CaTT: A type theoretic approach to weak ω-categories

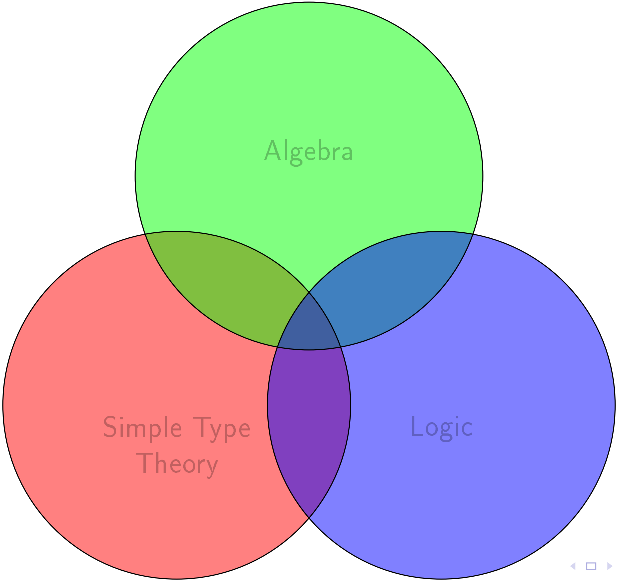
Thibaut Benjamin, Eric Finster, Samuel Mimram

15 February 2021

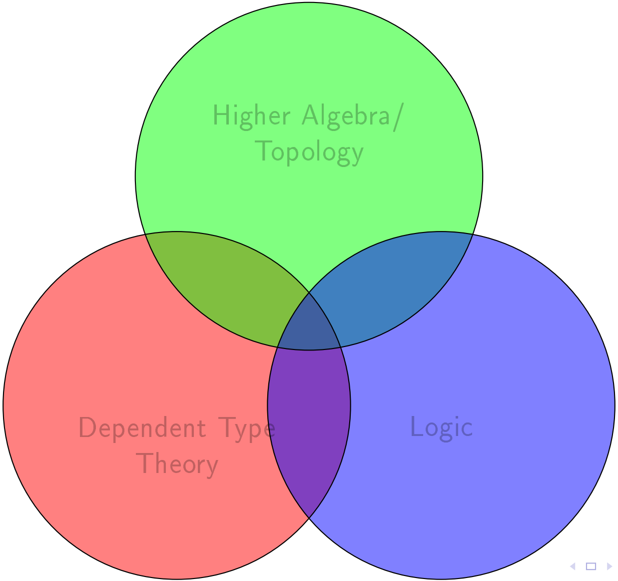
Seminaire LACL

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

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

compositions

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

Motivations: A bit of category theory

Categories are made of

objects

$\bullet$

arrows

$\bullet \longrightarrow \bullet$

Equipped with

identities

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

compositions

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

Satisfying

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

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

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

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

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

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
- A rewriting system is a category : objects = terms, morphisms = rewriting relations.

