

# Synthesising a type theory for cartesian closed bicategories

Marcelo Fiore<sup>†</sup> and Philip Saville<sup>◇</sup>

<sup>†</sup>University of Cambridge  
Department of Computer Science and Technology

<sup>◇</sup>University of Edinburgh  
School of Informatics

## **Aim of this work:**

A principled construction of a type theory  
for cartesian closed bicategories (= CCCs up-to-isomorphism)

## Aim of this work:

A principled construction of a type theory  
for cartesian closed bicategories (= CCCs up-to-isomorphism)

Originally motivated by **coherence** but interesting in its right!

## Aim of this work:

A **principled** construction of a type theory  
for cartesian closed bicategories (= CCCs up-to-isomorphism)

## Aim of this work:

A **principled** construction of an **internal language**  
for cartesian closed bicategories (= CCCs up-to-isomorphism)

## Aim of this work:

A **principled** construction of an **internal language**  
for cartesian closed bicategories (= CCCs up-to-isomorphism)

## What is an internal language?

## Aim of this work:

A **principled** construction of an **internal language**  
for cartesian closed bicategories (= CCCs up-to-isomorphism)

**What is an internal language?** generally an *informal* term

## Aim of this work:

A **principled** construction of an **internal language**  
for cartesian closed bicategories (= CCCs up-to-isomorphism)

**What is an internal language?** generally an *informal* term

A **syntax** to describe some **semantic** structure

STLC  $\longleftrightarrow$  CCCs

string diagrams  $\longleftrightarrow$  PROPs

MLTT  $\longleftrightarrow$  LCCCs



## Aim of this work:

A **principled** construction of an **internal language**  
for cartesian closed bicategories (= CCCs up-to-isomorphism)

**What is an internal language?** generally an *informal* term

A **syntax** to describe some **semantic** structure

Typically, a **tool for reasoning** (e.g. less coherence data)

## Aim of this work:

A **principled** construction of an **internal language**  
for cartesian closed bicategories (= CCCs up-to-isomorphism)

## What is an internal language? generally an *informal* term

A **syntax** to describe some **semantic** structure

Typically, a **tool for reasoning** (e.g. less coherence data)

Language should be **sound** and **complete**

## Aim of this work:

A **principled** construction of an **internal language**  
for cartesian closed bicategories (= CCCs up-to-isomorphism)

## What is an internal language? generally an *informal* term

A **syntax** to describe some **semantic** structure

Typically, a **tool for reasoning** (e.g. less coherence data)

Language should be **sound** and **complete**

## What is **principled**?

No arbitrary choices

Based on analysis of algebraic structure

Parallel situation for cartesian closed categories

## Aim of this work:

A **principled** construction of an **internal language**  
for cartesian closed bicategories (= CCCs up-to-isomorphism)

## What is an internal language? generally an *informal* term

A **syntax** to describe some **semantic** structure

Typically, a **tool for reasoning** (e.g. less coherence data)

Language should be **sound** and **complete**

## What is **principled**?

No arbitrary choices

Based on analysis of algebraic structure

Parallel situation for cartesian closed categories

## Many **benefits!** makes life *much* easier

New information about cc-bicats, simpler proofs, new relationships, ...

**Strategy:** use tools from universal algebra

**Strategy:** use tools from universal algebra

1. cc-bicategories = CCCs up-to-isomorphism

**Strategy:** use tools from universal algebra

1. cc-bicategories = CCCs up-to-isomorphism
2. STLC = internal language of **cc-clones** (*c.f.* Lambek)

**Strategy:** use tools from universal algebra

1. cc-bicategories = CCCs up-to-isomorphism
2. STLC = internal language of [cc-clones](#) (c.f. Lambek)
3. bicategorify: get [cc-biclones](#)




**Strategy:** use tools from universal algebra

1. cc-bicategories = CCCs up-to-isomorphism
2. STLC = internal language of [cc-clones](#) (c.f. Lambek)
3. bicategorify: get [cc-biclones](#)
4.  $\Lambda_{\text{ps}}^{\times, \rightarrow}$  = internal language of cc-biclones

**Strategy:** use tools from universal algebra

1. cc-bicategories = CCCs up-to-isomorphism
2. STLC = internal language of **cc-clones** (c.f. Lambek)
3. bicategorify: get **cc-biclones**
4.  $\Lambda_{\text{ps}}^{\times, \rightarrow}$  = internal language of cc-biclones

**Today:** **principles** underlying the construction of  $\Lambda_{\text{ps}}^{\times, \rightarrow}$   
 focus on STLC

## cc-Bicategories

## Bicategories

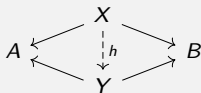
Categories with axioms 'up to isomorphism'.

e.g. profunctors,  $\text{Span}(\mathbb{C})$ , bicategories of relations,  $\text{Cat}$ , ...

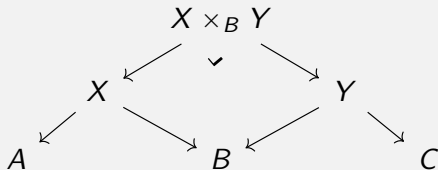
## Composition by universal property $\Rightarrow$ bicategory

In a category  $\mathbb{C}$  with pullbacks:

1. objects: objects of  $\mathbb{C}$ ,
2. 1-cells  $A \rightsquigarrow B$ : spans  $(A \leftarrow X \rightarrow B)$ ,
3. 2-cells: commutative squares



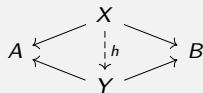
Composition defined by pullback:



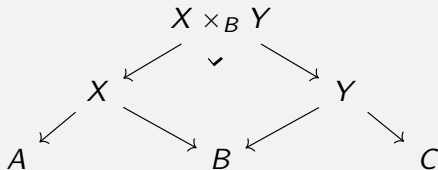
## Composition by universal property $\Rightarrow$ bicategory

In a category  $\mathbb{C}$  with pullbacks:

1. objects: objects of  $\mathbb{C}$ ,
2. 1-cells  $A \rightsquigarrow B$ : spans  $(A \leftarrow X \rightarrow B)$ ,
3. 2-cells: commutative squares



Composition defined by pullback:  $\rightsquigarrow$  associative up to iso



# Bicategories

# Bicategories

- Objects  $X \in ob(\mathcal{B})$ ,



# Bicategories

- Objects  $X \in ob(\mathcal{B})$ ,
- *Hom-categories*  $(\mathcal{B}(X, Y), \bullet, id)$ :

# Bicategories

- Objects  $X \in ob(\mathcal{B})$ ,
- Hom-categories  $(\mathcal{B}(X, Y), \bullet, id)$ :

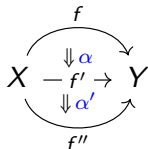
$$\begin{array}{l} \text{1-cells } X \xrightarrow{f} Y \\ \text{2-cells } X \begin{array}{c} \xrightarrow{f} \\ \Downarrow \alpha \\ \xrightarrow{f'} \end{array} Y \end{array}$$

# Bicategories

- Objects  $X \in \text{ob}(\mathcal{B})$ ,
- Hom-categories  $(\mathcal{B}(X, Y), \bullet, \text{id})$ :

1-cells  $X \xrightarrow{f} Y$

2-cells  $X \begin{array}{c} \xrightarrow{f} \\ \Downarrow \alpha \\ \xrightarrow{f'} \end{array} Y$



# Bicategories

- Objects  $X \in \text{ob}(\mathcal{B})$ ,
- Hom-categories  $(\mathcal{B}(X, Y), \bullet, \text{id})$ :

$$\begin{array}{l} \text{1-cells } X \xrightarrow{f} Y \\ \text{2-cells } X \begin{array}{c} \xrightarrow{f} \\ \Downarrow \alpha \\ \xrightarrow{f'} \end{array} Y \end{array}$$

- Identities  $\text{Id}_X : X \rightarrow X$  and composition

$$\mathcal{B}(Y, Z) \times \mathcal{B}(X, Y) \xrightarrow{\circ_{X, Y, Z}} \mathcal{B}(X, Z)$$

# Bicategories

- Objects  $X \in \text{ob}(\mathcal{B})$ ,
- Hom-categories  $(\mathcal{B}(X, Y), \bullet, \text{id})$ :

1-cells  $X \xrightarrow{f} Y$

2-cells  $X \begin{array}{c} \xrightarrow{f} \\ \Downarrow \alpha \\ \xrightarrow{f'} \end{array} Y$

- Identities  $\text{Id}_X : X \rightarrow X$  and composition

$$\mathcal{B}(Y, Z) \times \mathcal{B}(X, Y) \xrightarrow{\circ_{X, Y, Z}} \mathcal{B}(X, Z)$$

$$X \begin{array}{c} \xrightarrow{f} \\ \Downarrow \alpha \\ \xrightarrow{f'} \end{array} Y \begin{array}{c} \xrightarrow{g} \\ \Downarrow \beta \\ \xrightarrow{g'} \end{array} Z$$

# Bicategories

- Objects  $X \in \text{ob}(\mathcal{B})$ ,
- Hom-categories  $(\mathcal{B}(X, Y), \bullet, \text{id})$ :

$$\begin{array}{l} \text{1-cells } X \xrightarrow{f} Y \\ \text{2-cells } X \begin{array}{c} \xrightarrow{f} \\ \Downarrow \alpha \\ \xrightarrow{f'} \end{array} Y \end{array}$$

- Identities  $\text{Id}_X : X \rightarrow X$  and composition

$$\mathcal{B}(Y, Z) \times \mathcal{B}(X, Y) \xrightarrow{\circ_{X, Y, Z}} \mathcal{B}(X, Z)$$

- Invertible 2-cells

$$(h \circ g) \circ f \xrightarrow{a_{h, g, f}} h \circ (g \circ f)$$

$$\text{Id}_X \circ f \xrightarrow{l_f} f$$

$$g \circ \text{Id}_X \xrightarrow{r_g} g$$

subject to a triangle law and pentagon law.

