Towards a fully formalised proof of the syllepsis in triply monoidal ω-categories

Samuel Mimram, Eric Finster, Thibaut Benjamin

Ecole Polytechnique

07/07/2018

Goal

Give a formalisation of the definition of syllepsis in triply monoidal ω-categories, using the type theory CaTT defined by Finster and Mimram

The implementation we are using is available at `https://github.com/ThiBen/catt`

Motivations

- First proof for which the prover is needed

expected, but not proven result

Motivations

- First proof for which the prover is needed
- Extensive test of the implementation

various bugs could stay unnoticed on smaller projects

Motivations

- First proof for which the prover is needed
- Extensive test of the implementation
- Understanding of the practical use of CaTT

will guide the future developments

Outline

Type: globular sets

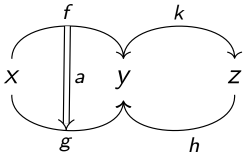
- To start with, we have a type theory with two type constructors

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

- Contexts of this theory describe globular sets in the following sense

```
x:*, y:*, z:*,
f:x->y, g:x->y,
h:z->y, k:y->z,
a:f->g
```

$\simeq$



Terms: weak ω-categories

- Operations and axioms of categories are generated using terms constructors

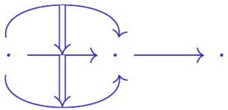
Terms: weak ω-categories

- Operations and axioms of categories are generated using terms constructors
- Pasting scheme

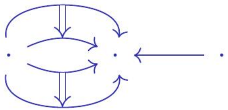
Definition

Pasting scheme are the globular sets that are in a "generic composable position"

Example



a pasting scheme



not a pasting scheme

Terms: weak ω-categories

- Operations and axioms of categories are generated using terms constructors
- Pasting scheme
- Coherence: operation of composing a pasting scheme

Examples

Operation	Pasting scheme	Result
Identity	x	$x \to x$

Terms: weak ω-categories

- Operations and axioms of categories are generated using terms constructors
- Pasting scheme
- Coherence: operation of composing a pasting scheme

Examples

Operation	Pasting scheme	Result
Identity	x	$x \to x$
Composition	$x \xrightarrow{f} y \xrightarrow{g} z$	$x \to z$

Terms: weak ω-categories

- Operations and axioms of categories are generated using terms constructors
- Pasting scheme
- Coherence: operation of composing a pasting scheme

Examples

Operation	Pasting scheme	Result
Identity	x	$x \to x$
Composition	$x \xrightarrow{f} y \xrightarrow{g} z$	$x \to z$
Associativity	$x \xrightarrow{f} y \xrightarrow{g} z \xrightarrow{h} w$	$f \star (g \star h) \to (f \star g) \star h$

Terms: weak ω-categories

- Operations and axioms of categories are generated using terms constructors
- Pasting scheme
- Coherence: operation of composing a pasting scheme

Examples

Operation	Pasting scheme	Result
• Identity	x	$x \to x$
Composition	$x \xrightarrow{f} y \xrightarrow{g} z$	$x \to z$
Associativity	$x \xrightarrow{f} y \xrightarrow{g} z \xrightarrow{h} w$	$f \star (g \star h) \to (f \star g) \star h$

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

Terms: weak ω-categories

- Operations and axioms of categories are generated using terms constructors
- Pasting scheme
- Coherence: operation of composing a pasting scheme

Examples

Operation	Pasting scheme	Result
Identity	x	$x \to x$
• Composition	$x \xrightarrow{f} y \xrightarrow{g} z$	$x \to z$
Associativity	$x \xrightarrow{f} y \xrightarrow{g} z \xrightarrow{h} w$	$f \star (g \star h) \to (f \star g) \star h$

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

Terms: weak ω-categories

- Operations and axioms of categories are generated using terms constructors
- Pasting scheme
- Coherence: operation of composing a pasting scheme

Examples

Operation	Pasting scheme	Result
Identity	x	$x \to x$
Composition	$x \xrightarrow{f} y \xrightarrow{g} z$	$x \to z$
• Associativity	$x \xrightarrow{f} y \xrightarrow{g} z \xrightarrow{h} w$	$f \star (g \star h) \to (f \star g) \star h$

