Sigma\gamma]$$

$$\Sigma(\langle\rangle) = \langle x_0 \mapsto x_0, y_0 \mapsto y_0\rangle \qquad \Sigma(\langle\gamma, x \mapsto t\rangle) = \langle\Sigma\gamma, x \mapsto \Sigma t\rangle$$

Well-definedness of the suspension

$$\boxed{\begin{array}{rcl@{\qquad}rcl}
\Sigma\varnothing & = & (x_0 : *, y_0 : *) & \Sigma(\Gamma, x : A) & = & \Sigma\Gamma, x : \Sigma A \\
\Sigma\star & = & x_0 \to y_0 & \Sigma(t \underset{A}{\to} u) & = & \Sigma t \underset{\Sigma A}{\longrightarrow} \Sigma u \\
\Sigma x & = & x & \Sigma(\mathsf{coh}_{\Gamma,A}[\gamma]) & = & \mathsf{coh}_{\Sigma\Gamma,\Sigma A}[\Sigma\gamma] \\
\Sigma(\langle\rangle) & = & \langle x_0 \mapsto x_0, y_0 \mapsto y_0 \rangle & \Sigma(\langle\gamma, x \mapsto t\rangle) & = & \langle\Sigma\gamma, x \mapsto \Sigma t\rangle
\end{array}}$$

Theorem (B., Mimram)

The following rules are derivable

$$\frac{\Gamma \vdash}{\Sigma\Gamma \vdash} \qquad \frac{\Gamma \vdash A}{\Sigma\Gamma \vdash \Sigma A} \qquad \frac{\Gamma \vdash t : A}{\Sigma\Gamma \vdash \Sigma t : \Sigma A} \qquad \frac{\Delta \vdash \gamma : \Gamma}{\Sigma\Delta \vdash \Sigma\gamma : \Sigma\Gamma}$$

Well-definedness of the suspension

$$\boxed{\begin{array}{ll} \Sigma\varnothing = (x_0 : *, y_0 : *) & \Sigma(\Gamma, x : A) = \Sigma\Gamma, x : \Sigma A \\ \Sigma\star = x_0 \to y_0 & \Sigma(t \underset{A}{\to} u) = \Sigma t \underset{\Sigma A}{\longrightarrow} \Sigma u \\ \Sigma x = x & \Sigma(\mathsf{coh}_{\Gamma,A}[\gamma]) = \mathsf{coh}_{\Sigma\Gamma,\Sigma A}[\Sigma\gamma] \\ \Sigma(\langle\rangle) = \langle x_0 \mapsto x_0, y_0 \mapsto y_0\rangle & \Sigma(\langle\gamma, x \mapsto t\rangle) = \langle\Sigma\gamma, x \mapsto \Sigma t\rangle \end{array}}$$

Theorem (B., Mimram)

The following rules are derivable

$$\frac{\Gamma \vdash}{\Sigma\Gamma \vdash} \qquad \frac{\Gamma \vdash A}{\Sigma\Gamma \vdash \Sigma A} \qquad \frac{\Gamma \vdash t : A}{\Sigma\Gamma \vdash \Sigma t : \Sigma A} \qquad \frac{\Delta \vdash \gamma : \Gamma}{\Sigma\Delta \vdash \Sigma\gamma : \Sigma\Gamma}$$

Proof.

By induction over the rules of the theory

$\square$

Example: A development using the suspension

```
coh id (x:∗):x->x
coh comp (x:∗)(y:∗)(f:x->y)(z:∗)(g:y->z):x->z
coh unit (x:∗)(y:∗)(f:x->y):comp (id x) f -> f
let unit2 (x:∗)(y:∗)(f:x->y)(g:x->y)(a:f->g)=unit a
```

Example: A development using the suspension

```
coh id (x:*):x->x
coh comp (x:*)(y:*)(f:x->y)(z:*)(g:y->z):x->z
coh unit (x:*)(y:*)(f:x->y):comp (id x) f -> f
let unit2 (x:*)(y:*)(f:x->y)(g:x->y)(a:f->g)=unit a
```

Internally:

Example: A development using the suspension

```
coh id (x:*):x->x
coh comp (x:*)(y:*)(f:x->y)(z:*)(g:y->z):x->z
coh unit (x:*)(y:*)(f:x->y):comp (id x) f -> f
let unit2 (x:*)(y:*)(f:x->y)(g:x->y)(a:f->g)=unit a
```

Internally:

- The system computes the difference in the dimension of the argument
  `unit` expects an argument of dimension 1,a is of dimension 2 $\rightsquigarrow$ suspend once

