An unbiased definition of simply typed λ-calculus

Samuel Mimram

École polytechnique

CoREACT meeting
November 29, 2024

Part I

A type theory for weak higher categories

Higher categories

The definition of (strict) ω-**category** generalizes categories by taking higher cells into account.

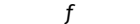
Higher categories

The definition of (strict) ω-**category** generalizes categories by taking higher cells into account.

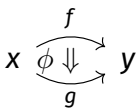
In such a category, you have

- 0-cells (objects):

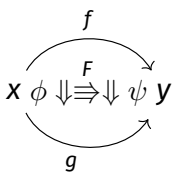
- 1-cells (morphisms):



- 2-cells:



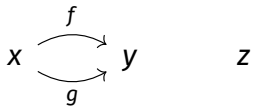
- 3-cells:



Higher categories

The definition of (strict) ω-**category** generalizes categories by taking higher cells into account.

In such a category, you have **compositions**



Weak higher categories

In a **weak** higher category, all the axioms hold up to a

A type-theoretic definition of weak ω-categories

We have introduced a definition of weak ω-categories as models of a type theory:

- *A type-theoretical definition of weak ω-categories*.
  Eric Finster and Samuel Mimram. In 2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), pages 1–12, 2017.
- *Globular weak ω-categories as models of a type theory*.
  Thibaut Benjamin, Eric Finster, and Samuel Mimram. Higher Structures, 8(2):1–69, 2024.

A type-theoretic definition of categories

Cartmell, 1984:

- type constructors:

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

A type-theoretic definition of categories

Cartmell, 1984:

- type constructors:

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

- term constructors:

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

$$\frac{}{x : \star, y : \star, f : x \to y, z : \star, g : y \to z \vdash \mathsf{comp}(f, g) : x \to z}$$

A type-theoretic definition of categories

Cartmell, 1984:

- type constructors:

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

- term constructors:

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

$$\frac{}{x : \star, y : \star, f : x \to y, z : \star, g : y \to z \vdash \mathsf{comp}(f, g) : x \to z}$$

- axioms:

$$\frac{\Gamma \vdash f : x \to y}{\Gamma \vdash \mathsf{comp}(\mathsf{id}(x), f) = f} \qquad \frac{\Gamma \vdash f : x \to y}{\Gamma \vdash \mathsf{comp}(f, \mathsf{id}(y)) = f} \qquad \dots$$

A type-theoretic definition of categories

Cartmell, 1984:

- type constructors:

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

- term constructors:

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

$$\frac{}{x : \star, y : \star, f : x \to y, z : \star, g : y \to z \vdash \mathsf{comp}(f, g) : x \to z}$$

- axioms:

$$\frac{\Gamma \vdash f : x \to y}{\Gamma \vdash \mathsf{comp}(\mathsf{id}(x), f) = f} \qquad \frac{\Gamma \vdash f : x \to y}{\Gamma \vdash \mathsf{comp}(f, \mathsf{id}(y)) = f} \qquad \dots$$

- plus “standard rules” (contexts, weakening, substitutions, …)

Unbiased definition

Since the composition is associative for categories, the composite of any diagram like

$$x_0 \xrightarrow{f_1} x_1 \xrightarrow{f_2} \dots \xrightarrow{f_n} x_n$$

is uniquely defined.

So, instead of having a binary composition and identities, we could have a more general rule

$$\frac{}{x_0 : \star, x_1 : \star, f_1 : x_0 \to x_1, \dots, x_n : \star, f_n : x_{n-1} \to x_n \vdash \mathsf{comp}(f_1, \dots, f_n) : x_0 \to x_n}$$

Globular sets

Proposition

Globular sets are precisely the models of the type theory

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

Globular sets

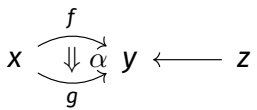
Proposition

Globular sets are precisely the models of the type theory

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

Remark

A finite globular set



can be encoded as a context

$$x : \star, y : \star, z : \star, f : x \underset{\star}{\to} y, g : x \underset{\star}{\to} y, h : z \underset{\star}{\to} y, \alpha : f \underset{x \underset{\star}{\to} y}{\to} g$$

Globular sets

Proposition

Globular sets are precisely the models of the type theory

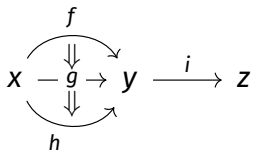
$$\frac{\Gamma \vdash}{\Gamma \vdash \star} \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : A}{\Gamma \vdash t \underset{A}{\to} u} \qquad \dots$$

Proposition

The syntactic category (of contexts and substitutions) of this type theory is the opposite of the category of finite globular sets.

Pasting schemes

A **pasting scheme** is a diagram for which we expect to have unique composition.