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

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

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

$$\bullet \longrightarrow \bullet$$

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

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

Equipped with

identities

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

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

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

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

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

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

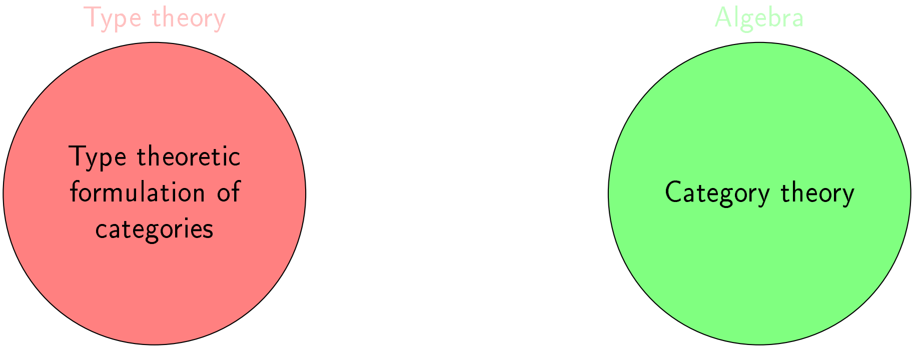
unitality

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

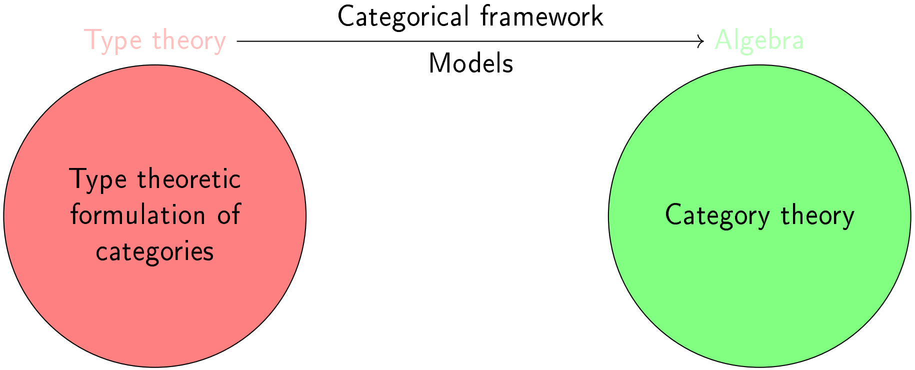
associativity

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

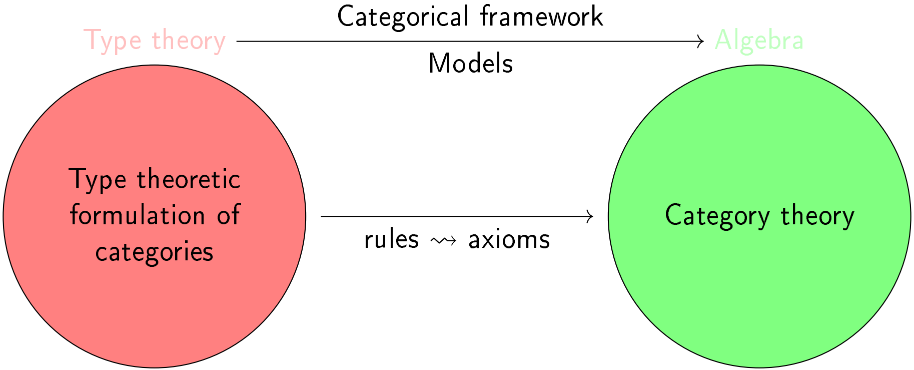
Semantics


Type theory
Type theoretic formulation of categories
Algebra
Category theory

Semantics


Type theory
Categorical framework
Models
Algebra
Type theoretic formulation of categories
Category theory