Example: A development using the suspension

```
coh id (x:*):x->x
coh comp (x:*)(y:*)(f:x->y)(z:*)(g:y->z):x->z
coh unit (x:*)(y:*)(f:x->y):comp (id x) f -> f
let unit2 (x:*)(y:*)(f:x->y)(g:x->y)(a:f->g)=unit a
```

Internally:

- The system computes the difference in the dimension of the argument
  `unit` expects an argument of dimension 1,`a` is of dimension 2 $\rightsquigarrow$ suspend once
- Compute the suspension of `unit`

Example: A development using the suspension

```
coh id (x:∗):x->x
coh comp (x:∗)(y:∗)(f:x->y)(z:∗)(g:y->z):x->z
coh unit (x:∗)(y:∗)(f:x->y):comp (id x) f -> f
let unit2 (x:∗)(y:∗)(f:x->y)(g:x->y)(a:f->g)=unit a
```

Internally:

- The system computes the difference in the dimension of the argument
  `unit` expects an argument of dimension 1,a is of dimension 2 ⇝ suspend once
- Compute the suspension of `unit`
- Compute the suspension of `comp`

Example: A development using the suspension

```
coh id (x:*):x->x
coh comp (x:*)(y:*)(f:x->y)(z:*)(g:y->z):x->z
coh unit (x:*)(y:*)(f:x->y):comp (id x) f -> f
let unit2 (x:*)(y:*)(f:x->y)(g:x->y)(a:f->g)=unit a
```

Internally:

- The system computes the difference in the dimension of the argument
  `unit` expects an argument of dimension 1,a is of dimension 2 ⇝ suspend once
- Compute the suspension of `unit`
- Compute the suspension of `comp`
- compute the suspension of `id`

A categorical view on the suspension

Recall:

- Information about the derivable syntax is stored in the syntactic category
- The suspension defined on the syntax respects the derivability

A categorical view on the suspension

Recall:

- ▷ Information about the derivable syntax is stored in the syntactic category
- ▷ The suspension defined on the syntax respects the derivability

↳ The suspension lifts on the category syntactic

A categorical view on the suspension

Recall:

- ▷ Information about the derivable syntax is stored in the syntactic category
- ▷ The suspension defined on the syntax respects the derivability

↳ The suspension lifts on the category syntactic

$$\mathcal{S}_{\mathsf{CaTT}} \xrightarrow{\Sigma()} \mathcal{S}_{\mathsf{CaTT}}$$

$\Sigma()$ is a morphism of CwF

A categorical view on the suspension

Recall:

- ▷ Information about the derivable syntax is stored in the syntactic category
- ▷ The suspension defined on the syntax respects the derivability

↳ The suspension lifts on the category syntactic

$$\mathcal{S}_{\mathsf{CaTT}} \xrightarrow{\Sigma()} \mathcal{S}_{\mathsf{CaTT}}$$

$\Sigma()$ is a morphism of CwF

See the semantics

Another example: The functorialization

The functorialization is another meta-operation on the syntax

Another example: The functorialization

The functorialization is another meta-operation on the syntax

It translates another property of the theory of weak ω-categories
The operations in a higher category define functors

Another example: The functorialization

The functorialization is another meta-operation on the syntax

It translates another property of the theory of weak ω-categories
The operations in a higher category define functors

Problem: This operation is not always well-defined. It only defines a partial endomorphism of CwFs

Semantics

The Grothendieck-Maltsiniotis definition of ω-categories

- Define the category Θ_0 of pasting schemes
  Characterized as colimits: globular sums

- Complete it into the category Θ_∞
  Add lift conditions, while preserving globular sums. The extension $\Theta_0 \hookrightarrow \Theta_\infty$ defined by universal property

- The weak ω-categories are functors $\Theta_\infty \to \mathrm{Set}$ which preserve the globular sums

A categorical view of ps-contexts

Define the full subcategory $\mathcal{S}_{\mathsf{PS}} \hookrightarrow \mathcal{S}_{\mathsf{CaTT}}$:

- objects=ps-contexts

A categorical view of ps-contexts

Define the full subcategory $\mathcal{S}_{\mathsf{PS}} \hookrightarrow \mathcal{S}_{\mathsf{CaTT}}$:

- objects=ps-contexts

Theorem (Finster, Mimram)

There is an equivalence of categories $\Theta_0^{\mathrm{op}} \simeq \mathcal{S}_{\mathsf{PS},0}$

A categorical view of ps-contexts

Define the full subcategory $\mathcal{S}_{\mathsf{PS}} \hookrightarrow \mathcal{S}_{\mathsf{CaTT}}$:

- ▷ objects=ps-contexts

Theorem (Finster, Mimram)

There is an equivalence of categories $\Theta_0^{\mathrm{op}} \simeq \mathcal{S}_{\mathsf{PS},0}$

Note: $\mathcal{S}_{\mathsf{PS},0}$ is the category $\mathcal{S}_{\mathsf{PS}}$ removing all non-variable terms

Ps-contexts with coherences

Theorem (B., Finster, Mimram)

There is an equivalence of categories $\Theta_\infty^{\mathsf{op}} \simeq \mathcal{S}_{\mathsf{PS}}$

Proof.