We write

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

to indicate that Γ encodes a pasting scheme.

Type-theoretic ω-categories

- operations:

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

  whenever

$$\mathsf{FV}(t) = \mathsf{FV}(\partial^-(\Gamma)) \qquad \text{and} \qquad \mathsf{FV}(u) = \mathsf{FV}(\partial^+(\Gamma))$$

- coherences:

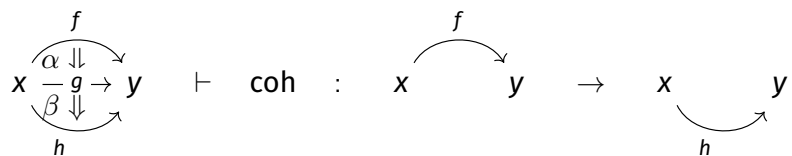
$$\frac{\Gamma \vdash_{\mathsf{ps}} \qquad \Gamma \vdash A}{\Gamma \vdash \mathsf{coh}_{\Gamma, A} : A}$$

  whenever

$$\mathsf{FV}(A) = \mathsf{FV}(\Gamma)$$

Type-theoretic ω-categories

A typical example of **operation** is *composition*



(this coherence is noted "**comp**" in the following).

Type-theoretic ω-categories

Note that we also have *unbiased compositions*:

$$x \xrightarrow{f} y \xrightarrow{g} z \xrightarrow{h} w \quad \vdash \quad \mathsf{coh} \quad : \quad x \quad \to \quad w$$

Type-theoretic ω-categories

A typical example of **coherence** is *associativity*

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

$$\vdash$$

$$\mathsf{coh} \quad : \quad x \xrightarrow{\mathrm{comp}(\mathrm{comp}(f,g),h)} w \quad \to \quad x \xrightarrow{\mathrm{comp}(f,\mathrm{comp}(g,h))} w$$

The general scheme

The general pattern is that

- we identify things that should be contractible (the pasting schemes)
- and formally make them contractible

Part II

Unbiased cartesian closed categories

Next steps

Previous work is nice but

- the type theory is very limited (no Σ- or Π-types, etc.)
- we would like to be able to consider categories with structure ([locally] cartesian [closed]...)

Here

1. we restrict to 1-categories
2. we extend with products and internal homs

If we restrict our theory for weak ω-categories to consider that 2-cells (and higher are identities), we obtain a theory for **unbiased categories**:

$$x \xrightarrow{f} y \xrightarrow{g} z \xrightarrow{h} w \quad \vdash \quad \mathsf{coh} \quad : \quad x \quad \to \quad w$$

Unbiased cartesian closed 1-categories

If we restrict our theory for weak ω-categories to consider that 2-cells (and higher are identities), we obtain a theory for **unbiased categories**:

$$x \xrightarrow{f} y \xrightarrow{g} z \xrightarrow{h} w \quad \vdash \quad \mathsf{coh} \quad : \quad x \quad \to \quad w$$

Our aim is to extend this to have a definition for **unbiased cartesian closed categories** (which could hopefully extend in higher dimensions).

Unbiased cartesian closed 1-categories

There are various definition of cartesian closed categories:

- the traditional categorical definition
- simply-typed λ-calculus
- combinatory logic (**I**, **K**, **S**)
- categorical combinators
- ...

We want here

- an agnostic approach in which we could implement most of the above
- a "nice" definition which does not require substitution/α-conversion, weird rules, etc.
- on the long term, we would like an "equality-free" definition of MLTT...

Simply-typed λ-calculus

We consider simply-typed λ-calculus where types are

$$A \quad ::= \quad X \quad | \quad A \to B \quad | \quad \dots$$

terms are

$$t \quad ::= \quad x \quad | \quad \lambda x^A.t \quad | \quad t\,u$$

rules are

$$\frac{}{\Gamma, x : A, \Delta \vdash x : A}$$

$$\frac{\Gamma \vdash t : A \to B \qquad \Gamma \vdash u : A}{\Gamma \vdash t\,u : B} \qquad\qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x^A.t : A \to B}$$

and equality is *extensional equality*

$$(\lambda x^A.t)\,u = t[u/x] \qquad\qquad t = \lambda x^A.t\,x$$

Note: we consider the implicational fragment for simplicity

Contractible types

We say that a type is

- **propositional** when there is at most one inhabitant (modulo extensional equality)
- **contractible** when there is exactly one inhabitant

Contractible types

We say that a type is

- **propositional** when there is at most one inhabitant (modulo extensional equality)
- **contractible** when there is exactly one inhabitant

For instance:

- $(A \to B) \to A \to B$ is contractible
- $B \to A$ is propositional
- $(A \to A) \to (A \to A)$ is not contractible

Komori's conjecture

We write $A \leq B$ when B can be obtained from A by replacing variables, for instance

$$A \to A \leq (A \to B) \to (A \to B)$$

Komori's conjecture

We write $A \leq B$ when B can be obtained from A by replacing variables, for instance

$$A \to A \leq (A \to B) \to (A \to B)$$

From Yuichi Komori. *BCK algebras and lambda calculus.* In Proceedings of the 10th Symposium on Semigroups, pages 5–11, 1987:

Conjecture

A minimal provable type is contractible.