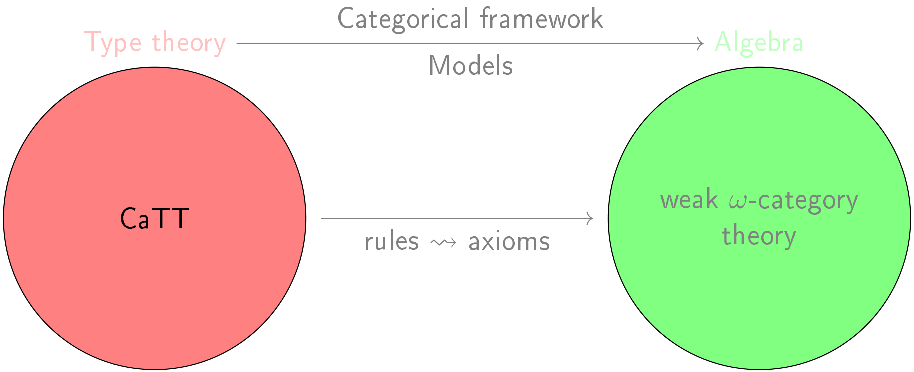
Semantics



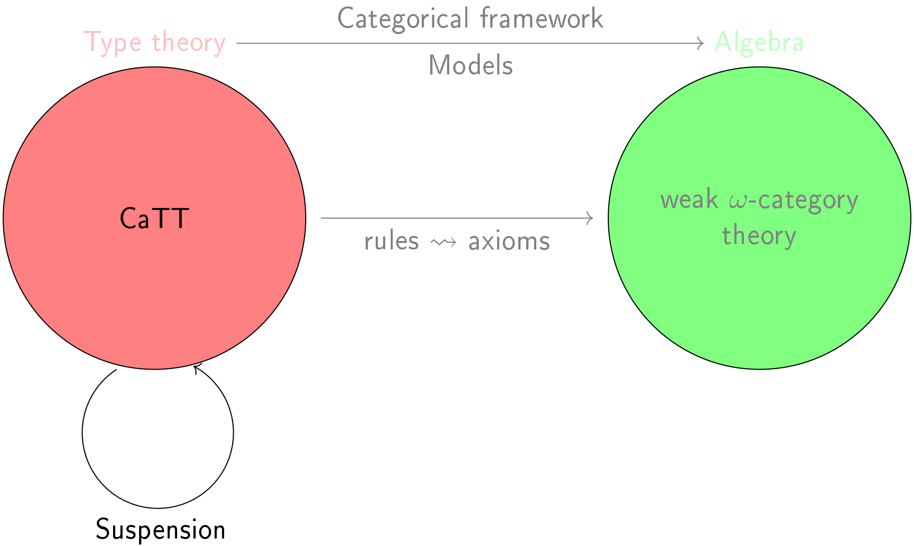
Theorem

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

In this presentation



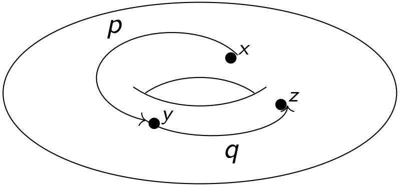
In this presentation



Motivations: When categories are insufficient

Some situation that ought to be categories

- paths in topological spaces



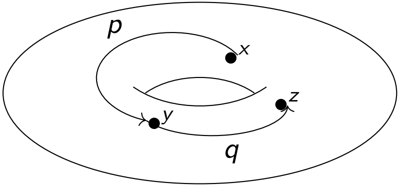
But: The composition is not associative

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

Motivations: When categories are insufficient

Some situation that ought to be categories

- paths in topological spaces



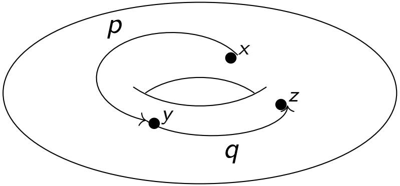
The composition is associative up to homotopy

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

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

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

$$\texttt{trans}\ A\ x\ x\ x\ \texttt{refl}\ \texttt{refl}\ \texttt{:=}\ \texttt{refl}$$

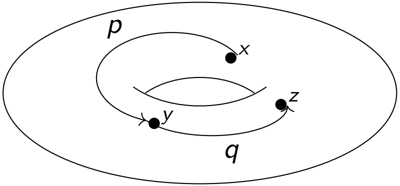
But: Transitivity is not associative

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

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

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

There is a proof term for associativity

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

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

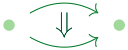
- *points* (or *objects*) : •

A globular set is composed of

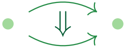
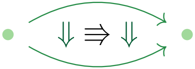
- *points* (or *objects*) : $\bullet$
- *arrows* (or 1-*cells*) : $\bullet \longrightarrow \bullet$

Globular sets

A globular set is composed of

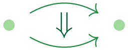
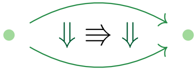
- *points* (or *objects*) : $\bullet$
- *arrows* (or 1-*cells*) : $\bullet \longrightarrow \bullet$
- 2-*cells* : 

A globular set is composed of

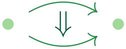
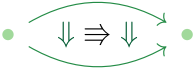
- *points* (or *objects*) : 
- *arrows* (or 1-*cells*) : 
- 2-*cells* : 
- 3-*cells* : 

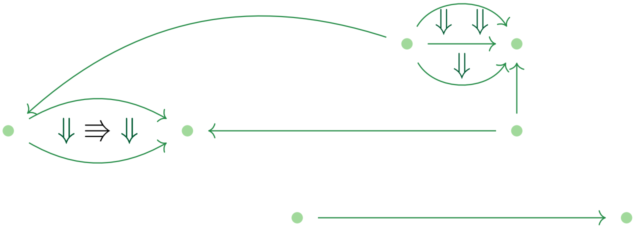
Globular sets

A globular set is composed of

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

A globular set is composed of

- *points* (or *objects*) : 
- *arrows* (or 1-*cells*) : 
- 2-*cells* : 
- 3-*cells* : 
- etc