- ▷ The inclusion $\mathcal{S}_{\mathsf{PS},0} \to \mathcal{S}_{\mathsf{PS}}$ preserves the globular products
- ▷ The inclusion $\mathcal{S}_{\mathsf{PS},0} \to \mathcal{S}_{\mathsf{PS}}$ satisfies the universal property dual of the inclusion $\Theta_0 \to \Theta_\infty$

□

The syntactic category $\mathcal{S}_{\mathsf{CaTT}}$

Theorem (B., Finster, Mimram)

The inclusion functor $\mathcal{S}_{\mathsf{PS}} \hookrightarrow \mathcal{S}_{\mathsf{CaTT}}$ *is codense*

The syntactic category $\mathcal{S}_{\mathsf{CaTT}}$

Theorem (B., Finster, Mimram)

The inclusion functor $\mathcal{S}_{\mathsf{PS}} \hookrightarrow \mathcal{S}_{\mathsf{CaTT}}$ *is codense*

The syntactic category $\mathcal{S}_{\mathsf{CaTT}}$

Theorem (B., Finster, Mimram)

The inclusion functor $\mathcal{S}_{\mathsf{PS}} \hookrightarrow \mathcal{S}_{\mathsf{CaTT}}$ *is codense*

Every context is canonically a limit of ps-contexts

A universal property of the extension

Corollary

The inclusion functor $\mathcal{S}_{\mathsf{PS}} \hookrightarrow \mathcal{S}_{\mathsf{CaTT}}$ *induces an equivalence of categories*

$$[\mathcal{S}_{\mathsf{CaTT}}, \mathrm{Set}]_{\textit{canonical limits}} \simeq [\mathcal{S}_{\mathsf{PS}}, \mathrm{Set}]_{\textit{globular products}}$$

$$\begin{tikzcd}
\mathcal{S}_{\mathsf{CaTT}} \arrow[r, "\text{canonical limits}"] & \mathrm{Set} \\
\mathcal{S}_{\mathsf{PS}} \arrow[u, hook] \arrow[ur, dotted, "\exists!\ \text{globular products}"']
\end{tikzcd}$$

The models of CaTT

Theorem (B., Finster, Mimram)

The models of CaTT are equivalent to the weak ω-categories

The models of CaTT

Theorem (B., Finster, Mimram)

The models of CaTT are equivalent to the weak ω-categories

Proof.

$$[\mathcal{S}_{\mathsf{CaTT}}, \mathrm{Set}]_{\mathsf{CwF}} \simeq [\mathcal{S}_{\mathsf{CaTT}}, \mathrm{Set}]_{\mathsf{canonical\ limits}}$$

☐

The models of CaTT

Theorem (B., Finster, Mimram)

The models of CaTT are equivalent to the weak ω-categories

Proof.

$$\begin{aligned}[\mathcal{S}_{\mathsf{CaTT}}, \mathrm{Set}]_{\mathsf{CwF}} &\simeq [\mathcal{S}_{\mathsf{CaTT}}, \mathrm{Set}]_{\text{canonical limits}} \\ &\simeq [\mathcal{S}_{\mathsf{PS}}, \mathrm{Set}]_{\text{globular products}}\end{aligned}$$

☐

The models of CaTT

Theorem (B., Finster, Mimram)

The models of CaTT are equivalent to the weak ω-categories

Proof.

$$\begin{aligned}[\mathcal{S}_{\mathsf{CaTT}}, \mathrm{Set}]_{\mathsf{CwF}} &\simeq [\mathcal{S}_{\mathsf{CaTT}}, \mathrm{Set}]_{\text{canonical limits}} \\ &\simeq [\mathcal{S}_{\mathsf{PS}}, \mathrm{Set}]_{\text{globular products}} \\ &\simeq [\Theta_\infty^{\mathsf{op}}, \mathrm{Set}]_{\text{globular sums}}\end{aligned}$$

☐

Thank you!