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

Terms: weak ω-categories

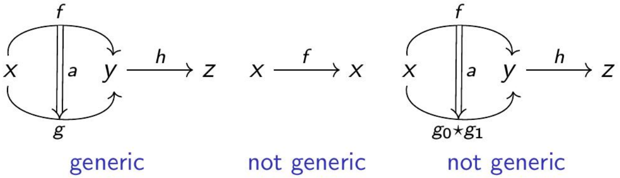
- Operations and axioms of categories are generated using terms constructors
- Pasting scheme
- Coherence: operation of composing a pasting scheme
- Checking coherences
  We add term constructor to ensure coherences correspond to correct operations of weak ω-category theory.

$$\frac{\Gamma \text{ pasting scheme} \qquad \Gamma \vdash u \to v}{\Gamma \vdash \mathsf{coh}_{\Gamma,u,v} : u \to v} \qquad (+\text{ extra conditions})$$

Terms: weak ω-categories

- Operations and axioms of categories are generated using terms constructors
- Pasting scheme
- Coherence: operation of composing a pasting scheme
- Checking coherences
- Note: Genericity

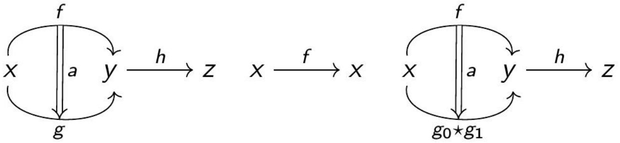
Example



Terms: weak ω-categories

- Operations and axioms of categories are generated using terms constructors
- Pasting scheme
- Coherence: operation of composing a pasting scheme
- Checking coherences
- Note: Genericity

Example



generic not generic not generic

```
coh sq (x : *) (f : x -> x) : x -> x
```

Some metatheoretical principles

- Implicit arguments

Example

Operation	Pasting scheme	Result
Associativity	$x \xrightarrow{f} y \xrightarrow{g} z \xrightarrow{h} w$	$f \star (g \star h) \to (f \star g) \star h$

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

Some metatheoretical principles

- Implicit arguments

Example

Operation	Pasting scheme	Result
Associativity	$x \xrightarrow{f} y \xrightarrow{g} z \xrightarrow{h} w$	$f \star (g \star h) \to (f \star g) \star h$

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

Some metatheoretical principles

- Implicit arguments
- Non-primitive operations

Example (squaring an endomorphism)

```
coh sq (x : *) (f : x -> x) : x -> x
```

Some metatheoretical principles

- Implicit arguments
- Non-primitive operations

Example (squaring an endomorphism)

```
coh sq (x : *) (f : x -> x) : x -> x
let sq (x : *) (f : x -> x) = comp f f
```