For instance,

$$(A \to A) \to (A \to A)$$

is not contractible, but this is the case of the smaller

$$(A \to B) \to A \to B$$

Komori's conjecture

We write $A \leq B$ when B can be obtained from A by replacing variables, for instance

$$A \to A \leq (A \to B) \to (A \to B)$$

From Yuichi Komori. *BCK algebras and lambda calculus.* In Proceedings of the 10th Symposium on Semigroups, pages 5–11, 1987:

Conjecture

A minimal provable type is contractible.

For instance,

$$(A \to A) \to (A \to A)$$

is not contractible, but this is the case of the smaller

$$(A \to B) \to A \to B$$

This does not hold, but we will see that contractible types still generate.

Komori's conjecture

The conjecture does hold for formulas of depth $\leq \mathbf{2}$

Komori's conjecture

The conjecture does hold for formulas of depth ≤ 2
but various counter-examples to the conjecture were found:

- Mint'90: $((((A \to B) \to A) \to A) \to B) \to B$
- Aoto'99: $((A \to B) \to A) \to ((A \to B) \to B)$

```
fun x -> fun y -> y (x (fun z -> y z))
fun x -> fun y -> y (x (fun z -> y (x (fun z -> y z))))
fun x -> fun y -> y (x (fun z -> y (x (fun z -> y (x (fun z -> y z))))))
...
```

Propositional formulas

People tried to come up with conditions which would imply that a formula has at most one proof:

- Hirokawa'93:
  in implication fragments of BCI and BCK (no contraction), minimal inhabited types are contractible.
- Aoto'99:
  provable without *non-prime contraction*, i.e. an implication introduction rule whose canceled assumption differs from a propositional variable and appears more than once in the proof.

Propositional formulas

Another trend of work is in

- Mint'82, Babaev&Solov'ev'82 (coherence for CCC):
  a formula which is balanced (no variable occurs more than twice) admits at most one inhabitant

For instance,

- $(A \to B) \to (A \to B)$ is balanced and thus contractible
- $(A \to B) \to (B \to A)$ is balanced by not inhabited
- $(A \to B \to C) \to (A \to B) \to A \to C$ is not balanced but contractible (this is $\mathbf{S}$!)

Propositional formulas

This is generalized in

- Takahito Aoto and Hiroakira Ono. *Uniqueness of normal proofs in $\{\rightarrow, \wedge\}$-fragment of NJ*. Technical Report IS-RR-94-0024F, School of Information Science, JAIST, 1994. Research report.
- Pierre Bourreau and Sylvain Salvati. *Game semantics and uniqueness of type inhabitance in the simply-typed λ-calculus*. In 10th International Conference, TLCA 2011, Novi Sad, Serbia, June 1-3, 2011. Proceedings, volume 6690 of LNCS, pages 61–75. Springer, 2011.
- Sabine Broda and Luiís Damas. *On long normal inhabitants of a type*. J. Log. Comput., 15(3):353–390, 2005.

Propositional formulas

A type is **non-negatively duplicating** when every variable has at most one negative occurrence, e.g.

$$\mathsf{S} \quad : \quad (A^+ \to B^+ \to C^-) \to (A^+ \to B^-) \to A^- \to C^+$$

Propositional formulas

A type is **non-negatively duplicating** when every variable has at most one negative occurrence, e.g.

$$\mathsf{S} \quad : \quad (A^+ \to B^+ \to C^-) \to (A^+ \to B^-) \to A^- \to C^+$$

Theorem
A non-negatively duplicating type is propositional.

Propositional formulas

A type is **non-negatively duplicating** when every variable has at most one negative occurrence, e.g.

$$S \quad : \quad (A^+ \to B^+ \to C^-) \to (A^+ \to B^-) \to A^- \to C^+$$

Theorem
A non-negatively duplicating type is propositional.