# Bicategories

- Objects  $X \in \text{ob}(\mathcal{B})$ ,
- Hom-categories  $(\mathcal{B}(X, Y), \bullet, \text{id})$ :

$$\begin{array}{l} \text{1-cells } X \xrightarrow{f} Y \\ \text{2-cells } X \begin{array}{c} \xrightarrow{f} \\ \Downarrow \alpha \\ \xrightarrow{f'} \end{array} Y \end{array}$$

- Identities  $\text{Id}_X : X \rightarrow X$  and composition

$$\mathcal{B}(Y, Z) \times \mathcal{B}(X, Y) \xrightarrow{\circ_{X, Y, Z}} \mathcal{B}(X, Z)$$

- Invertible 2-cells

$$(h \circ g) \circ f \xrightarrow{a_{h, g, f}} h \circ (g \circ f)$$

$$\text{Id}_X \circ f \xrightarrow{l_f} f$$

$$g \circ \text{Id}_X \xrightarrow{r_g} g$$

subject to a triangle law and pentagon law.

~~~~> get a 2-category if  $a, l, r$  all id

## Bicategories

Categories with axioms 'up to isomorphism'.

e.g. profunctors,  $\text{Span}(\mathbb{C})$ , bicategories of relations,  $\text{Cat}$ , ...



## Bicategories

Categories with axioms 'up to isomorphism'.

e.g. profunctors,  $\text{Span}(\mathbb{C})$ , bicategories of relations,  $\text{Cat}$ , ...

## Cartesian closed bicategories

Cartesian closed categories 'up to isomorphism'.

Examples:

- Generalised species and cartesian distributors  
bicategorical models of LL, higher category theory  
(Fiore, Gambino, Hyland, Winskel), (Fiore & Joyal)
- Categorical algebra (operads)  
(Gambino & Joyal)
- Game semantics (concurrent games)  
(Yamada & Abramsky, Winskel *et al.*, Paquet)

## Cartesian closed bicategories = cc-bicategories

Bicategories  $\mathcal{B}$  equipped with families of **equivalences**

$$\mathcal{B}(X, \prod_{i=1}^n A_i) \simeq \prod_{i=1}^n \mathcal{B}(X, A_i)$$

$$\mathcal{B}(X, A \Rightarrow B) \simeq \mathcal{B}(X \times A, B)$$

Not related to Carboni & Walters' "cartesian bicategories"!

# Cartesian closed bicategories = cc-bicategories

Bicategories  $\mathcal{B}$  equipped with families of **equivalences**

$$\begin{array}{ccc} & (\pi_1 \circ -, \dots, \pi_n \circ -) & \\ & \curvearrowright & \\ \mathcal{B}(X, \prod_{i=1}^n A_i) & \perp \simeq & \prod_{i=1}^n \mathcal{B}(X, A_i) \\ & \curvearrowleft & \\ & \langle -, \dots, = \rangle & \\ & \text{(tupling)} & \end{array}$$

$$\begin{array}{ccc} & \text{eval}_{A,B} \circ (- \times A) & \\ & \curvearrowright & \\ \mathcal{B}(X, A \Rightarrow B) & \perp \simeq & \mathcal{B}(X \times A, B) \\ & \curvearrowleft & \\ & \lambda & \\ & \text{(currying)} & \end{array}$$

# Cartesian closed bicategories = cc-bicategories

Bicategories  $\mathcal{B}$  equipped with families of **equivalences**

$$\begin{array}{ccc} & (\pi_1 \circ -, \dots, \pi_n \circ -) & \\ & \curvearrowright & \\ \mathcal{B}(X, \prod_{i=1}^n A_i) & \perp \simeq & \prod_{i=1}^n \mathcal{B}(X, A_i) \\ & \curvearrowleft & \\ & \langle -, \dots, = \rangle & \\ & \text{(tupling)} & \end{array}$$

$$\begin{array}{ccc} & \text{eval}_{A,B} \circ (- \times A) & \\ & \curvearrowright & \\ \mathcal{B}(X, A \Rightarrow B) & \perp \simeq & \mathcal{B}(X \times A, B) \\ & \curvearrowleft & \\ & \lambda & \\ & \text{(currying)} & \end{array}$$

Triangle laws:

$$\begin{array}{ccc} & \pi_i \circ \langle \pi_1 \circ t, \dots, \pi_n \circ t \rangle & \\ \pi_i \circ t & \xrightarrow{\pi_i \circ \text{unit}} & \xrightarrow{\text{counit}} \pi_i \circ t \\ & \xlongequal{\quad\quad\quad} & \end{array}$$

## **Aim of this work:**

A principled construction of an internal language  
for cartesian closed bicategories (= CCCs up-to-isomorphism)

## **Aim of this work:**

A principled construction of an internal language  
for cartesian closed bicategories (= CCCs up-to-isomorphism)

## **Our starting point:**

A principled construction of an internal language  
for cartesian closed **categories**

## **Aim of this work:**

A principled construction of an internal language  
for cartesian closed bicategories (= CCCs up-to-isomorphism)

## **Our starting point:** folklore(?), *c.f.* Lambek, Jacobs, . . .

A principled construction of an internal language  
for cartesian closed **categories**

An internal language for CCCs



# Curry–Howard–Lambek correspondence



# Curry–Howard–Lambek correspondence



For every **graph** [= choice of base types and constants] get  $\Lambda^{x,\rightarrow}(G)$ ,

# Curry–Howard–Lambek correspondence



For every **graph** [= choice of base types and constants] get  $\Lambda^{x,\rightarrow}(G)$ ,

... and a CCC  $\mathbb{F}^{x,\rightarrow}[G]$ :

**objects:**  $\Lambda^{x,\rightarrow}(G)$ -types,  
**maps**  $A \rightarrow B$ : terms  $(x : A \vdash t : B) \text{ mod } \alpha\beta\eta$   
**composition:** substitution

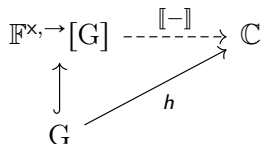
# Curry–Howard–Lambek correspondence



For every **graph** [= choice of base types and constants] get  $\Lambda^{x, \rightarrow}(G)$ ,  
... and a CCC  $\mathbb{F}^{x, \rightarrow}[G]$ :

objects:  $\Lambda^{x, \rightarrow}(G)$ -types,  
maps  $A \rightarrow B$ : terms  $(x : A \vdash t : B)$  mod  $\alpha\beta\eta$   
composition: substitution

with a **free property**:



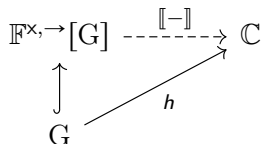
# Curry–Howard–Lambek correspondence



For every **graph** [= choice of base types and constants] get  $\Lambda^{x, \rightarrow}(G)$ ,  
... and a CCC  $\mathbb{F}^{x, \rightarrow}[G]$ :

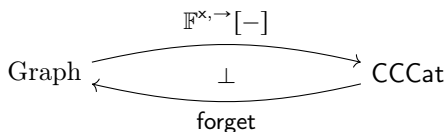
**objects:**  $\Lambda^{x, \rightarrow}(G)$ -types,  
**maps**  $A \rightarrow B$ : terms  $(x : A \vdash t : B)$  mod  $\alpha\beta\eta$   
**composition:** substitution

with a **free property**:



$\rightsquigarrow t =_{\beta\eta} t' \iff \llbracket t \rrbracket = \llbracket t' \rrbracket$  for every  $\mathbb{C}$

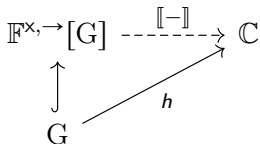
# Curry–Howard–Lambek correspondence



For every **graph** [= choice of base types and constants] get  $\Lambda^{x, \rightarrow}(G)$ ,  
... and a CCC  $\mathbb{F}^{x, \rightarrow}[G]$ :

objects:  $\Lambda^{x, \rightarrow}(G)$ -types,  
maps  $A \rightarrow B$ : terms  $(x : A \vdash t : B)$  mod  $\alpha\beta\eta$   
composition: substitution

with a **free property**:



$\rightsquigarrow t =_{\beta\eta} t' \iff \llbracket t \rrbracket = \llbracket t' \rrbracket$  for every  $\mathbb{C}$

# How good a fit is it, really?

Categories:

1-input, 1-output



Typed terms

$(x_1 : A_1, \dots, x_n : A_n \vdash t : B)$

have  $n$  inputs

# How good a fit is it, really?

Categories:

1-input, 1-output



Typed terms

$(x_1 : A_1, \dots, x_n : A_n \vdash t : B)$

have  $n$  inputs

Forced to **restrict** to unary contexts



## How good a fit is it, really?

Categories:

1-input, 1-output



Typed terms

$(x_1 : A_1, \dots, x_n : A_n \vdash t : B)$

have  $n$  inputs

Forced to **restrict** to unary contexts

Relying on the fact

$$\llbracket x_1 : A_1, \dots, x_n : A_n \rrbracket \cong \llbracket p : \prod_n (A_1, \dots, A_n) \rrbracket$$

## How good a fit is it, really?

Categories:

1-input, 1-output



Typed terms

$(x_1 : A_1, \dots, x_n : A_n \vdash t : B)$

have  $n$  inputs

Forced to **restrict** to unary contexts

Relying on the fact

$$\begin{aligned} \llbracket x_1 : A_1, \dots, x_n : A_n \rrbracket &\cong \llbracket p : \prod_n (A_1, \dots, A_n) \rrbracket \\ &\cong \llbracket \prod_n (A_1, \dots, A_n) \rrbracket \end{aligned}$$

# How good a fit is it, really?

Categories:

1-input, 1-output



Typed terms


$(x_1 : A_1, \dots, x_n : A_n \vdash t : B)$

have  $n$  inputs

Forced to **restrict** to unary contexts

Relying on the fact

$$\begin{aligned} \llbracket x_1 : A_1, \dots, x_n : A_n \rrbracket &\cong \llbracket p : \prod_n (A_1, \dots, A_n) \rrbracket \\ &\cong \llbracket \prod_n (A_1, \dots, A_n) \rrbracket \end{aligned}$$