With compositions

We require that these cells compose

With compositions

We require that these cells compose

- Arrows:

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

With compositions

We require that these cells compose

- Arrows:

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

With compositions

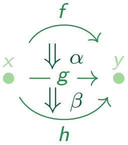
We require that these cells compose

- Arrows:

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

- 2-cells:

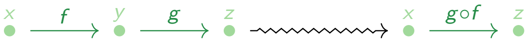
Vertical composition



With compositions

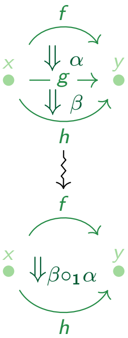
We require that these cells compose

- Arrows:



- 2-cells:

Vertical composition



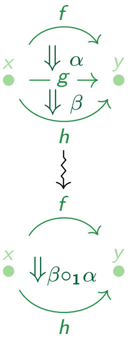
We require that these cells compose

- Arrows:

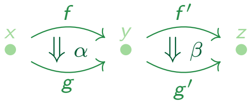
$$x \xrightarrow{f} y \xrightarrow{g} z \rightsquigarrow x \xrightarrow{g \circ f} z$$

- 2-cells:

Vertical composition



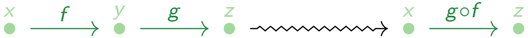
Horizontal composition



With compositions

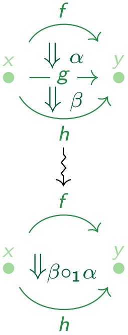
We require that these cells compose

- Arrows:

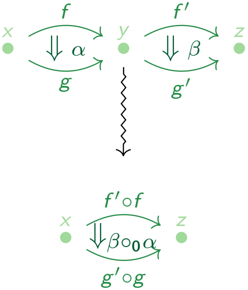


- 2-cells:

Vertical composition



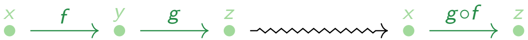
Horizontal composition



With compositions

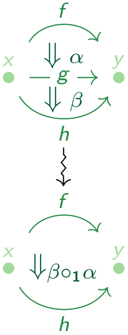
We require that these cells compose

- Arrows:

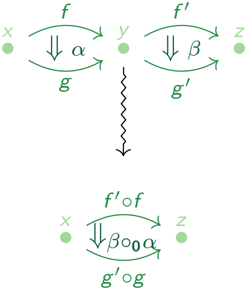


- 2-cells:

Vertical composition



Horizontal composition



- etc

Satisfying associativity

The compositions are required to be associative

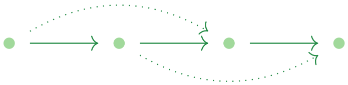
Satisfying associativity

The compositions are required to be associative

- Arrows:

Satisfying associativity

The compositions are required to be associative

- Arrows: 

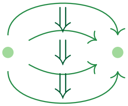
Satisfying associativity

The compositions are required to be associative

- Arrows:

- 2-cells:

Vertical composition



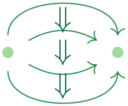
Satisfying associativity

The compositions are required to be associative

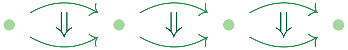
- Arrows:

- 2-cells:

Vertical composition



Horizontal composition



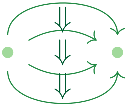
Satisfying associativity

The compositions are required to be associative

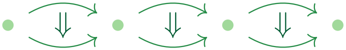
- Arrows:

- 2-cells:

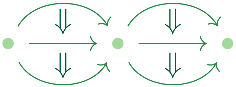
Vertical composition



Horizontal composition



Exchange law



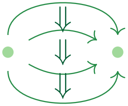
Satisfying associativity

The compositions are required to be associative

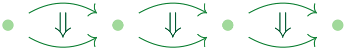
- Arrows:

- 2-cells:

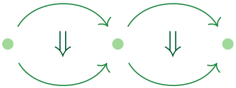
Vertical composition



Horizontal composition



Exchange law



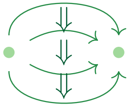
Satisfying associativity

The compositions are required to be associative

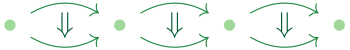
- Arrows:

- 2-cells:

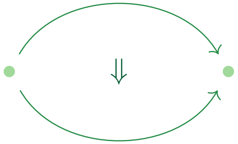
Vertical composition



Horizontal composition



Exchange law



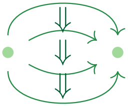
Satisfying associativity

The compositions are required to be associative

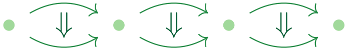
- Arrows:

- 2-cells:

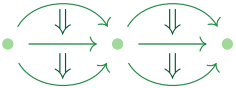
Vertical composition



Horizontal composition



Exchange law



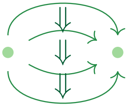
Satisfying associativity

The compositions are required to be associative

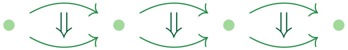
- Arrows:

- 2-cells:

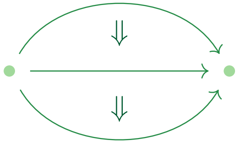
Vertical composition



Horizontal composition



Exchange law



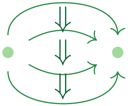
Satisfying associativity

The compositions are required to be associative

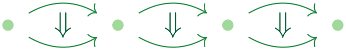
- Arrows:

- 2-cells:

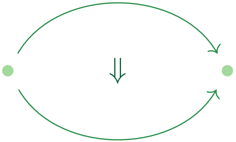
Vertical composition



Horizontal composition



Exchange law



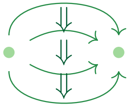
Satisfying associativity

The compositions are required to be associative

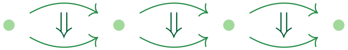
- ▷ Arrows:

- ▷ 2-cells:

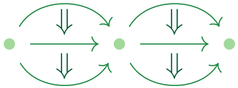
Vertical composition



Horizontal composition



Exchange law



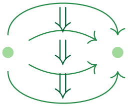
Satisfying associativity

The compositions are required to be associative

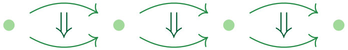
- Arrows:

- 2-cells:

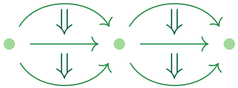
Vertical composition



Horizontal composition



Exchange law



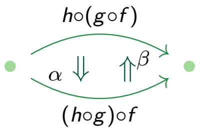
- etc

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



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

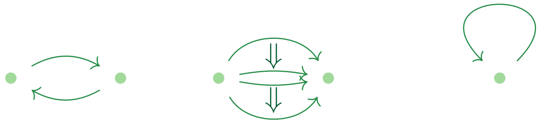
Examples

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 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

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

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
```

Return type

Globular set

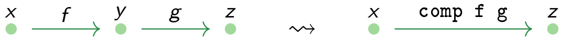
Name of the operation

`comp`

Keyword: declaration of an operation

First Examples

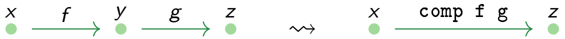
```
coh comp (x:*)(y:*)(f:x->y)(z:*)(g:y->z): x->z
```



Composition of two arrows

First Examples

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

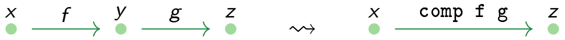


Composition of two arrows

```
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 $\to$ for arrows:

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

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

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

- etc

Types for cells

In order to manipulate ω-categories, we need:

- a type $\star$ for objects:

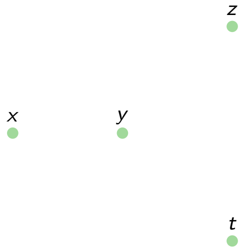
$$\frac{\Gamma \vdash}{\Gamma \vdash \star}$$

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

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

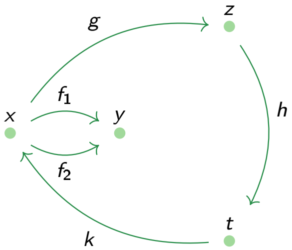
Contexts are diagrams!