Some metatheoretical principles

- Implicit arguments
- Non-primitive operations
- Lifting of operations

Example

```
coh id0 (x : *) : x -> x

coh id1 (x : *) (y : *) (f : x -> y): f -> f

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

Some metatheoretical principles

- Implicit arguments
- Non-primitive operations
- Lifting of operations

Solution
Declare only

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

and generate all the others on the fly

Outline

doubly monoidal ω-categories

Definition

A doubly monoidal ω-category is a weak ω-category with only one object, and one 1-cell

doubly monoidal ω-categories

Definition

A doubly monoidal ω-category is a weak ω-category with only one object, and one 1-cell

In CaTT:

- No way to specify that there is only one of a kind

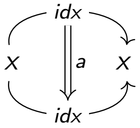
doubly monoidal ω-categories

Definition

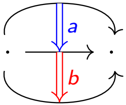
A doubly monoidal ω-category is a weak ω-category with only one object, and one 1-cell

In CaTT:

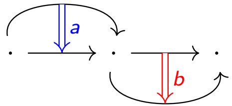
- No way to specify that there is only one of a kind
- Nevertheless, we can restrict ourselves to using only one object and one 1-cell



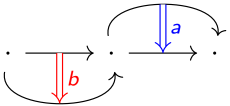
Definition of the braiding



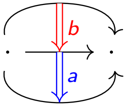
Definition of the braiding



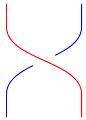
Definition of the braiding



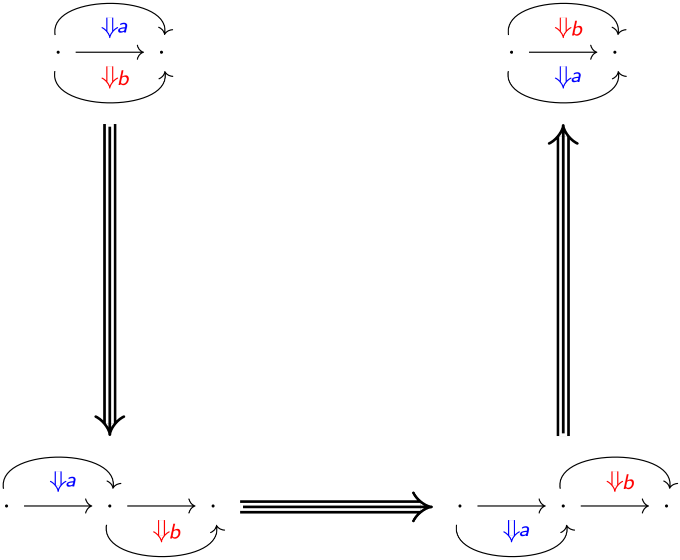
Definition of the braiding



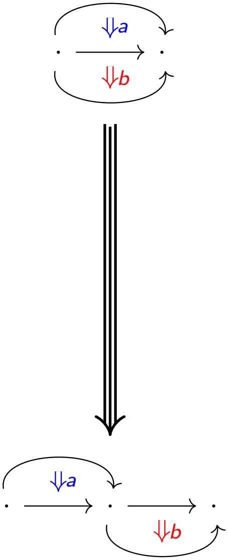
Definition of the braiding



Formalisation - First Plan

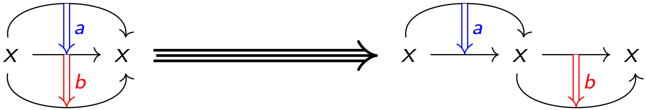


Formalisation - Attempt



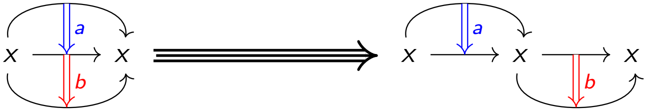
Formalisation Attempt

Translate these successive diagrams into terms of the theory



Formalisation Attempt

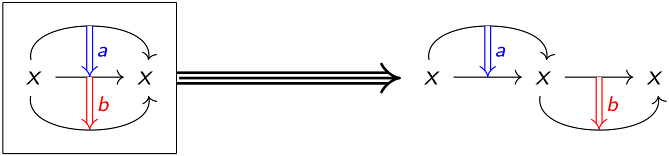
Translate these successive diagrams into terms of the theory



$\Gamma =$`(x : *) (a : id x -> id x) (b : id x -> id x)`

Formalisation Attempt

Translate these successive diagrams into terms of the theory

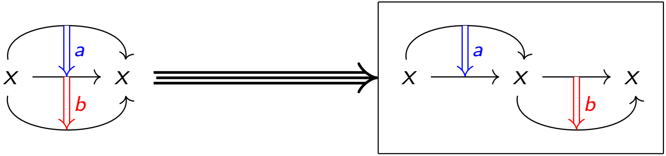


$\Gamma =$`(x : *) (a : id x -> id x) (b : id x -> id x)`

comp a b

Formalisation Attempt

Translate these successive diagrams into terms of the theory



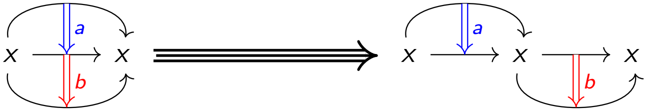
$\Gamma =$`(x : *) (a : id x -> id x) (b : id x -> id x)`

comp a b

comp (whiskR a (id x)) (whiskL (id x) b)

Formalisation Attempt

Translate these successive diagrams into terms of the theory



$\Gamma =$`(x : *) (a : id x -> id x) (b : id x -> id x)`

comp a b $\Longrrightarrow$ comp (whiskR a (id x)) (whiskL (id x) b)

Formalisation Attempt - First step

comp a b $\Rrightarrow$ comp (whiskR a (id x)) (whiskL (id x) b)

Formalisation Attempt - First step

$$\texttt{a} \Longrrightarrow \text{whiskR a (id x)}$$

Idea

Handle separately `a` and `b`

Formalisation Attempt - First step

a

whiskR a (id y)

To apply a coherence, we unfold into a generic position

```
(x : *) (y : *) (f : x -> y)
                (g : x -> y) (a : f -> g)
```

Formalisation Attempt - First step

a

whiskR a (id y)

Try a coherence