 are these the same? Why?

# How good a fit is it, really?

Categories:

1-input, 1-output



Typed terms


$(x_1 : A_1, \dots, x_n : A_n \vdash t : B)$

have  $n$  inputs

Forced to **restrict** to unary contexts

Relying on the fact

$$\begin{aligned} \llbracket x_1 : A_1, \dots, x_n : A_n \rrbracket &\cong \llbracket p : \prod_n (A_1, \dots, A_n) \rrbracket \\ &\cong \llbracket \prod_n (A_1, \dots, A_n) \rrbracket \end{aligned}$$

 are these the same? Why?

Can have type-theoretic exponentials without products

# How good a fit is it, really?

Categories:

1-input, 1-output



Typed terms


$(x_1 : A_1, \dots, x_n : A_n \vdash t : B)$

have  $n$  inputs

Forced to **restrict** to unary contexts

Relying on the fact

$$\begin{aligned} \llbracket x_1 : A_1, \dots, x_n : A_n \rrbracket &\cong \llbracket p : \prod_n (A_1, \dots, A_n) \rrbracket \\ &\cong \llbracket \prod_n (A_1, \dots, A_n) \rrbracket \end{aligned}$$

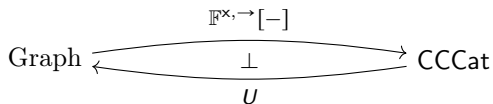
 are these the same? Why?

Can have type-theoretic exponentials without products

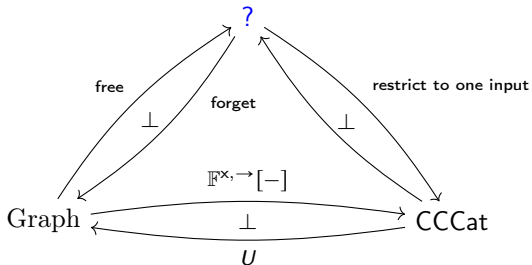
What if we take **many-input** behaviour seriously?

What if we take **many-input** behaviour seriously?

What if we take **many-input** behaviour seriously?

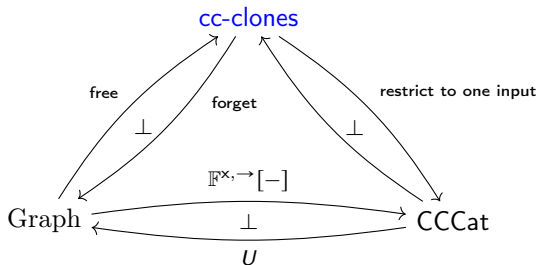


What if we take **many-input** behaviour seriously?

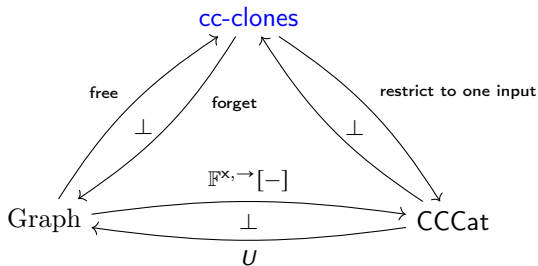




What if we take **many-input** behaviour seriously?

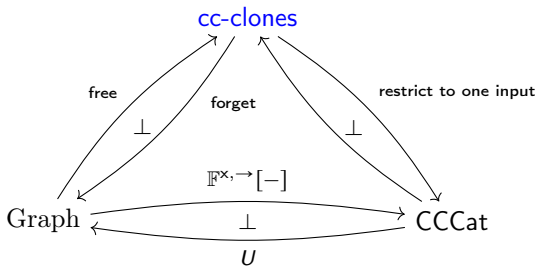


What if we take **many-input** behaviour seriously?



free CCC on  $G$   
 $\cong$   
 free cc-clone on  $G$   
 and restricting to one input

What if we take **many-input** behaviour seriously?



giving a syntax for CCCs

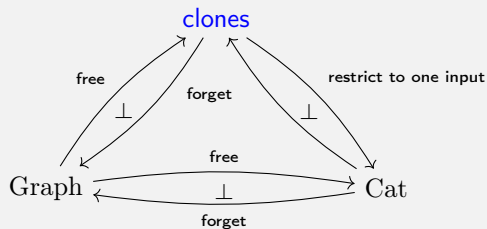
||

giving a syntax for cc-clones  
and restricting to unary contexts

# Defining an internal language for categories

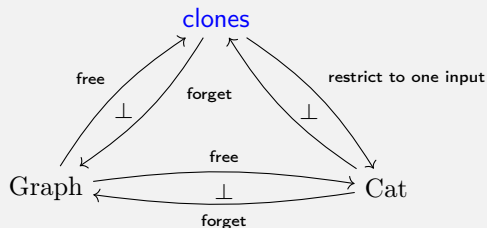
# Defining an internal language for categories

1. Observe clones [Hall, '81] factor the adjunction:



# Defining an internal language for categories

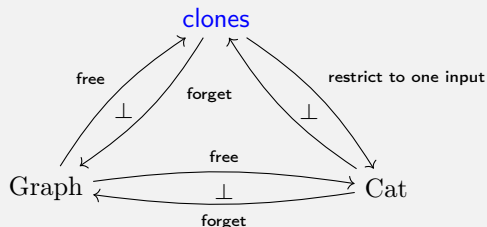
1. Observe clones [Hall, '81] factor the adjunction:



2. Define internal language of a clone,

# Defining an internal language for categories

1. Observe clones [Hall, '81] factor the adjunction:



2. Define internal language of a clone,
3. Internal language of categories  
def. internal language of free clone,  
restricted to unary contexts

Abstract clone  $(S, \mathbb{C}) =$  abstract theory of substitution:



Abstract clone  $(S, \mathbb{C}) =$  abstract theory of substitution:

- Sorts  $S$ ,

Abstract clone  $(S, \mathbb{C}) =$  abstract theory of substitution:

- Sorts  $S$ ,
- Hom-sets  $\mathbb{C}(X_1, \dots, X_n; Y)$  of *operations*  $X_1, \dots, X_n \xrightarrow{t} Y$ ,

Abstract clone  $(S, \mathbb{C}) =$  abstract theory of substitution:

- Sorts  $S$ ,
- Hom-sets  $\mathbb{C}(X_1, \dots, X_n; Y)$  of operations  $X_1, \dots, X_n \xrightarrow{t} Y$ ,
- Projections  $X_1, \dots, X_n \xrightarrow{p_{X_1, \dots, X_n}^{(i)}} X_i$  ( $1 \leq i \leq n$ ),

Abstract clone  $(S, \mathbb{C}) =$  abstract theory of substitution:

- Sorts  $S$ ,
- Hom-sets  $\mathbb{C}(X_1, \dots, X_n; Y)$  of operations  $X_1, \dots, X_n \xrightarrow{t} Y$ ,
- Projections  $X_1, \dots, X_n \xrightarrow{p_{X_1, \dots, X_n}^{(i)}} X_i$  ( $1 \leq i \leq n$ ),
- *Substitution* mappings

$$\begin{aligned} \mathbb{C}(X_1, \dots, X_n; Y) \times \prod_{i=1}^n \mathbb{C}(\Gamma; X_i) &\rightarrow \mathbb{C}(\Gamma; Y) \\ t, (u_1, \dots, u_n) &\mapsto t[u_1, \dots, u_n] \end{aligned}$$

Abstract clone  $(S, \mathbb{C}) =$  abstract theory of substitution:

- Sorts  $S$ ,
- Hom-sets  $\mathbb{C}(X_1, \dots, X_n; Y)$  of operations  $X_1, \dots, X_n \xrightarrow{t} Y$ ,
- Projections  $X_1, \dots, X_n \xrightarrow{p_{X_1, \dots, X_n}^{(i)}} X_i$  ( $1 \leq i \leq n$ ),
- *Substitution* mappings

$$\begin{aligned} \mathbb{C}(X_1, \dots, X_n; Y) \times \prod_{i=1}^n \mathbb{C}(\Gamma; X_i) &\rightarrow \mathbb{C}(\Gamma; Y) \\ t, (u_1, \dots, u_n) &\mapsto t[u_1, \dots, u_n] \end{aligned}$$

such that

$$\begin{aligned} p^{(k)}[u_1, \dots, u_n] &= u_k & (1 \leq k \leq n) \\ t[p^{(1)}, \dots, p^{(n)}] &= t \\ t[u_\bullet][v_\bullet] &= t[u_\bullet[v_\bullet]] \end{aligned}$$

Abstract clone  $(S, \mathbb{C}) =$  abstract theory of substitution:

- Sorts  $S$ ,
- Hom-sets  $\mathbb{C}(X_1, \dots, X_n; Y)$  of operations  $X_1, \dots, X_n \xrightarrow{t} Y$ ,
- Projections  $X_1, \dots, X_n \xrightarrow{p_{X_1, \dots, X_n}^{(i)}} X_i$  ( $1 \leq i \leq n$ ),
- *Substitution* mappings

$$\begin{aligned} \mathbb{C}(X_1, \dots, X_n; Y) \times \prod_{i=1}^n \mathbb{C}(\Gamma; X_i) &\rightarrow \mathbb{C}(\Gamma; Y) \\ t, (u_1, \dots, u_n) &\mapsto t[u_1, \dots, u_n] \end{aligned}$$

such that

$$\begin{aligned} p^{(k)}[u_1, \dots, u_n] &= u_k & (1 \leq k \leq n) \\ t[p^{(1)}, \dots, p^{(n)}] &= t \\ t[u_\bullet][v_\bullet] &= t[u_\bullet \cdot v_\bullet] \end{aligned}$$

Every clone  $(S, \mathbb{C})$  defines a **category**  $\bar{\mathbb{C}}$  and a **multicategory**  $\text{MC}$

Every clone  $(S, \mathbb{C})$  has an **internal language** [c.f. Lambek]:

$$(x_1 : A_1, \dots, x_n : A_n \vdash t : B) \iff t : A_1, \dots, A_n \rightarrow B$$

Every clone  $(S, \mathbb{C})$  has an **internal language** [c.f. Lambek]:

$$(x_1 : A_1, \dots, x_n : A_n \vdash t : B) \iff t : A_1, \dots, A_n \rightarrow B$$

$$(x_1 : A_1, \dots, x_n : A_n \vdash x_j : A_j) \rightsquigarrow p^{(j)} : A_1, \dots, A_n \rightarrow A_j$$

$$t[x_i \mapsto u_i] \rightsquigarrow t[u_1, \dots, u_n]$$



Every clone  $(S, \mathbb{C})$  has an **internal language** [c.f. Lambek]:

$$(x_1 : A_1, \dots, x_n : A_n \vdash t : B) \iff t : A_1, \dots, A_n \rightarrow B$$

$$(x_1 : A_1, \dots, x_n : A_n \vdash x_i : A_i) \rightsquigarrow p^{(i)} : A_1, \dots, A_n \rightarrow A_i$$

$$t[x_i \mapsto u_i] \rightsquigarrow t[u_1, \dots, u_n]$$

The clone axioms become:

$$x_k[x_i \mapsto u_i] = u_k \quad (1 \leq k \leq n)$$

$$t[x_i \mapsto x_i] = t$$

$$t[x_i \mapsto u_i][y_j \mapsto v_j] = t[x_i \mapsto u_i][y_j \mapsto v_j]$$

Every clone  $(S, \mathbb{C})$  has an **internal language** [c.f. Lambek]:

$$(x_1 : A_1, \dots, x_n : A_n \vdash t : B) \iff t : A_1, \dots, A_n \rightarrow B$$

$$(x_1 : A_1, \dots, x_n : A_n \vdash x_i : A_i) \rightsquigarrow p^{(i)} : A_1, \dots, A_n \rightarrow A_i$$

$$t[x_i \mapsto u_i] \rightsquigarrow t[u_1, \dots, u_n]$$

Free clone on a graph  $G$ :

$$\frac{c \in G(A, B)}{c : A \rightarrow B} \quad \frac{}{p^{(i)} : A_1, \dots, A_n \rightarrow A_i}$$

$$\frac{t : A_1, \dots, A_n \rightarrow B \quad (u_i : \Delta \rightarrow A_i)_{i=1, \dots, n}}{t[u_1, \dots, u_n] : \Delta \rightarrow B}$$

Every clone  $(S, \mathbb{C})$  has an **internal language** [c.f. Lambek]:

$$(x_1 : A_1, \dots, x_n : A_n \vdash t : B) \iff t : A_1, \dots, A_n \rightarrow B$$

$$(x_1 : A_1, \dots, x_n : A_n \vdash x_i : A_i) \rightsquigarrow p^{(i)} : A_1, \dots, A_n \rightarrow A_i$$

$$t[x_i \mapsto u_i] \rightsquigarrow t[u_1, \dots, u_n]$$

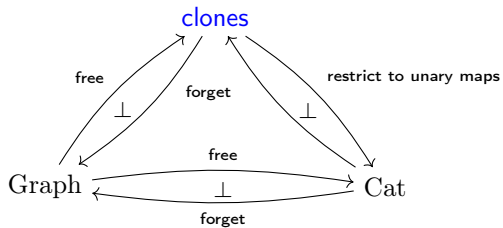
Free clone on a graph  $G$ :

$$\frac{c \in G(A, B)}{x : A \vdash c : B} \text{const} \quad \frac{}{x_1 : A_1, \dots, x_n : A_n \vdash x_i : A_i} \text{var}$$

$$\frac{x_1 : A_1, \dots, x_n : A_n \vdash t : B \quad (\Delta \vdash u_i : A_i)_{i=1, \dots, n}}{\Delta \vdash t[x_i \mapsto u_i] : B} \text{subst}$$