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



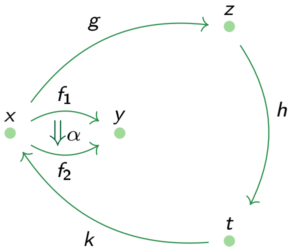
Contexts are diagrams!

$$\Gamma \begin{cases} x : \star, \quad y : \star, \quad z : \star, \quad t : \star \\ f_1 : x \underset{\star}{\rightarrow} y, \quad f_2 : x \underset{\star}{\rightarrow} y, \quad g : x \underset{\star}{\rightarrow} z, \quad h : z \underset{\star}{\rightarrow} t, \quad k : t \underset{\star}{\rightarrow} x \end{cases}$$



Contexts are diagrams!

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



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 $\Gamma \vdash_{\mathsf{ps}}$

- This judgment is decidable with an algorithm

$$\frac{}{x : \star \vdash_{\mathsf{ps}} x : \star} \qquad \frac{\Gamma \vdash_{\mathsf{ps}} x : A}{\Gamma, y : A, f : x \underset{A}{\longrightarrow} y \vdash_{\mathsf{ps}} f : x \underset{A}{\longrightarrow} y}$$

$$\frac{\Gamma \vdash_{\mathsf{ps}} f : x \underset{A}{\longrightarrow} y}{\Gamma \vdash_{\mathsf{ps}} y : A} \qquad \frac{\Gamma \vdash_{\mathsf{ps}} x : \star}{\Gamma \vdash_{\mathsf{ps}}}$$

Source and target

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

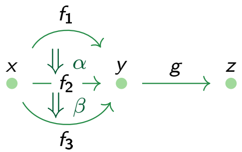
Intuitively, composing fully the ps-context yields a cell from its source to its target

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



Example

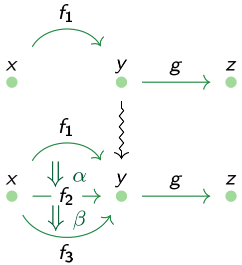
Source and target

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

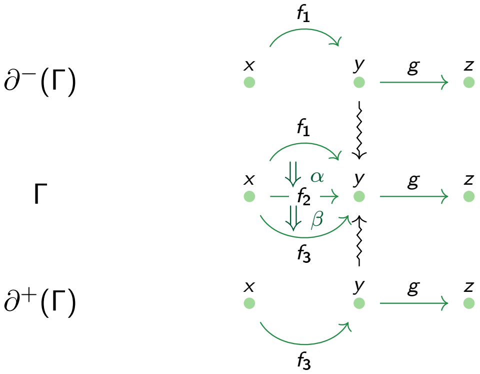


Example

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 $\Gamma \vdash_{\mathrm{ps}}$:

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

  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:

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

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

Operations and coherences

- Existence of compositions:
  Every pasting scheme can be composed

Rule for generating these compositions:

$$\frac{\Gamma \vdash_{\mathsf{ps}} \qquad \partial^-(\Gamma) \vdash t : A \qquad \partial^+(\Gamma) \vdash u : A}{\Gamma \vdash \mathsf{op}_{\Gamma, t \underset{A}{\to} u} : t \underset{A}{\to} u} \qquad \begin{array}{l} \mathsf{Var}(t : A) = \mathsf{Var}(\partial^-(\Gamma)) \\ \mathsf{Var}(u : A) = \mathsf{Var}(\partial^+(\Gamma)) \end{array}$$

$$\Gamma = x : \star, y : \star, f : x \underset{\star}{\to} y, z : \star, g : y \underset{\star}{\to} z$$
$$\Gamma \vdash \mathtt{comp} : x \to z$$

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:

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

- 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:

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

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

$$\Gamma \vdash \mathsf{assoc} : \texttt{comp f (comp g h)} \to \texttt{comp (comp f g) h}$$

Example

Suspension in CaTT

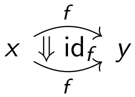
A problem with CaTT

Writing in CaTT may be very tedious :