```
coh shiftL (x : *) (y : *) (f : x -> y)
                                (g : x -> y) (a : f -> g) :
    a -> whiskR a (id y)
```

Formalisation Attempt - First step

a

whiskR a (id y)

Result

This returns an error: the type of the result is ill-formed

```
a : f -> g

whiskR a (id y) : comp f (id y) -> comp g (id y)
```

Formalisation Attempt - First step

a

comp3 ...

Solution

Modify the target of the arrow

Formalisation Attempt - First step

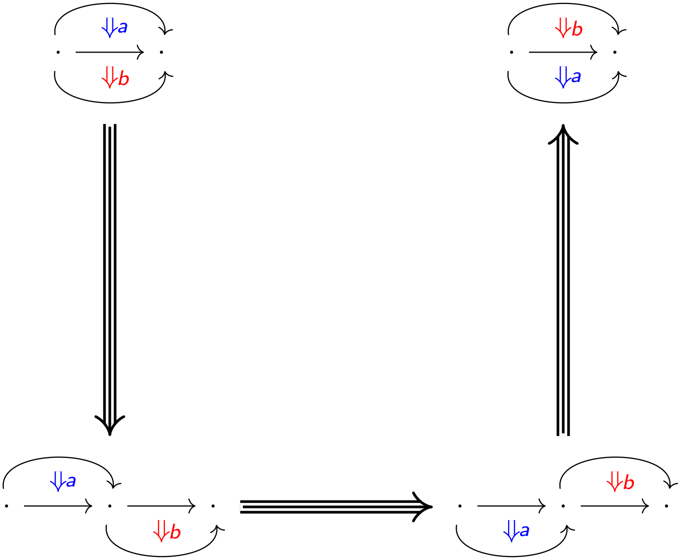
a comp3 ...

Solution

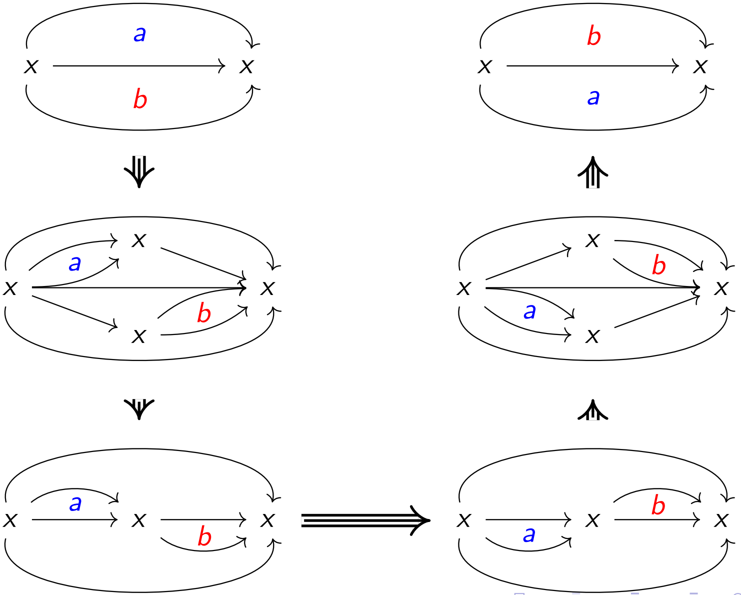
We now have a coherence