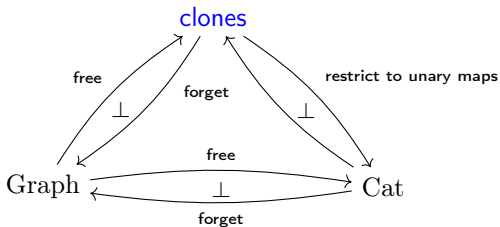
free category on  $G$

$\cong$  free clone on  $G$ , restricted to unary maps



free category on  $G$

$\cong$  free clone on  $G$ , restricted to unary maps



giving a syntax for categories

||

giving a syntax for clones  
and restricting to unary contexts

giving a syntax for categories

||

giving a syntax for clones  
and restricting to unary contexts

'internal language of categories' [informal]

||

internal language [formal] of the free clone

## Defining an internal language for cartesian categories

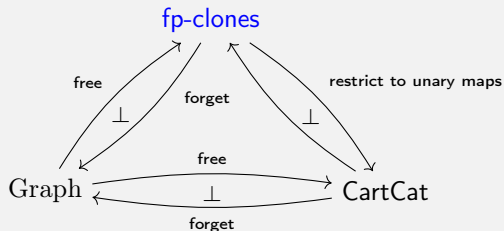


## Defining an internal language for cartesian categories

1. Define fp-clones,

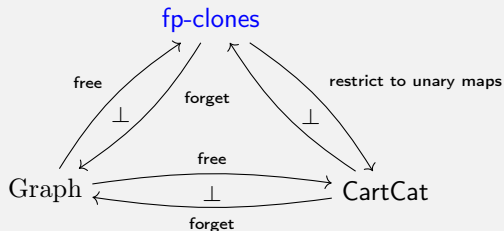
# Defining an internal language for cartesian categories

1. Define fp-clones,
2. Observe that:



# Defining an internal language for cartesian categories

1. Define fp-clones,
2. Observe that:



3. Internal language of cartesian categories  
def. internal language of free fp-clone,  
restricted to unary contexts

An **fp-clone** [fp = finite-product] is a clone  $(S, \mathbb{C})$  with

An **fp-clone** [fp = finite-product] is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,

An **fp-clone** [fp = finite-product] is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$  inducing isomorphisms

$$\mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) \cong \prod_{i=1}^n \mathbb{C}(\Gamma; A_i)$$

$\xrightarrow{(\pi_1[-], \dots, \pi_n[-])}$   
 $\xleftarrow{\langle -, \dots, = \rangle}$

An **fp-clone** [fp = finite-product] is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$  inducing isomorphisms

$$\mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) \cong \prod_{i=1}^n \mathbb{C}(\Gamma; A_i)$$

$\xrightarrow{(\pi_1[-], \dots, \pi_n[-])}$   
 $\xleftarrow{\langle -, \dots, = \rangle}$

Internal language of free fp-clone:

An **fp-clone** [fp = finite-product] is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$  inducing isomorphisms

$$\mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) \cong \prod_{i=1}^n \mathbb{C}(\Gamma; A_i)$$

$\xrightarrow{(\pi_1[-], \dots, \pi_n[-])}$   
 $\xleftarrow{\langle -, \dots, = \rangle}$

Internal language of free fp-clone:

$$\frac{A_1, \dots, A_n \text{ type}}{\prod_n(A_1, \dots, A_n) \text{ type}}$$



An **fp-clone** [fp = finite-product] is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$  inducing isomorphisms

$$\mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) \cong \prod_{i=1}^n \mathbb{C}(\Gamma; A_i)$$

$\xrightarrow{(\pi_1[-], \dots, \pi_n[-])}$   
 $\xleftarrow{\langle -, \dots, = \rangle}$

Internal language of free fp-clone:

$$\frac{}{p : \prod_n(A_1, \dots, A_n) \vdash \pi_i(p) : A_i} \quad (1 \leq i \leq n)$$

An **fp-clone** [fp = finite-product] is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$  inducing isomorphisms

$$\mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) \cong \prod_{i=1}^n \mathbb{C}(\Gamma; A_i)$$

$\xrightarrow{(\pi_1[-], \dots, \pi_n[-])}$   
 $\xleftarrow{\langle -, \dots, = \rangle}$

Internal language of free fp-clone:

$$\frac{}{p : \prod_n(A_1, \dots, A_n) \vdash \pi_i(p) : A_i} \quad (1 \leq i \leq n)$$

$$\frac{(\Gamma \vdash u_i : A_i)_{i=1, \dots, n}}{\Gamma \vdash \langle u_1, \dots, u_n \rangle : \prod_n(A_1, \dots, A_n)}$$

An **fp-clone** [fp = finite-product] is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$  inducing isomorphisms

$$\mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) \cong \prod_{i=1}^n \mathbb{C}(\Gamma; A_i)$$

$\xrightarrow{(\pi_1[-], \dots, \pi_n[-])}$   
 $\xleftarrow{\langle -, \dots, = \rangle}$

Internal language of free fp-clone:

$$\frac{}{p : \prod_n(A_1, \dots, A_n) \vdash \pi_i(p) : A_i} \quad (1 \leq i \leq n)$$

$$\frac{(\Gamma \vdash u_i : A_i)_{i=1, \dots, n}}{\Gamma \vdash \langle u_1, \dots, u_n \rangle : \prod_n(A_1, \dots, A_n)}$$

$$\pi_i(p)[p \mapsto \langle u_1, \dots, u_n \rangle] = u_i \quad (\beta)$$

$$\langle \pi_1(p)[p \mapsto u], \dots, \pi_n(p)[p \mapsto u] \rangle = u \quad (\eta)$$

An **fp-clone** [fp = finite-product] is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$  inducing isomorphisms

$$\mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) \cong \prod_{i=1}^n \mathbb{C}(\Gamma; A_i)$$

$\xrightarrow{(\pi_1[-], \dots, \pi_n[-])}$   
 $\xleftarrow{\langle -, \dots, \rangle}$

Internal language of free fp-clone:

$$\frac{\Gamma \vdash t : \prod_n(A_1, \dots, A_n)}{\Gamma \vdash \pi_i(t) : A_i} \quad (1 \leq i \leq n)$$

$$\frac{(\Gamma \vdash u_i : A_i)_{i=1, \dots, n}}{\Gamma \vdash \langle u_1, \dots, u_n \rangle : \prod_n(A_1, \dots, A_n)}$$

$$\pi_i(\langle u_1, \dots, u_n \rangle) = u_i \quad (\beta)$$

$$\langle \pi_1(t), \dots, \pi_n(t) \rangle = t \quad (\eta)$$

An **fp-clone** [fp = finite-product] is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$  inducing isomorphisms

$$\mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) \cong \prod_{i=1}^n \mathbb{C}(\Gamma; A_i)$$

$(\pi_1[-], \dots, \pi_n[-])$   
 $\langle -, \dots, = \rangle$

Internal language of free fp-clone  $\rightsquigarrow$  products in STLC

An **fp-clone** [fp = finite-product] is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$  inducing isomorphisms

$$\mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) \cong \prod_{i=1}^n \mathbb{C}(\Gamma; A_i)$$

$(\pi_1[-], \dots, \pi_n[-])$   
 $\langle -, \dots, = \rangle$

Internal language of free fp-clone  $\rightsquigarrow$  products in STLC

free cartesian category on  $G$

$\cong$  free fp-clone on  $G$ , restricted to unary contexts

An **fp-clone** [fp = finite-product] is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$  inducing isomorphisms

$$\mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) \cong \prod_{i=1}^n \mathbb{C}(\Gamma; A_i)$$

$(\pi_1[-], \dots, \pi_n[-])$   
 $\langle -, \dots, = \rangle$

Internal language of free fp-clone  $\rightsquigarrow$  products in STLC

free cartesian category on  $G$

$\cong$  free fp-clone on  $G$ , restricted to unary contexts

---

'internal language of fp-categories' [informal]

$\parallel$

internal language [formal] of the free fp-clone

An **fp-clone** [fp = finite-product] is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$  inducing isomorphisms

$$\mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) \cong \prod_{i=1}^n \mathbb{C}(\Gamma; A_i)$$

$\xrightarrow{(\pi_1[-], \dots, \pi_n[-])}$   
 $\xleftarrow{\langle -, \dots, = \rangle}$

~> Equivalent to requiring MC is representable



An **fp-clone** [fp = finite-product] is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$  inducing isomorphisms

$$\mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) \cong \prod_{i=1}^n \mathbb{C}(\Gamma; A_i)$$

$(\pi_1[-], \dots, \pi_n[-])$   
 $\langle -, \dots, = \rangle$

In an fp-clone, **contexts and products coincide**.

An **fp-clone** [fp = finite-product] is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$  inducing isomorphisms

$$\mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) \cong \prod_{i=1}^n \mathbb{C}(\Gamma; A_i)$$

$\xrightarrow{(\pi_1[-], \dots, \pi_n[-])}$   
 $\xleftarrow{\langle -, \dots, = \rangle}$

In an fp-clone, **contexts and products coincide**.

There exist maps

$$(u_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i)_{i=1, \dots, n}$$

$$t : A_1, \dots, A_n \rightarrow \prod_n(A_1, \dots, A_n)$$

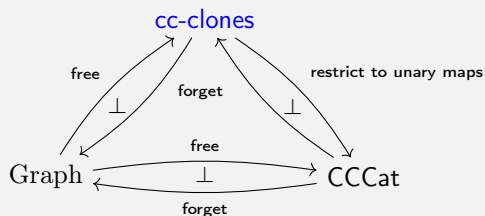
such that