Proof.
By cut-elimination, we can look for a proof which corresponds to a λ-term which is β-normal and η-long (we take as many variables as there are arrows), i.e. terms of the form

$$\lambda x_1 x_2 \dots x_n . x_i t_1 t_2 \dots t_k$$

the choice of the head variable must have a type whose target is *the* negative occurrence of the variable being proved. □

Propositional formulas

Proof search in $\beta\eta$-long form can be performed with

- the introduction

$$\frac{\Gamma, x_1 : A_1, \ldots, x_n : A_n \vdash t : B}{\Gamma \vdash \lambda x_1 \ldots x_n . t : A_1 \to \ldots \to A_n \to B} \ (\to_{\mathrm{I}})$$

  where B is not an arrow,

- the elimination

$$\frac{\Gamma \vdash t_1 : A_1 \qquad \ldots \qquad \Gamma \vdash t_n : A_n}{\Gamma \vdash x\, t_1 \ldots t_n : X} \ (\to_{\mathrm{E}})$$

  with $x : A_1 \to \ldots \to A_n \to X$ in Γ.

Propositional formulas

$$\vdash \qquad\qquad : (A^+ \to B^+ \to C^-) \to (A^+ \to B^-) \to A^- \to C^+$$

Propositional formulas

$$\frac{f : A^+ \to B^+ \to C^-, g : A^+ \to B^-, x : A^- \vdash \qquad\qquad : C^+}{\vdash \lambda f g x. \qquad\qquad : (A^+ \to B^+ \to C^-) \to (A^+ \to B^-) \to A^- \to C^+}$$

Propositional formulas

$$\dfrac{\dfrac{\Gamma \vdash \quad : A^+ \qquad \Gamma \vdash \quad : B^+}{f : A^+ \to B^+ \to C^-, g : A^+ \to B^-, x : A^- \vdash f \qquad\quad : C^+}}{\vdash \lambda fgx.f \qquad\quad : (A^+ \to B^+ \to C^-) \to (A^+ \to B^-) \to A^- \to C^+}$$

Propositional formulas

$$\dfrac{\dfrac{\overline{\Gamma \vdash x : A^+} \qquad \Gamma \vdash \quad : B^+}{f : A^+ \to B^+ \to C^-, g : A^+ \to B^-, x : A^- \vdash f\, x \qquad : C^+}}{\vdash \lambda f g x. f\, x \qquad : (A^+ \to B^+ \to C^-) \to (A^+ \to B^-) \to A^- \to C^+}$$

Propositional formulas

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash x : A^+} \qquad \dfrac{\Gamma \vdash \quad : A^+}{\Gamma \vdash g \quad : B^+}}{f : A^+ \to B^+ \to C^-, g : A^+ \to B^-, x : A^- \vdash f\, x\, (g \quad) : C^+}}{\vdash \lambda f g x. f\, x\, (g \quad) : (A^+ \to B^+ \to C^-) \to (A^+ \to B^-) \to A^- \to C^+}$$

Propositional formulas

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash x : A^+}}{f : A^+ \to B^+ \to C^-, g : A^+ \to B^-, x : A^- \vdash f\, x\, (g\, x) : C^+}}{\vdash \lambda f g x. f\, x\, (g\, x) : (A^+ \to B^+ \to C^-) \to (A^+ \to B^-) \to A^- \to C^+}}$$

Propositional formulas

This can be generalized to **deterministic** formulas.

This can be generalized to **deterministic** formulas.

Remark
Every non-negatively duplicated formula is deterministic but there are more:

$$(((A \to B) \to A \to C) \to (B \to C) \to D) \to (B \to C) \to D$$

This can be generalized to **deterministic** formulas.

Remark
Every non-negatively duplicated formula is deterministic but there are more:

$$(((A \to B) \to A \to C) \to (B \to C) \to D) \to (B \to C) \to D$$

Remark
There are non-deterministic formulas which are contractible:

- $(A \to B) \to B \to B$
- $(A \to A) \to (A \to B) \to B \to B$

Contractible formulas

A (apparently new) remark is that in the rule

$$\frac{\Gamma \vdash t_1 : A_1 \qquad \dots \qquad \Gamma \vdash t_n : A_n}{\Gamma \vdash x\, t_1 \dots t_n : X}\ (\to_{\mathrm{E}})$$

(with $x : A_1 \to \dots \to A_n \to X$ in Γ), we never use x in the t_i (otherwise the proof would be "infinite" by determinism). Because of this,

Proposition

Contractibility is (very easily) decidable.

Pasting types

Definition
A type A is a **pasting type** when it is non-negatively duplicated and inhabited.

Pasting types

Definition
A type A is a **pasting type** when it is non-negatively duplicated and inhabited.

Proposition
Any pasting type is contractible.

Pasting types

Definition
A type A is a **pasting type** when it is non-negatively duplicated and inhabited.

Proposition
Any pasting type is contractible.

Definition
We say that

$$A_1, \ldots, A_n \vdash A$$

is a **pasting scheme** when

$$A_1 \to \ldots \to A_n \to A$$

is a pasting type.