```
coh shiftL (x : *) (y : *) (f : x -> y)
                          (g : x -> y) (a : f -> g) :
   a -> comp3 ...
```

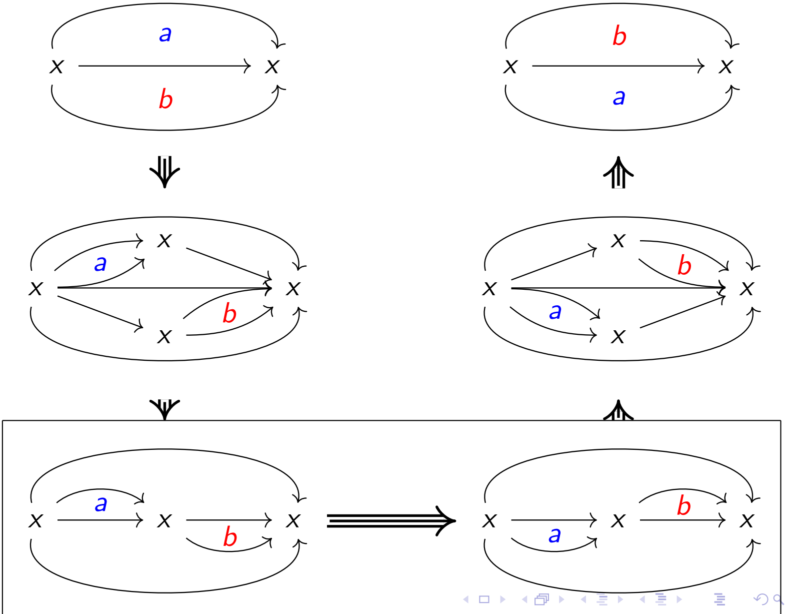
Formalisation - First Plan



Formalisation - Plan

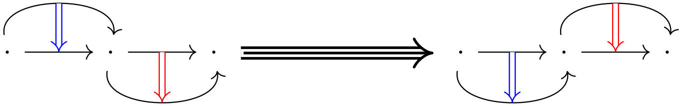


Formalisation - Core



Formalisation - Core

- Exchange law :



- This is a primitive operation of the theory

```
coh exch (x : *) (y : *) (f : x -> y) (g : x -> y)
                                (a : f -> g)
                (z : *) (h : y -> z) (k : y -> z)
                                (b : h -> k) :
    comp (whiskR a h) (whiskL g b) ->
    comp (whiskL f b) (whiskR a k)
```

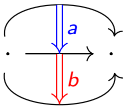
Outline

Inverse of the braiding

- The braiding has an obvious inverse

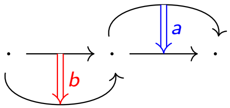
Inverse of the braiding

- The braiding has an obvious inverse
- Geometrically :



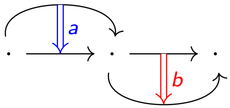
Inverse of the braiding

- The braiding has an obvious inverse
- Geometrically :



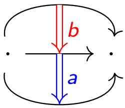
Inverse of the braiding

- The braiding has an obvious inverse
- Geometrically :



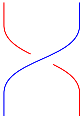
Inverse of the braiding

- The braiding has an obvious inverse
- Geometrically :



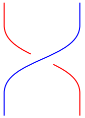
Inverse of the braiding

- The braiding has an obvious inverse
- Geometrically :



Inverse of the braiding

- The braiding has an obvious inverse
- Geometrically :



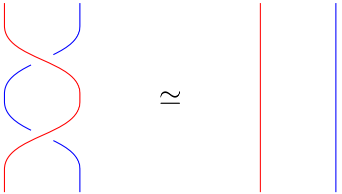
- Formally : The two definition have a very similar structure, but some of the arguments need to be inverted.

Cancellation witness

- The braiding and its inverse cancel out

Cancellation witness

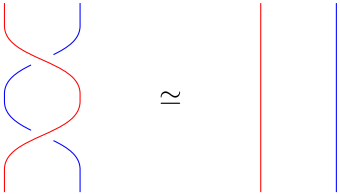
- The braiding and its inverse cancel out
- Geometrically :
  Reidemeister move of type II



Cancellation witness

- The braiding and its inverse cancel out

- Geometrically :
  Reidemeister move of type II



- Formally :
  Providing cancellation witnesses for all intermediate, we can assemble them to build a cancellation for the complete braiding.

Automation

- This example shows how inversion works

Automation

- This example shows how inversion works
- It would be possible to generate inverses and cancellation witnesses algorithmically (but it is more technical than it looks, and has not been implemented yet)

Outline

triply monoidal ω-categories

Definition

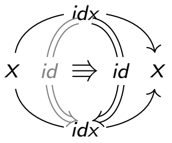
A triply monoidal ω-category is a weak ω-category with only one object, one 1-cell, and one 2-cell

triply monoidal ω-categories

Definition

A triply monoidal ω-category is a weak ω-category with only one object, one 1-cell, and one 2-cell

In CaTT:



triply monoidal ω-categories

Definition

A triply monoidal ω-category is a weak ω-category with only one object, one 1-cell, and one 2-cell

In CaTT:

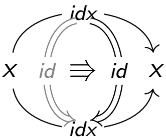
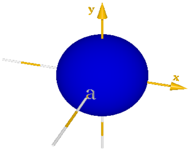


Diagram:



Geometric idea

- The "suspended" braiding

animations/eh1-3D/eh01.png