$$t[u_1, \dots, u_n] = p_{\prod_n(A_1, \dots, A_n)}^{(1)}$$

$$u_i[t] = p_{A_1, \dots, A_n}^{(i)} \quad (i=1, \dots, n)$$

## Defining an internal language for cc-structure

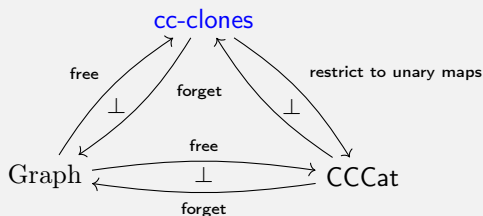
## Defining an internal language for cc-structure



---

free cartesian closed category on  $G$   
 $\cong$  free cc-clone on  $G$ , restricted to unary maps

## Defining an internal language for cc-structure



---

free cartesian closed category on  $G$   
 $\cong$  free cc-clone on  $G$ , restricted to unary maps

1. Define cc-clones,
2. Internal language of CCCs  
def. internal language of free cc-clone,  
restricted to unary contexts

A clone  $(S, \mathbb{C})$  has **exponentials** if  $\text{MC}$  has exponentials.

A clone  $(S, \mathbb{C})$  has **exponentials** if  $\text{MC}$  has exponentials.

---

A clone  $(S, \mathbb{C})$  **with exponentials** has

1. for every  $A, B$ , an object  $A \Rightarrow B$ ,

A clone  $(S, \mathbb{C})$  has **exponentials** if MC has exponentials.

A clone  $(S, \mathbb{C})$  **with exponentials** has

1. for every  $A, B$ , an object  $A \Rightarrow B$ ,
2. maps  $\text{eval}_{A,B} : A \Rightarrow B, A \rightarrow B$  inducing isomorphisms

$$\begin{array}{ccc} & \text{eval}[(-)[p^{(1)}, \dots, p^{(n)}, p^{(n+1)}] & \\ & \curvearrowright & \\ \mathbb{C}(X_1, \dots, X_n; A \Rightarrow B) & \cong & \mathbb{C}(X_1, \dots, X_n, A; B) \\ & \curvearrowleft & \\ & \lambda & \end{array}$$



A clone  $(S, \mathbb{C})$  has **exponentials** if MC has exponentials.

A clone  $(S, \mathbb{C})$  **with exponentials** has

1. for every  $A, B$ , an object  $A \Rightarrow B$ ,
2. maps  $\text{eval}_{A,B} : A \Rightarrow B, A \rightarrow B$  inducing isomorphisms

$$\begin{array}{ccc} & \text{eval}[(-)[p^{(1)}, \dots, p^{(n)}, p^{(n+1)}] & \\ & \curvearrowright & \\ \mathbb{C}(X_1, \dots, X_n; A \Rightarrow B) & \cong & \mathbb{C}(X_1, \dots, X_n, A; B) \\ & \curvearrowleft & \\ & \lambda & \end{array}$$

for  $n = 1$ :

if  $t : X, A \rightarrow B$  then  $\text{eval}[t[p^{(1)}], p^{(2)}] : X \rightarrow (A \Rightarrow B)$

A clone  $(S, \mathbb{C})$  has **exponentials** if MC has exponentials.

A clone  $(S, \mathbb{C})$  **with exponentials** has

1. for every  $A, B$ , an object  $A \Rightarrow B$ ,
2. maps  $\text{eval}_{A,B} : A \Rightarrow B, A \rightarrow B$  inducing isomorphisms

$$\begin{array}{ccc} & \text{eval}[(-)[p^{(1)}, \dots, p^{(n)}, p^{(n+1)}] & \\ & \curvearrowright & \\ \mathbb{C}(X_1, \dots, X_n; A \Rightarrow B) & \cong & \mathbb{C}(X_1, \dots, X_n, A; B) \\ & \curvearrowleft & \\ & \lambda & \end{array}$$

for  $n = 1$ :

if  $t : X, A \rightarrow B$  then  $\text{eval}[t[p^{(1)}], p^{(2)}] : X \rightarrow (A \Rightarrow B)$

$\rightsquigarrow$  c.f.  $\text{eval} \circ \langle f \circ \pi_1, \pi_2 \rangle = \text{eval} \circ (f \times A)$

An **fp-clone** is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$  inducing isomorphisms

$$\mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) \cong \prod_{i=1}^n \mathbb{C}(\Gamma; A_i)$$

$\xrightarrow{(\pi_1[-], \dots, \pi_n[-])}$   
 $\xleftarrow{\langle -, \dots, = \rangle}$

A clone  $(S, \mathbb{C})$  **with exponentials** has

1. for every  $A, B$ , an object  $A \Rightarrow B$ ,
2. maps  $\text{eval}_{A,B} : A \Rightarrow B, A \rightarrow B$  inducing isomorphisms

$$\mathbb{C}(X_1, \dots, X_n; A \Rightarrow B) \cong \mathbb{C}(X_1, \dots, X_n, A; B)$$

$\xrightarrow{\text{eval}[(-)][p^{(1)}, \dots, p^{(n)}, p^{(n+1)}]}$   
 $\xleftarrow{\lambda}$

An **fp-clone** is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$  inducing isomorphisms

$$\mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) \cong \prod_{i=1}^n \mathbb{C}(\Gamma; A_i)$$

$\xrightarrow{(\pi_1[-], \dots, \pi_n[-])}$   
 $\xleftarrow{\langle -, \dots, = \rangle}$

A clone  $(S, \mathbb{C})$  **with exponentials** has

1. for every  $A, B$ , an object  $A \Rightarrow B$ ,
2. maps  $\text{eval}_{A,B} : A \Rightarrow B, A \rightarrow B$  inducing isomorphisms

$$\mathbb{C}(X_1, \dots, X_n; A \Rightarrow B) \cong \mathbb{C}(X_1, \dots, X_n, A; B)$$

$\xrightarrow{\text{eval}[(-)][p^{(1)}, \dots, p^{(n)}, p^{(n+1)}]}$   
 $\xleftarrow{\lambda}$

A clone is **cartesian closed** [cc-] if it is an fp-clone with exponentials.

internal language of free cc-clone  
||  
simply-typed lambda calculus

internal language of free cc-clone

||

simply-typed lambda calculus

---

'internal language of cartesian closed categories' [informal]

||

internal language [formal] of the free cc-clone

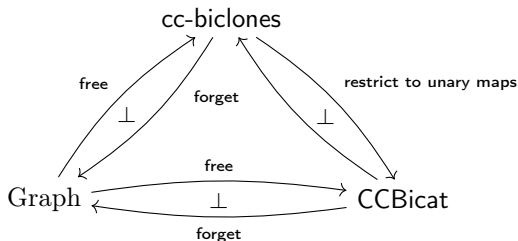
||

simply-typed lambda calculus

$\Lambda_{\text{ps}}^{\times, \rightarrow}$ : an internal language for cc-bicategories

# A recipe for $\Lambda_{ps}^{\times, \rightarrow}$ , modulo technicalities

1. Define cc-biclones,
2. Define internal language of a biclone,

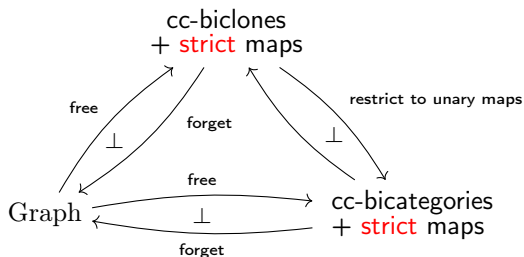


3. Internal language of cc-bicategories  
def. internal language of free cc-biclone,  
restricted to unary contexts



# A recipe for $\Lambda_{ps}^{\times, \rightarrow}$ , modulo technicalities

1. Define cc-biclones,
2. Define internal language of a biclone,



3. Internal language of cc-bicategories  
def. internal language of free cc-biclone,  
restricted to unary contexts

## A recipe for $\Lambda_{\text{ps}}^{\times, \rightarrow}$ , modulo technicalities

1. Define cc-biclones,
2. Define **internal language** of a biclone,
3. Internal language of cc-bicategories  
 $\stackrel{\text{def.}}{=}$  internal language of free cc-biclone,  
restricted to unary contexts

# A recipe for an internal language for bicategories

1. Define **biclones**,
2. Define notion of **internal language**,
3. Internal language of bicategories  
**def.** internal language of free biclone,  
restricted to unary contexts

Biclone  $(S, \mathcal{C})$  = abstract theory of bicategorical substitution:

Biclone  $(S, \mathcal{C})$  = abstract theory of bicategorical substitution:

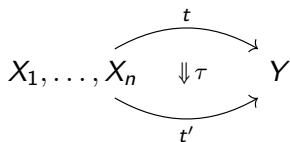
- *Sorts*  $S$ ,

Biclone  $(S, \mathcal{C})$  = abstract theory of bicategorical substitution:

- *Sorts*  $S$ ,
- Hom-categories  $(\mathcal{C}(X_1, \dots, X_n; Y), \bullet, \text{id})$ ,

Biclone  $(S, \mathcal{C}) =$  abstract theory of bicategorical substitution:

- Sorts  $S$ ,
- Hom-categories  $(\mathcal{C}(X_1, \dots, X_n; Y), \bullet, \text{id})$ ,



Biclone  $(S, \mathcal{C}) =$  abstract theory of bicategorical substitution:

- *Sorts*  $S$ ,
- Hom-categories  $(\mathcal{C}(X_1, \dots, X_n; Y), \bullet, \text{id})$ ,
- *Projection* 1-cells  $p_{X_\bullet}^{(i)} : X_1, \dots, X_n \rightarrow X_i$  ( $1 \leq i \leq n$ ),



Biclone  $(S, \mathcal{C}) =$  abstract theory of bicategorical substitution:

- *Sorts*  $S$ ,
- Hom-categories  $(\mathcal{C}(X_1, \dots, X_n; Y), \bullet, \text{id})$ ,
- *Projection* 1-cells  $p_{X_\bullet}^{(i)} : X_1, \dots, X_n \rightarrow X_i$  ( $1 \leq i \leq n$ ),
- *Substitution functors*

$$\begin{aligned} \mathcal{C}(X_1, \dots, X_n; Y) \times \prod_{i=1}^n \mathcal{C}(\Gamma; X_i) &\rightarrow \mathcal{C}(\Gamma; Y) \\ t, (u_1, \dots, u_n) &\mapsto t[u_1, \dots, u_n] \\ \tau, (\sigma_1, \dots, \sigma_n) &\mapsto \tau[\sigma_1, \dots, \sigma_n] \end{aligned}$$

Biclone  $(S, \mathcal{C}) =$  abstract theory of bicategorical substitution:

- Sorts  $S$ ,
- Hom-categories  $(\mathcal{C}(X_1, \dots, X_n; Y), \bullet, \text{id})$ ,
- Projection 1-cells  $p_{X_\bullet}^{(i)} : X_1, \dots, X_n \rightarrow X_i$  ( $1 \leq i \leq n$ ),