Pasting types

The rules for **pasting types** are

$$\frac{\Theta, \mathsf{TV}(A); \Gamma, A \vdash_{\mathsf{ps}} B}{\Theta; \Gamma \vdash_{\mathsf{ps}} A \to B}$$

when $\mathsf{TV}(A) \notin \Theta$, and

$$\frac{\Theta; \Gamma, \Gamma' \vdash A_1 \qquad \dots \qquad \Theta; \Gamma, \Gamma' \vdash A_n}{\Theta; \Gamma, A_1 \to \dots \to A_n \to A, \Gamma' \vdash A}$$

when A is a variable.

Pasting types

The rules for **pasting types** are

$$\frac{\Theta, \mathsf{TV}(A); \Gamma, A \vdash_{\mathsf{ps}} B}{\Theta; \Gamma \vdash_{\mathsf{ps}} A \to B}$$

when $\mathsf{TV}(A) \notin \Theta$, and

$$\frac{\Theta; \Gamma, \Gamma' \vdash A_1 \qquad \dots \qquad \Theta; \Gamma, \Gamma' \vdash A_n}{\Theta; \Gamma, A_1 \to \dots \to A_n \to A, \Gamma' \vdash A}$$

when A is a variable.

This is in between non-negatively duplicated and deterministic.

CCCaTT

The type theory **CCCaTT** has rules

$$\frac{}{\Gamma \vdash \star} \qquad \frac{\Gamma \vdash a : \star \qquad \Gamma \vdash b : \star}{\Gamma \vdash a \to b : \star} \qquad \frac{\Gamma \vdash a : \star}{\Gamma \vdash a}$$

$$\frac{\Gamma \vdash_{\mathsf{ps}} a}{\Gamma \vdash \mathsf{coh} : a} \qquad \frac{\Gamma \vdash_{\mathsf{ps}} a \qquad \Gamma \vdash t : a \qquad \Gamma \vdash u : a}{\Gamma \vdash t = u : a}$$

plus

- $=$ is a congruence
- closure under substitution so that

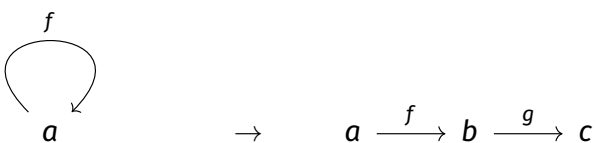
$$\frac{\Delta \vdash \sigma : \Gamma \qquad \Gamma \vdash t : a}{\Delta \vdash t[\sigma] : a[\sigma]}$$

  is derivable

Substitutions

Note that substitutions can replace both types and morphisms.

For instance, we have a substitution

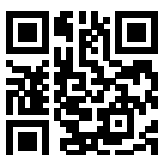


$$(a : \star, f : a \to a) \quad \vdash \quad \langle a, a, a, f, g \rangle : (a : \star, b : \star, c : \star, f : a \to b, g : b \to c)$$

CCCaTT

For instance, we can derive

- $\mathsf{I} = \lambda x.x : A \to A$ (but also meta-identity)
- $\mathsf{K} = \lambda xy.x : A \to B \to A$ (but also the variant of type $A \to A \to A$)
- $\mathsf{S} = \lambda fgx.fx(gx) : (A \to B \to C) \to (A \to B) \to A \to C$
- application $f : A \to B, x : A \vdash fx : B$
- expected equalities such as $\mathsf{I}\ x = x$



`https://cccatt.mimram.fr/`

Combinatory logic

Combinatory logic is defined as the closure under applications of I, K and S.

Combinatory logic

Combinatory logic is defined as the closure under applications of I, K and S.

With rules

$$\mathsf{I}\,t = t \qquad \mathsf{K}\,tu = t \qquad \mathsf{S}\,tuv = tv(uv)$$

Combinatory logic

Combinatory logic is defined as the closure under applications of I, K and S.

With rules

$$\mathsf{I}\,t = t \qquad \mathsf{K}\,tu = t \qquad \mathsf{S}\,tuv = tv(uv)$$

along with

- $\mathsf{S}(\mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}\,\mathsf{K})(\mathsf{S}(\mathsf{K}\,\mathsf{S})\,\mathsf{K})))(\mathsf{K}\,\mathsf{K}) = \mathsf{S}(\mathsf{K}\,\mathsf{K})$
- $\mathsf{S}(\mathsf{S}(\mathsf{K}\,\mathsf{S})\,\mathsf{K})(\mathsf{K}\,\mathsf{I}) = \mathsf{I}$
- $\mathsf{S}(\mathsf{K}\,\mathsf{I}) = \mathsf{I}$
- $\mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}\,\mathsf{K})) = \mathsf{K}$
- $\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}\,\mathsf{S})))(\mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}\,\mathsf{S}))) = \mathsf{S}(\mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}\,\mathsf{K})(\mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}\,\mathsf{S})))\,\mathsf{S}))))(\mathsf{K}\,\mathsf{S})$

