CaTT: A type theory for weak ω-categories

Thibaut Benjamin

CEA LIST, work conducted at Ecole Polytechnique

HoTTEST Event for Junior Researchers
January 13, 2022

Introduction

- I will present the type theory CaTT, introduced by Finster and Mimram.

Introduction

- I will present the type theory CaTT, introduced by Finster and Mimram.

- I have conducted my PhD thesis about this theory.

Introduction

- I will present the type theory CaTT, introduced by Finster and Mimram.

- I have conducted my PhD thesis about this theory.

- I am currently a postdoc at CEA LIST, where I work on runtime verification of C programs (in frama-c).

HoTT, Groupoids, and Brunerie's Type Theory

HoTT and ω-Groupoids

- The iterated identity types endow types with a structure of a weak ω-groupoid.

HoTT and ω-Groupoids

- The iterated identity types endow types with a structure of a weak ω-groupoid.
- But we can abstract out this structure and define it as a minimal type theory (Brunerie)

Brunerie's Type Theory

- HoTT, but stripped off of everything except identities

Brunerie's Type Theory

- HoTT, but stripped off of everything except identities

- So a context looks like :

  `(x:*, y:*, z:*, f:x=y, f':x=y, g:z=y, a:f=f')`

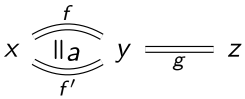
Brunerie's Type Theory

- HoTT, but stripped off of everything except identities

- So a context looks like :

  `(x:*, y:*, z:*, f:x=y, f':x=y, g:z=y, a:f=f')`

- These contexts describe *arbitrary equality situations* (a.k.a computads for weak ω-groupoids)



Compositions

- In HoTT, identity types have recursors that allows to combine them.

Compositions

- In HoTT, identity types have recursors that allows to combine them.

- In Brunerie's Type Theory, we define *compositions* and *axioms* for that.

Compositions

- In HoTT, identity types have recursors that allows to combine them.

- In Brunerie's Type Theory, we define *compositions* and *axioms* for that.

- This allows for instance to derive the terms

  `(x:*, y:*, z:*, f:x=y, g:y=z)` $\vdash$ `c(f,g) : x=z`

  $x \overset{f}{=\!=\!=} y \overset{g}{=\!=\!=} z$

Compositions

- In HoTT, identity types have recursors that allows to combine them.

- In Brunerie's Type Theory, we define *compositions* and *axioms* for that.

- This allows for instance to derive the terms

  `(x:*, y:*, z:*, f:x=y, g:y=z)` $\vdash$ `c(f,g) : x=z`

  $$x \overset{f}{=\!=\!=} y \overset{g}{=\!=\!=} z$$

  `(x:*, y:*, z:*, w:*, f:x=y, g:y=z, h:z=w)`
  $\vdash$ `a(f,g,h) : c(f,c(g,h))=c(c(f,g),h)`

  $$x \overset{f}{=\!=\!=} y \overset{g}{=\!=\!=} z \overset{h}{=\!=\!=} w$$

CaTT : A Type Theory for Weak ω-Categories

General Idea

- Same as Brunerie's Type Theory, but for weak ω-categories
  Replace equalities with rewriting relations

General Idea

- Same as Brunerie's Type Theory, but for weak ω-categories
  Replace equalities with rewriting relations

- Contexts are *arbitrary rewriting situations* (a.k.a computads for weak ω-categories)

General Idea

- Same as Brunerie's Type Theory, but for weak ω-categories
  Replace equalities with rewriting relations
- Contexts are *arbitrary rewriting situations* (a.k.a computads for weak ω-categories)
- Define compositions and axioms for those.

Arbitrary rewriting situations

- Like in Brunerie's Type Theory, but replace equalities with arrows

Arbitrary rewriting situations

- Like in Brunerie's Type Theory, but replace equalities with arrows

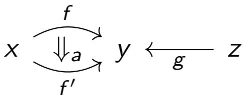
- Example :

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

Arbitrary rewriting situations

- Like in Brunerie's Type Theory, but replace equalities with arrows
- Example :

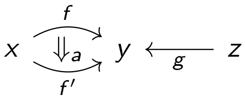
  `(x:*, y:*, z:*, f:x->y, f':x->y, g:z->y, a:f->f')`
- This context corresponds to the following diagram (globular set)



Arbitrary rewriting situations

- Like in Brunerie's Type Theory, but replace equalities with arrows
- Example :

  `(x:*, y:*, z:*, f:x->y, f':x->y, g:z->y, a:f->f')`
- This context corresponds to the following diagram (globular set)



- We will see more general contexts later

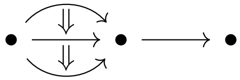
Pasting schemes

- Pasting schemes are the context that describe essentially a single unambiguous rewriting situation.

Pasting schemes

- Pasting schemes are the context that describe essentially a single unambiguous rewriting situation.

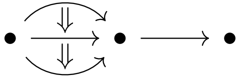
- Example :



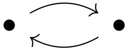
Pasting schemes

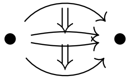
- Pasting schemes are the context that describe essentially a single unambiguous rewriting situation.

- Example :



- Counter-example :





Pasting schemes

- Pasting schemes are the context that describe essentially a single unambiguous rewriting situation.

- Example :

- Counter-example :

- We can recognize them algorithmically

Composition in CaTT

- Composition express the idea that the space of composite of a pasting schemes is contractible.

Composition in CaTT

- Composition express the idea that the space of composite of a pasting schemes is contractible.

- **First rule** (operations) :
  Every pasting scheme has a composition

Composition in CaTT

- Composition express the idea that the space of composite of a pasting schemes is contractible.

- **First rule** (operations) :
  Every pasting scheme has a composition

- **Second rule** (axioms) :
  Any two compositions of the same pasting scheme are related by a higher cell.

Composition in CaTT

- Composition express the idea that the space of composite of a pasting schemes is contractible.

- **First rule** (operations) :
  Every pasting scheme has a composition

- **Second rule** (axioms) :
  Any two compositions of the same pasting scheme are related by a higher cell.

- Let's see this live !