- Substitution functors

$$\begin{aligned} \mathcal{C}(X_1, \dots, X_n; Y) \times \prod_{i=1}^n \mathcal{C}(\Gamma; X_i) &\rightarrow \mathcal{C}(\Gamma; Y) \\ t, (u_1, \dots, u_n) &\mapsto t[u_1, \dots, u_n] \\ \tau, (\sigma_1, \dots, \sigma_n) &\mapsto \tau[\sigma_1, \dots, \sigma_n] \end{aligned}$$

- Structural isomorphisms

$$\begin{aligned} p^{(k)}[u_1, \dots, u_n] &\xrightarrow{\rho_{u_\bullet}^{(k)}} u_k \quad (1 \leq k \leq n) \\ t[p^{(1)}, \dots, p^{(n)}] &\xrightarrow{\iota_t} t \\ t[u_\bullet][v_\bullet] &\xrightarrow{\text{assoc}_{t; u_\bullet; v_\bullet}} t[u_\bullet[v_\bullet]] \end{aligned}$$

Biclone  $(S, \mathcal{C}) =$  abstract theory of bicategorical substitution:

- Sorts  $S$ ,
- Hom-categories  $(\mathcal{C}(X_1, \dots, X_n; Y), \bullet, \text{id})$ ,
- Projection 1-cells  $p_{X_\bullet}^{(i)} : X_1, \dots, X_n \rightarrow X_i$  ( $1 \leq i \leq n$ ),
- Substitution functors

$$\begin{aligned} \mathcal{C}(X_1, \dots, X_n; Y) \times \prod_{i=1}^n \mathcal{C}(\Gamma; X_i) &\rightarrow \mathcal{C}(\Gamma; Y) \\ t, (u_1, \dots, u_n) &\mapsto t[u_1, \dots, u_n] \\ \tau, (\sigma_1, \dots, \sigma_n) &\mapsto \tau[\sigma_1, \dots, \sigma_n] \end{aligned}$$

- Structural isomorphisms

$$\begin{aligned} p^{(k)}[u_1, \dots, u_n] &\xrightarrow{\rho_{u_\bullet}^{(k)}} u_k \quad (1 \leq k \leq n) \\ t[p^{(1)}, \dots, p^{(n)}] &\xrightarrow{\iota_t} t \\ t[u_\bullet][v_\bullet] &\xrightarrow{\text{assoc}_{t; u_\bullet; v_\bullet}} t[u_\bullet[v_\bullet]] \end{aligned}$$

subject to a triangle law and pentagon law.

Biclone  $(S, \mathcal{C})$  = abstract theory of bicategorical substitution:

- Sorts  $S$ ,

Every biclone defines a **bicategory** and a **bi-multicategory**

- Hom-categories  $(\mathcal{C}(X_1, \dots, X_n; Y), \bullet, \text{id})$ ,

- Projection 1-cells  $p_{X_\bullet}^{(i)} : X_1, \dots, X_n \rightarrow X_i$  ( $1 \leq i \leq n$ ),

- Substitution functors

$$\begin{aligned} \mathcal{C}(X_1, \dots, X_n; Y) \times \prod_{i=1}^n \mathcal{C}(\Gamma; X_i) &\rightarrow \mathcal{C}(\Gamma; Y) \\ t, (u_1, \dots, u_n) &\mapsto t[u_1, \dots, u_n] \\ \tau, (\sigma_1, \dots, \sigma_n) &\mapsto \tau[\sigma_1, \dots, \sigma_n] \end{aligned}$$

- Structural isomorphisms

$$\begin{aligned} p^{(k)}[u_1, \dots, u_n] &\xrightarrow{\rho_{u_\bullet}^{(k)}} u_k \quad (1 \leq k \leq n) \\ t[p^{(1)}, \dots, p^{(n)}] &\xrightarrow{\iota_t} t \\ t[u_\bullet][v_\bullet] &\xrightarrow{\text{assoc}_{t; u_\bullet; v_\bullet}} t[u_\bullet[v_\bullet]] \end{aligned}$$

subject to a triangle law and pentagon law.

# Internal language for biclones [c.f. Hilken, Seely, Hirschowitz]

# Internal language for biclones [c.f. Hilken, Seely, Hirschowitz]

terms:

$$(x_1 : A_1, \dots, x_n : A_n \vdash t : B)$$



$$t : A_1, \dots, A_n \rightarrow B$$

# Internal language for biclones [c.f. Hilken, Seely, Hirschowitz]

terms:

$$\begin{array}{c} (x_1 : A_1, \dots, x_n : A_n \vdash t : B) \\ \Downarrow \\ t : A_1, \dots, A_n \rightarrow B \end{array}$$

rewrites:

$$\begin{array}{c} (x_1 : A_1, \dots, x_n : A_n \vdash \tau : t \Rightarrow t' : B) \\ \Downarrow \\ \tau : t \Rightarrow t' : A_1, \dots, A_n \rightarrow B \end{array}$$

# Internal language for biclones [c.f. Hilken, Seely, Hirschowitz]

terms:

$$\begin{array}{c} (x_1 : A_1, \dots, x_n : A_n \vdash t : B) \\ \Downarrow \\ t : A_1, \dots, A_n \rightarrow B \end{array}$$

rewrites:

$$\begin{array}{c} (x_1 : A_1, \dots, x_n : A_n \vdash \tau : t \Rightarrow t' : B) \\ \Downarrow \\ \tau : t \Rightarrow t' : A_1, \dots, A_n \rightarrow B \end{array}$$

what is the internal language of the free biclone on  $G$ ?



## A type theory for biclones = internal language of free biclone

Hom-categories  $(\mathcal{C}(X_1, \dots, X_n; Y), \bullet, \text{id})$

---

## A type theory for biclones = internal language of free biclone

Hom-categories  $(\mathcal{C}(X_1, \dots, X_n; Y), \bullet, \text{id})$

---

Judgements:

## A type theory for biclones = internal language of free biclone

Hom-categories  $(\mathcal{C}(X_1, \dots, X_n; Y), \bullet, \text{id})$

---

Judgements:

- Relating *terms*:  $\Gamma \vdash t : B$

## A type theory for biclones = internal language of free biclone

Hom-categories  $(\mathcal{C}(X_1, \dots, X_n; Y), \bullet, \text{id})$

---

Judgements:

- Relating *terms*:  $\Gamma \vdash t : B$
- Relating *rewrites*:  $\Gamma \vdash \tau : t \Rightarrow t' : B$

## A type theory for bicolones = internal language of free bicolone

Hom-categories  $(\mathcal{C}(X_1, \dots, X_n; Y), \bullet, \text{id})$

---

Judgements:

- Relating *terms*:  $\Gamma \vdash t : B$
- Relating *rewrites*:  $\Gamma \vdash \tau : t \Rightarrow t' : B$
- Equational theory  $\Gamma \vdash \tau \equiv \tau' : t \Rightarrow t' : B$

# A type theory for bicolones = internal language of free bicolone

Hom-categories  $(\mathcal{C}(X_1, \dots, X_n; Y), \bullet, \text{id})$

---

Judgements:

- Relating *terms*:  $\Gamma \vdash t : B$
- Relating *rewrites*:  $\Gamma \vdash \tau : t \Rightarrow t' : B$
- Equational theory  $\Gamma \vdash \tau \equiv \tau' : t \Rightarrow t' : B$

Vertical composition: 
$$\frac{\Gamma \vdash \tau' : t' \Rightarrow t'' : B \quad \Gamma \vdash \tau : t \Rightarrow t' : B}{\Gamma \vdash \tau' \bullet \tau : t \Rightarrow t'' : B}$$

Identities: 
$$\frac{\Gamma \vdash t : B}{\Gamma \vdash \text{id}_t : t \Rightarrow t : B}$$

## A type theory for biclones = internal language of free biclone

A *substitution functor*

$$\mathcal{C}(X_1, \dots, X_n; Y) \times \prod_{i=1}^n \mathcal{C}(\Gamma; X_i) \rightarrow \mathcal{C}(\Gamma; Y)$$

$$t, (u_1, \dots, u_n) \mapsto t[u_1, \dots, u_n]$$

$$\tau, (\sigma_1, \dots, \sigma_n) \mapsto \tau[\sigma_1, \dots, \sigma_n]$$

---

# A type theory for bicolones = internal language of free bicolone

A substitution functor

$$\begin{aligned} \mathcal{C}(X_1, \dots, X_n; Y) \times \prod_{i=1}^n \mathcal{C}(\Gamma; X_i) &\rightarrow \mathcal{C}(\Gamma; Y) \\ t, (u_1, \dots, u_n) &\mapsto t[u_1, \dots, u_n] \\ \tau, (\sigma_1, \dots, \sigma_n) &\mapsto \tau[\sigma_1, \dots, \sigma_n] \end{aligned}$$

Explicit substitution:

$$\frac{x_1 : A_1, \dots, x_n : A_n \vdash t : B \quad (\Delta \vdash u_i : A_i)_{i=1..n}}{\Delta \vdash t \{x_i \mapsto u_i\} : B}$$

$$\frac{x_1 : A_1, \dots, x_n : A_n \vdash \tau : t \Rightarrow t' : B \quad (\Delta \vdash \sigma_i : u_i \Rightarrow u'_i : A_i)_{i=1, \dots, n}}{\Delta \vdash \tau \{x_i \mapsto \sigma_i\} : t \{x_i \mapsto u_i\} \Rightarrow t' \{x_i \mapsto u'_i\} : B}$$

$\rightsquigarrow$  binds the variables  $x_1, \dots, x_n$



## A type theory for biclones

Structural isomorphisms  $\varrho^{(k)}, \iota, \text{assoc}$

---

# A type theory for biclones

Structural isomorphisms  $\varrho^{(k)}, \iota, \text{assoc}$

Distinguished invertible rewrites e.g.:

$$\frac{(\Delta \vdash u_i : A_i)_{i=1, \dots, n}}{x_1 : A_1, \dots, x_n : A_n \vdash \varrho_{u_\bullet}^{(k)} : x_k \{x_i \mapsto u_i\} \xrightarrow{\cong} u_k : A_k} \quad (1 \leq k \leq n)$$

the free bicategory on  $G$   
 $\simeq$  free biclone on  $G$ , restricted to unary maps

the free bicategory on  $G$   
 $\simeq$  free biclone on  $G$ , restricted to unary maps

syntax for bicategories  
 $\parallel$   
syntax for free biclone, restricted to unary contexts

the free bicategory on  $G$   
 $\simeq$  free biclone on  $G$ , restricted to unary maps

syntax for bicategories

||


syntax for free biclone, restricted to unary contexts

---

'internal language of bicategories' [informal]