Combinatory logic vs λ-calculus

We can translate from CC to λ:

$$\mathsf{I} = \lambda x.x \qquad \mathsf{K} = \lambda xy.x \qquad \mathsf{S} = \lambda fgx.fx(gx)$$

Combinatory logic vs λ-calculus

We can translate from CC to λ:

$$\mathsf{I} = \lambda x.x \qquad \mathsf{K} = \lambda xy.x \qquad \mathsf{S} = \lambda fgx.fx(gx)$$

On the other side, we have that $[\![\lambda x.t]\!] = \Lambda x.[\![t]\!]$ with

$$\begin{aligned} \Lambda x.x &= \mathsf{I} && \\ \Lambda x.t &= \mathsf{K}\ t && \text{for } x \notin \mathsf{FV}(t) \\ \Lambda x.t\,u &= \mathsf{S}(\Lambda x.t)(\Lambda x.u) && \text{otherwise} \end{aligned}$$

Combinatory logic vs λ-calculus

We can translate from CC to λ:

$$\mathsf{I} = \lambda x.x \qquad \mathsf{K} = \lambda xy.x \qquad \mathsf{S} = \lambda fgx.fx(gx)$$

On the other side, we have that $[\![\lambda x.t]\!] = \Lambda x.[\![t]\!]$ with

$$\begin{aligned} \Lambda x.x &= \mathsf{I} && \\ \Lambda x.t &= \mathsf{K}\ t && \text{for } x \notin \mathsf{FV}(t) \\ \Lambda x.t\,u &= \mathsf{S}(\Lambda x.t)(\Lambda x.u) && \text{otherwise} \end{aligned}$$

For instance,

$$[\![\lambda xy.x]\!] = \mathsf{S}\,(\mathsf{K}\ \mathsf{K})\,\mathsf{I}$$

Combinatory logic vs λ-calculus

We can translate from CC to λ:

$$\mathsf{I} = \lambda x.x \qquad \mathsf{K} = \lambda xy.x \qquad \mathsf{S} = \lambda fgx.fx(gx)$$

On the other side, we have that $[\![\lambda x.t]\!] = \Lambda x.[\![t]\!]$ with

$$\begin{aligned} \Lambda x.x &= \mathsf{I} && \\ \Lambda x.t &= \mathsf{K}\ t && \text{for } x \notin \mathsf{FV}(t) \\ \Lambda x.t\,u &= \mathsf{S}(\Lambda x.t)(\Lambda x.u) && \text{otherwise} \end{aligned}$$

For instance,

$$[\![\lambda xy.x]\!] = \mathsf{S}\,(\mathsf{K}\ \mathsf{K})\,\mathsf{I} \neq \mathsf{K}$$

Adding products

Type isomorphism in cartesian closed categories is the congruence generated by

$$(A \times B) \times C = A \times (B \times C) \qquad \mathbf{1} \times A = A$$
$$A \times B = B \times A \qquad A \times \mathbf{1} = A$$
$$A \to (B \times C) = (A \to B) \times (A \to C) \qquad A \to \mathbf{1} = \mathbf{1}$$
$$(A \times B) \to C = A \to B \to C \qquad \mathbf{1} \to A = A$$

Adding products

Type isomorphism in cartesian closed categories is the congruence generated by

$$(A \times B) \times C = A \times (B \times C) \qquad 1 \times A = A$$
$$A \times B = B \times A \qquad A \times 1 = A$$
$$A \to (B \times C) = (A \to B) \times (A \to C) \qquad A \to 1 = 1$$
$$(A \times B) \to C = A \to B \to C \qquad 1 \to A = A$$

We consider the rewriting system

$$A \to (B \times C) \Rightarrow (A \to B) \times (A \to C) \qquad A \to 1 \Rightarrow 1$$
$$(A \times B) \to C \Rightarrow A \to B \to C \qquad 1 \to A \Rightarrow A$$

Adding products

Type isomorphism in cartesian closed categories is the congruence generated by

$$(A \times B) \times C = A \times (B \times C) \qquad 1 \times A = A$$
$$A \times B = B \times A \qquad A \times 1 = A$$
$$A \to (B \times C) = (A \to B) \times (A \to C) \qquad A \to 1 = 1$$
$$(A \times B) \to C = A \to B \to C \qquad 1 \to A = A$$

We consider the rewriting system

$$A \to (B \times C) \Rightarrow (A \to B) \times (A \to C) \qquad A \to 1 \Rightarrow 1$$
$$(A \times B) \to C \Rightarrow A \to B \to C \qquad 1 \to A \Rightarrow A$$

A product $A_1 \times \ldots \times A_n$ is **pasting** when all the A_i are.