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

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

$$x \xrightarrow{f} y \rightsquigarrow$$



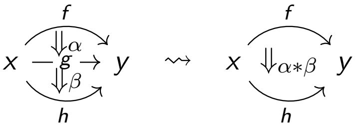

```
coh id2 (x:*)(y:*)(f:x->y):f->f
```

A problem with CaTT

Writing in CaTT may be very tedious :

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

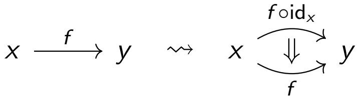
```
coh comp (x:*)(y:*)(f:x->y)
    (z:*)(g:y->z):x->z
```



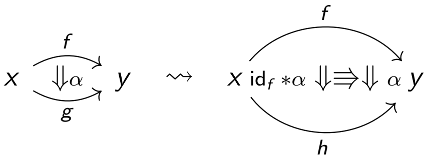

```
coh vcomp (x:*)(y:*)(f:x->y)(g:x->y)
    (a:f->g)(h:x->y)(b:g->h):f->h
```

A problem with CaTT

Writing in CaTT may be very tedious :




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




```
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
```

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.

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

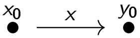
$$\Sigma\star = x_0 \underset{\star}{\to} y_0 \qquad \Sigma(t \underset{A}{\to} u) = \Sigma t \underset{\Sigma A}{\longrightarrow} \Sigma u$$

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

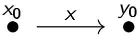
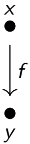
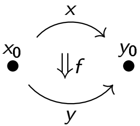
$$\Sigma(\langle\rangle) = \langle x_0 \mapsto x_0, y_0 \mapsto y_0\rangle \qquad \Sigma(\langle\gamma, x \mapsto t\rangle) = \langle\Sigma\gamma, x \mapsto \Sigma t\rangle$$

Illustration

On contexts:

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

On contexts:

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

Well-definedness of the suspension

$\Sigma\varnothing = (x_0 : *, y_0 : *)$	$\Sigma(\Gamma, x : A) = \Sigma\Gamma, x : \Sigma A$
$\Sigma\star = x_0 \to y_0$	$\Sigma(t \underset{A}{\to} u) = \Sigma t \underset{\Sigma A}{\longrightarrow} \Sigma u$
$\Sigma x = x$	$\Sigma(\mathrm{coh}_{\Gamma,A}[\gamma]) = \mathrm{coh}_{\Sigma\Gamma,\Sigma A}[\Sigma\gamma]$
$\Sigma(\langle\rangle) = \langle x_0 \mapsto x_0, y_0 \mapsto y_0\rangle$	$\Sigma(\langle\gamma, x \mapsto t\rangle) = \langle\Sigma\gamma, x \mapsto \Sigma t\rangle$

Well-definedness of the suspension

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

Theorem (B., Mimram)

The following rules are derivable

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

Well-definedness of the suspension

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

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

□

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 $\rightsquigarrow$ 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 $\rightsquigarrow$ suspend once
- Compute the suspension of `unit`
- Compute the suspension of `comp`
- compute the suspension of `id`

A categorical view on the suspension

- One can define the syntactic category $\mathcal{S}_{\mathsf{CaTT}}$
  - Objects = contexts
  - Morphisms = substitutions

A categorical view on the suspension

- One can define the syntactic category $\mathcal{S}_{\mathsf{CaTT}}$
  - Objects = contexts
  - Morphisms = substitutions

- The suspension defines an endofunctor

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

A categorical view on the suspension

- One can define the syntactic category $\mathcal{S}_{\mathsf{CaTT}}$
  - Objects = contexts
  - Morphisms = substitutions

- The suspension defines an endofunctor

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

- $\mathcal{S}_{\mathsf{CaTT}}$ is equipped with a structure of category with families, inherited from the type theory

A categorical view on the suspension

- One can define the syntactic category $\mathcal{S}_{\mathsf{CaTT}}$
  - Objects = contexts
  - Morphisms = substitutions

- The suspension defines an endofunctor

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

- $\mathcal{S}_{\mathsf{CaTT}}$ is equipped with a structure of category with families, inherited from the type theory

- The suspenson is a morphsim of category with families.

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

Thank you!