||

internal language [formal] of the free biclone

1. Define fp-biclones,
2. Extract internal language  $\Lambda_{\text{ps}}^{\times}$  of free fp-biclone,
3. Restrict to unary contexts  
 get internal language of fp-bicategories.

1. Define fp-biclones,
2. Extract internal language  $\Lambda_{\text{ps}}^{\times}$  of free fp-biclone,
3. Restrict to unary contexts  
 $\rightsquigarrow$  get internal language of fp-bicategories.

An fp-clone is a clone  $(S, \mathbb{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$   
 inducing isomorphisms

$$\begin{array}{ccc}
 & (\pi_1[-], \dots, \pi_n[-]) & \\
 & \xrightarrow{\quad\quad\quad} & \\
 \mathbb{C}(\Gamma; \prod_n(A_1, \dots, A_n)) & \cong & \prod_{i=1}^n \mathbb{C}(\Gamma; A_i) \\
 & \xleftarrow{\quad\quad\quad} & \\
 & \langle -, \dots, = \rangle & 
 \end{array}$$

$\rightsquigarrow$  equivalent to asking that MC is representable.

1. Define fp-biclones,
2. Extract internal language  $\Lambda_{\text{ps}}^{\times}$  of free fp-biclone,
3. Restrict to unary contexts  
 $\rightsquigarrow$  get internal language of fp-bicategories.

An fp-biclone is a biclone  $(S, \mathcal{C})$  with

1. for every  $A_1, \dots, A_n$ , an object  $\prod_n(A_1, \dots, A_n)$ ,
2. maps  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i$   
 inducing adjoint equivalences

$$\begin{array}{ccc}
 & (\pi_1[-], \dots, \pi_n[-]) & \\
 & \curvearrowright & \\
 \mathcal{C}(\Gamma; \prod_n(A_1, \dots, A_n)) & \perp \simeq & \prod_{i=1}^n \mathcal{C}(\Gamma; A_i) \\
 & \curvearrowleft & \\
 & \langle -, \dots, = \rangle & 
 \end{array}$$

$\rightsquigarrow$  equivalent to asking that  $\text{MC}$  is representable.



A type theory for fp-bicategories = internal language of free fp-biclone

## A type theory for fp-bicategories = internal language of free fp-biclone

1-cells  $\pi_i : \prod_n (A_1, \dots, A_n) \rightarrow A_i \quad (1 \leq i \leq n)$

---

# A type theory for fp-bicategories = internal language of free fp-biclone

1-cells  $\pi_i : \prod_n(A_1, \dots, A_n) \rightarrow A_i \quad (1 \leq i \leq n)$

Projections  $\frac{}{p : \prod_n(A_1, \dots, A_n) \vdash \pi_i(p) : A_i} \quad (1 \leq i \leq n)$

# A type theory for fp-bicategories = internal language of free fp-biclone

$$\text{Equivalences } \mathcal{B}(\Gamma, \prod_n(A_1, \dots, A_n)) \underset{\langle -, \dots, = \rangle}{\overset{(\pi_1[-], \dots, \pi_n[-])}{\perp \simeq}} \prod_{i=1}^n \mathcal{B}(\Gamma, A_i)$$

## A type theory for fp-bicategories = internal language of free fp-biclone

$$\text{Equivalences } \mathcal{B}(\Gamma, \prod_n(A_1, \dots, A_n)) \underset{\langle -, \dots, = \rangle}{\overset{(\pi_1[-], \dots, \pi_n[-])}{\simeq}} \prod_{i=1}^n \mathcal{B}(\Gamma, A_i)$$

- Mapping on objects  $\langle -, \dots, = \rangle$ .

# A type theory for fp-bicategories = internal language of free fp-biclone

$$\text{Equivalences } \mathcal{B}(\Gamma, \prod_n(A_1, \dots, A_n)) \underset{\langle -, \dots, = \rangle}{\overset{(\pi_1[-], \dots, \pi_n[-])}{\perp \simeq}} \prod_{i=1}^n \mathcal{B}(\Gamma, A_i)$$

- Mapping on objects  $\langle -, \dots, = \rangle$ .
- Counit  $\varpi_{t_\bullet}^{(i)} : \pi_i[\langle t_1, \dots, t_n \rangle] \xrightarrow{\cong} t_i$

# A type theory for fp-bicategories = internal language of free fp-biclone

$$\text{Equivalences } \mathcal{B}(\Gamma, \prod_n(A_1, \dots, A_n)) \underset{\langle -, \dots, = \rangle}{\overset{(\pi_1[-], \dots, \pi_n[-])}{\perp \simeq}} \prod_{i=1}^n \mathcal{B}(\Gamma, A_i)$$

- Mapping on objects  $\langle -, \dots, = \rangle$ .
- Counit  $\varpi_{t_\bullet}^{(i)} : \pi_i[\langle t_1, \dots, t_n \rangle] \xrightarrow{\cong} t_i$
- For every  $(\alpha_i : \pi_i[u] \Rightarrow t_i)_{i=1, \dots, n}$ ,

$$\begin{array}{ccc} & \pi_i[\langle t_1, \dots, t_n \rangle] & \\ \pi_i[\mathbf{p}^\dagger(\alpha_1, \dots, \alpha_n)] \nearrow & \xrightarrow{\quad} & \searrow \varpi_{t_\bullet}^{(i)} \\ \pi_i[u] & \xrightarrow{\quad \alpha_i \quad} & t_i \end{array}$$

$\exists!$

# A type theory for fp-bicategories = internal language of free fp-biclone

$$\text{Equivalences } \mathcal{B}(\Gamma, \prod_n(A_1, \dots, A_n)) \underset{\langle -, \dots, = \rangle}{\overset{(\pi_1\{-\}, \dots, \pi_n\{-\})}{\simeq}} \prod_{i=1}^n \mathcal{B}(\Gamma, A_i)$$

$$\text{Tupling map } \frac{(\Gamma \vdash t_i : A_i)_{i=1, \dots, n}}{\Gamma \vdash \text{tup}(t_1 \dots, t_n) : \prod_n(A_1, \dots, A_n)}$$



# A type theory for fp-bicategories = internal language of free fp-biclone

$$\text{Equivalences } \mathcal{B}(\Gamma, \prod_n(A_1, \dots, A_n)) \underset{\langle -, \dots, = \rangle}{\overset{(\pi_1\{-\}, \dots, \pi_n\{-\})}{\simeq}} \prod_{i=1}^n \mathcal{B}(\Gamma, A_i)$$

$$\text{Tupling map } \frac{(\Gamma \vdash t_i : A_i)_{i=1, \dots, n}}{\Gamma \vdash \text{tup}(t_1 \dots, t_n) : \prod_n(A_1, \dots, A_n)}$$

$$\text{Counit } (\beta\text{-law}) \frac{(\Gamma \vdash t_i : A_i)_{i=1, \dots, n}}{\Gamma \vdash \varpi_{t_\bullet}^{(k)} : \pi_k \{ \text{tup}(t_1 \dots, t_n) \} \xrightarrow{\cong} t_k : A_k} \quad (1 \leq k \leq n)$$

# A type theory for fp-bicategories = internal language of free fp-biclone

$$\text{Equivalences } \mathcal{B}(\Gamma, \prod_n(A_1, \dots, A_n)) \underset{\langle -, \dots, = \rangle}{\overset{(\pi_1\{-\}, \dots, \pi_n\{-\})}{\simeq}} \prod_{i=1}^n \mathcal{B}(\Gamma, A_i)$$

$$\text{Tupling map } \frac{(\Gamma \vdash t_i : A_i)_{i=1, \dots, n}}{\Gamma \vdash \text{tup}(t_1 \dots, t_n) : \prod_n(A_1, \dots, A_n)}$$

$$\text{Count } (\beta\text{-law}) \frac{(\Gamma \vdash t_i : A_i)_{i=1, \dots, n}}{\Gamma \vdash \varpi_{t_\bullet}^{(k)} : \pi_k \{ \text{tup}(t_1 \dots, t_n) \} \cong t_k : A_k} \quad (1 \leq k \leq n)$$

$$\text{Mediating 2-cell} \frac{(\Gamma \vdash \alpha_j : \pi_j \{u\} \Rightarrow t_j : A_j)_{j=1, \dots, n}}{\Gamma \vdash p^\dagger(\alpha_1, \dots, \alpha_n) : u \Rightarrow \text{tup}(t_1, \dots, t_n) : \prod_n(A_1, \dots, A_n)}$$

+ three equational rules.

$\rightsquigarrow$   $\eta$ -law is derivable

the free fp-bicategory on  $G$   
 $\simeq$  free fp-biclone on  $G$ , restricted to unary maps

the free fp-bicategory on  $G$   
 $\simeq$  free fp-biclone on  $G$ , restricted to unary maps

syntax for fp-bicategories  
 $\parallel$   
syntax for free fp-biclone, restricted to unary contexts

the free fp-bicategory on  $G$   
 $\simeq$  free fp-biclone on  $G$ , restricted to unary maps

syntax for fp-bicategories

||

syntax for free fp-biclone, restricted to unary contexts

---


'internal language of fp-bicategories' [informal]

||

internal language [formal] of the free fp-biclone


## The recipe for $\Lambda_{ps}^{\times, \rightarrow}$

---

1. Define **cc**-biclones,
2. Extract internal language  $\Lambda_{ps}^{\times, \rightarrow}$  of free cc-biclone,
3. Restrict to unary contexts  
 get internal language of cc-bicategories.

The recipe for  $\Lambda_{ps}^{\times, \rightarrow}$

---

1. Define **cc**-biclones,
2. Extract internal language  $\Lambda_{ps}^{\times, \rightarrow}$  of free cc-biclone,
3. Restrict to unary contexts  
 get internal language of cc-bicategories.

'internal language for cc-bicategories' [informal]


||

internal language [formal] of free cc-biclone

# $\Lambda_{\text{ps}}^{\times, \rightarrow}$ as STLC up-to-isomorphism



# $\Lambda_{\text{ps}}^{\times, \rightarrow}$ as STLC up-to-isomorphism

 Rewrites in  $\Lambda_{\text{ps}}^{\times, \rightarrow}$  witness  $\beta\eta$ -equalities