Adding products

For combinators we should add

$$\mathsf{P} : A \to B \to A \times B \qquad \mathsf{P}_1 : A \times B \to A \qquad \mathsf{P}_2 : A \times B \to A \qquad \mathsf{T} : \mathbf{1}$$

along with the obvious equations

Adding products

For combinators we should add

$$\mathsf{P} : A \to B \to A \times B \qquad \mathsf{P}_1 : A \times B \to A \qquad \mathsf{P}_2 : A \times B \to A \qquad \mathsf{T} : \mathbf{1}$$

along with the obvious equations

$$\begin{aligned} \mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}\,\mathsf{P}_1)))))(\mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}\,\mathsf{P}))) &= \mathsf{K} \\ \mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}\,\mathsf{P}_2)))))(\mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}\,\mathsf{P}))) &= \mathsf{K}\,\mathsf{I} \\ \mathsf{S}(\mathsf{S}(\mathsf{K}\,\mathsf{S})(\mathsf{S}(\mathsf{K}(\mathsf{S}(\mathsf{K}\,\mathsf{P})))(\mathsf{S}(\mathsf{K}\,\mathsf{P}_1))))(\mathsf{S}(\mathsf{K}\,\mathsf{P}_2)) &= \mathsf{I} \end{aligned}$$

The theorem

We have axiomatized cartesian closed categories.

Theorem
There is a bijection between

- *terms* $\vdash t : A$ *modulo equality (in contexts containing only type definitions),*
- λ*-terms of type* A *modulo* $\beta\eta$*-equality.*

Proof.
$\Rightarrow$ Pasting types are contractible so they correspond to (unique) λ-terms.
$\Leftarrow$ λ-terms can be implemented with combinators, which can be derived in CCCaTT. $\square$

Categorical combinators

AA_K:

(Ass)	$(x^{\sigma_3 \Rightarrow \sigma_4} \circ y^{\sigma_2 \Rightarrow \sigma_3}) \circ z^{\sigma_1 \Rightarrow \sigma_2} = x \circ (y \circ z)$
(IdL)	$\mathrm{Id}^{\tau \Rightarrow \tau} \circ x^{\sigma \Rightarrow \tau} = x^{\sigma \Rightarrow \tau}$
(IdR)	$x^{\sigma \Rightarrow \tau} \circ \mathrm{Id}^{\sigma \Rightarrow \sigma} = x$
(Fst)	$\mathrm{Fst}^{\tau_1, \tau_2} \circ \langle x^{\sigma \Rightarrow \tau_1},\ y^{\sigma \Rightarrow \tau_2} \rangle = x$
(Snd)	$\mathrm{Snd}^{\tau_1, \tau_2} \circ \langle x^{\sigma \Rightarrow \tau_1},\ y^{\sigma \Rightarrow \tau_2} \rangle = y$
(DPair)	$\langle x^{\sigma_1 \Rightarrow \tau_1},\ y^{\sigma_1 \Rightarrow \tau_2} \rangle \circ z^{\sigma \Rightarrow \sigma_1} = \langle x \circ z,\ y \circ z \rangle$
(Beta)	$\mathrm{App}^{\sigma_2, \sigma_3} \circ \langle \Lambda(x^{\sigma_1 \times \sigma_2 \Rightarrow \sigma_3}),\ y^{\sigma_1 \Rightarrow \sigma_2} \rangle = x \circ \langle \mathrm{Id}^{\sigma_1 \Rightarrow \sigma_1},\ y \rangle$
$(D\Lambda)$	$\Lambda(x^{\sigma_1 \times \sigma_2 \Rightarrow \sigma_3}) \circ y^{\sigma \Rightarrow \sigma_1} = \Lambda(x \circ \langle y \circ \mathrm{Fst}^{\sigma, \sigma_2},\ \mathrm{Snd}^{\sigma, \sigma_2} \rangle)$
(AI)	$\Lambda(\mathrm{App}^{\sigma, \tau}) = \mathrm{Id}^{(\sigma \Rightarrow \tau) \Rightarrow (\sigma \Rightarrow \tau)}$
(FSI)	$\langle \mathrm{Fst}^{\sigma, \tau},\ \mathrm{Snd}^{\sigma, \tau} \rangle = \mathrm{Id}^{\sigma \times \tau \Rightarrow \sigma \times \tau}$
(ass)	$(x^{\sigma_1 \Rightarrow \sigma_2} \circ y^{\sigma \Rightarrow \sigma_1})\ z^{\sigma} = x(yz)$
(fst)	$\mathrm{Fst}^{\sigma_1, \sigma_2}(x^{\sigma_1},\ y^{\sigma_2}) = x$
(snd)	$\mathrm{Snd}^{\sigma_1, \sigma_2}(x^{\sigma_1},\ y^{\sigma_2}) = y$
(dpair)	$\langle x^{\sigma \Rightarrow \tau_1},\ y^{\sigma \Rightarrow \tau_2} \rangle\ z^{\sigma} = (xz,\ yz)$
(app)	$\mathrm{App}^{\sigma, \tau}(x^{\sigma \Rightarrow \tau},\ y^{\sigma}) = xy$
(Quote_1)	$\Lambda(\mathrm{Fst}^{\sigma, \sigma_2})\ x^{\sigma} \circ y^{\sigma_1 \Rightarrow \sigma_2} = \Lambda(\mathrm{Fst}^{\sigma, \sigma_1})\ x$
(Quote_2)	$\mathrm{App}^{\sigma_2, \sigma_3} \circ \langle x^{\sigma \Rightarrow (\sigma_2 \Rightarrow \sigma_3)} \circ \Lambda(\mathrm{Fst}^{\sigma, \sigma_1})\ y^{\sigma},\ z^{\sigma_1 \Rightarrow \sigma_2} \rangle = xy \circ z.$

Abstraction vs meta-abstraction

Because of **ap** we have that if we have a coherence

$$\Gamma \vdash \mathsf{coh} : A \to B$$

then we have a coherence

$$\Gamma, x : A \vdash \mathsf{coh} : B$$

```
coh ap {a b : .} (f : a -> b) (x : a) : b
coh I {a : .} : a -> a
let id {a : .} (x : a) := ap I x
```

Abstraction vs meta-abstraction

Because of **ap** we have that if we have a coherence

$$\Gamma \vdash \mathsf{coh} : A \to B$$

then we have a coherence

$$\Gamma, x : A \vdash \mathsf{coh} : B$$

```
coh ap {a b : .} (f : a -> b) (x : a) : b
coh I {a : .} : a -> a
let id {a : .} (x : a) := ap I x
```

The converse is true, but at the “meta-level”, which corresponds to λ-abstraction!

Unbiasing equalities

Equalities are (for now) imposed to be congruences, e.g.

$$\frac{\Gamma \vdash t = u \qquad \Gamma \vdash u = v}{\Gamma \vdash t = v}$$

or

$$\frac{\Gamma \vdash t = u \qquad \Gamma \vdash t_1 = u_1 \qquad \dots \qquad \Gamma \vdash t_n = u_n}{\Gamma \vdash t\langle t_1, \dots, t_n\rangle = u\langle u_1, \dots, u_n\rangle}$$

e.g.

```
coh ap-cong
  {a b : .}
  {t t' : a → b} {u u' : a}
  (p : t = t') (q : u = u') : ap t u = ap t' u'
```

Unbiasing equalities

Equalities are (for now) imposed to be congruences, e.g.

$$\frac{\Gamma \vdash t = u \qquad \Gamma \vdash u = v}{\Gamma \vdash t = v}$$

or

$$\frac{\Gamma \vdash t = u \qquad \Gamma \vdash t_1 = u_1 \qquad \dots \qquad \Gamma \vdash t_n = u_n}{\Gamma \vdash t\langle t_1, \dots, t_n\rangle = u\langle u_1, \dots, u_n\rangle}$$

e.g.

```
coh ap-cong
  {a b : .}
  {t t' : a → b} {u u' : a}
  (p : t = t') (q : u = u') : ap t u = ap t' u'
```

which is biased…

Unbiasing equalities

Equalities are (for now) imposed to be congruences, e.g.

$$\frac{\Gamma \vdash t = u \qquad \Gamma \vdash u = v}{\Gamma \vdash t = v}$$

or

$$\frac{\Gamma \vdash t = u \qquad \Gamma \vdash t_1 = u_1 \qquad \dots \qquad \Gamma \vdash t_n = u_n}{\Gamma \vdash t\langle t_1, \dots, t_n \rangle = u\langle u_1, \dots, u_n \rangle}$$

e.g.

```
coh ap-cong
  {a b : .}
  {t t' : a → b} {u u' : a}
  (p : t = t') (q : u = u') : ap t u = ap t' u'
```

which is biased…We can also give unbiased rules!

Interesting subsystems

Interesting subsystems can be defined including

- monoidal categories
- symmetric monoidal categories
- cartesian categories

Part III

Conclusion

We have defined simply-typed λ-calculus without anything which looks like reduction / evaluation / substitution.

This is implemented at

`https://cccatt.mimram.fr/`

Questions?