# $\Lambda_{\text{ps}}^{\times, \rightarrow}$ as STLC up-to-isomorphism

↪ Rewrites in  $\Lambda_{\text{ps}}^{\times, \rightarrow}$  witness  $\beta\eta$ -equalities

STLC-terms **embed** into  $\Lambda_{\text{ps}}^{\times, \rightarrow}$

$\Lambda_{\text{ps}}^{\times, \rightarrow}$ -terms **evaluate** to STLC-terms  $\overline{t\{x \mapsto u\}} := \bar{t}[x \mapsto \bar{u}]$

# $\Lambda_{\text{ps}}^{\times, \rightarrow}$ as STLC up-to-isomorphism

↪ Rewrites in  $\Lambda_{\text{ps}}^{\times, \rightarrow}$  witness  $\beta\eta$ -equalities

STLC-terms **embed** into  $\Lambda_{\text{ps}}^{\times, \rightarrow}$

$\Lambda_{\text{ps}}^{\times, \rightarrow}$ -terms **evaluate** to STLC-terms  $\overline{t\{x \mapsto u\}} := \bar{t}[x \mapsto \bar{u}]$

$$(\text{STLC-terms } \Gamma \vdash t : B) / \beta\eta \cong (\Lambda_{\text{ps}}^{\times, \rightarrow}\text{-terms } \Gamma \vdash t : B) / \cong_B^\Gamma$$

$$t \cong_B^\Gamma t' \stackrel{\text{def.}}{\iff} \Gamma \vdash \tau : t \stackrel{\cong}{\implies} t' : B \text{ for some invertible } \tau$$

# $\Lambda_{\text{ps}}^{\times, \rightarrow}$ as STLC up-to-isomorphism

↪ Rewrites in  $\Lambda_{\text{ps}}^{\times, \rightarrow}$  witness  $\beta\eta$ -equalities

STLC-terms **embed** into  $\Lambda_{\text{ps}}^{\times, \rightarrow}$

$\Lambda_{\text{ps}}^{\times, \rightarrow}$ -terms **evaluate** to STLC-terms  $\overline{t\{x \mapsto u\}} := \bar{t}[x \mapsto \bar{u}]$

$$(\text{STLC-terms } \Gamma \vdash t : B) / \beta\eta \cong (\Lambda_{\text{ps}}^{\times, \rightarrow}\text{-terms } \Gamma \vdash t : B) / \cong_{\Gamma, B}$$

$$(\Gamma \vdash t =_{\beta\eta} t' : B) \iff (\Gamma \vdash \tau : \langle t \rangle \cong \langle t' \rangle : B)$$

$$t \cong_{\Gamma, B} t' \stackrel{\text{def.}}{\iff} \Gamma \vdash \tau : t \cong t' : B \text{ for some invertible } \tau$$

# $\Lambda_{\text{ps}}^{\times, \rightarrow}$ as a logic of program transformations

$$(\text{STLC-terms } \Gamma \vdash t : B) /_{\beta\eta} \cong (\Lambda_{\text{ps}}^{\times, \rightarrow}\text{-terms } \Gamma \vdash t : B) /_{\cong_{\Gamma B}}$$

$$(\Gamma \vdash t =_{\beta\eta} t' : B) \iff (\Gamma \vdash \tau : \langle t \rangle \xrightarrow{\cong} \langle t' \rangle : B)$$

# $\Lambda_{\text{ps}}^{\times, \rightarrow}$ as a logic of program transformations

$$(\text{STLC-terms } \Gamma \vdash t : B) /_{\beta\eta} \cong (\Lambda_{\text{ps}}^{\times, \rightarrow}\text{-terms } \Gamma \vdash t : B) /_{\cong_{\Gamma B}}$$

$$(\Gamma \vdash t =_{\beta\eta} t' : B) \iff (\Gamma \vdash \tau : \langle t \rangle \xrightarrow{\cong} \langle t' \rangle : B)$$

# $\Lambda_{ps}^{\times, \rightarrow}$ as a logic of program transformations

$$(\text{STLC-terms } \Gamma \vdash t : B) /_{\beta\eta} \cong (\Lambda_{ps}^{\times, \rightarrow}\text{-terms } \Gamma \vdash t : B) /_{\cong_{\Gamma B}}$$

$$(\Gamma \vdash t =_{\beta\eta} t' : B) \iff (\Gamma \vdash \tau : \langle t \rangle \xrightarrow{\cong} \langle t' \rangle : B)$$

↪ Equational theory = quotient out **loops** in rewriting:

$$\begin{array}{ccc}
 & \pi_i \{ \langle \pi_1 \circ t, \dots, \pi_n \circ t \rangle \} & \\
 \pi_i(\rho) \{ \rho \mapsto \eta \} \nearrow & & \searrow \beta \\
 \pi_i \{ t \} & \xlongequal{\quad} & \pi_i \{ t \}
 \end{array}$$

# $\Lambda_{ps}^{\times, \rightarrow}$ as a logic of program transformations

$$(\text{STLC-terms } \Gamma \vdash t : B) /_{\beta\eta} \cong (\Lambda_{ps}^{\times, \rightarrow}\text{-terms } \Gamma \vdash t : B) /_{\cong_{\Gamma B}}$$

$$(\Gamma \vdash t =_{\beta\eta} t' : B) \iff (\Gamma \vdash \tau : \langle t \rangle \cong \langle t' \rangle : B)$$

→ Equational theory = quotient out **loops** in rewriting:

$$\begin{array}{ccc}
 & \pi_i \{ \langle \pi_1 \circ t, \dots, \pi_n \circ t \rangle \} & \\
 \pi_i(p) \{ p \rightarrow \eta \} \nearrow & & \searrow \beta \\
 \pi_i \{ t \} & \xlongequal{\quad} & \pi_i \{ t \}
 \end{array}$$

→ With **coherence**: rewrites unique mod  $\equiv$



Coherence à la Mac-Lane: 'all diagrams commute'

# Using $\Lambda_{ps}^{\times, \rightarrow}$ to prove coherence [LICS'20, thesis]

Coherence à la Mac-Lane:

if  $\sigma, \tau : t \Rightarrow t'$  then  $\sigma \equiv \tau$

# Using $\Lambda_{\text{ps}}^{\times, \rightarrow}$ to prove coherence [LICS'20, thesis]

Coherence à la Mac-Lane:

if  $\sigma, \tau : t \Rightarrow t'$  then  $\sigma \equiv \tau$

Two proofs for cc-bicategories:

1. via the Yoneda embedding,
2. prove coherence of  $\Lambda_{\text{ps}}^{\times, \rightarrow}$

# Using $\Lambda_{ps}^{\times, \rightarrow}$ to prove coherence [LICS'20, thesis]

Coherence à la Mac-Lane: if  $\sigma, \tau : t \Rightarrow t'$  then  $\sigma \equiv \tau$

Two proofs for cc-bicategories:

1. via the Yoneda embedding,
2. prove coherence of  $\Lambda_{ps}^{\times, \rightarrow}$

Strategy:

bicategorify Fiore's [semantic normalisation-by-evaluation](#) for STLC

## Summary and future questions

- STLC is the internal language of the free cc-clone.

## Summary and future questions

- STLC is the internal language of the free cc-clone.
- To construct an internal language for cc-bicategories:
  1. Define cc-biclones,
  2. The internal language of cc-bicategories  $\Lambda_{ps}^{\times, \rightarrow}$  is the internal language of the free cc-biclone.

## Summary and future questions

- STLC is the internal language of the free cc-clone.
- To construct an internal language for cc-bicategories:
  1. Define cc-biclones,
  2. The internal language of cc-bicategories  $\Lambda_{\text{ps}}^{\times, \rightarrow}$  is the internal language of the free cc-biclone.
- $\Lambda_{\text{ps}}^{\times, \rightarrow}$  is a calculus for  $\beta\eta$ -rewriting modulo a specific equational theory.

## Summary and future questions

- STLC is the internal language of the free cc-clone.
- To construct an internal language for cc-bicategories:
  1. Define cc-biclones,
  2. The internal language of cc-bicategories  $\Lambda_{\text{ps}}^{\times, \rightarrow}$  is the internal language of the free cc-biclone.
- $\Lambda_{\text{ps}}^{\times, \rightarrow}$  is a calculus for  $\beta\eta$ -rewriting modulo a specific equational theory.

- Refine NbE argument to extract canonical **normal forms**.
- Does a similar argument work for the **linear** case?
- What does this say about bicategorical models of LL?
- What **denotational** meaning can we give to rewrites?
- Proof-theoretic implications?





1-cells

$$\text{eval}_{A,B} : (A \Rightarrow B) \times A \rightarrow B$$

Adjoint equivalences

$$\mathcal{B}(X, A \Rightarrow B) \begin{array}{c} \xrightarrow{\text{eval}_{A,B} \circ (- \times A)} \\ \perp \simeq \\ \xleftarrow{\lambda} \end{array} \mathcal{B}(X \times A, B)$$

# Rules for exponentials

$$\begin{array}{c}
 \text{explicit weakening by } x \quad \leftarrow \quad (x : A) \quad \leftarrow \quad \text{free variable in context} \\
 \left( \text{eval } \{(-)\} \{inc_x\}, x \right) \left( \frac{\text{eval } \{u\} \{inc_x\}, x \Rightarrow t : B}{u \Rightarrow \lambda x.t : A \Rightarrow B} \right) \left( e^\dagger(x. -) \right)
 \end{array}$$

$$\frac{}{f : A \Rightarrow B, x : A \vdash \text{eval}(f, x) : B} \text{eval} \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \Rightarrow B} \text{lam}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma, x : A \vdash \epsilon_t : \text{eval} \{(\lambda x.t)\} \{inc_x\}, x \Rightarrow t : B} \text{e-intro } (\beta\text{-rule})$$

$$\frac{\Gamma, x : A \vdash t : B \quad \Gamma \vdash u : A \Rightarrow B \quad \Gamma, x : A \vdash \alpha : \text{eval} \{u\} \{inc_x\}, x \Rightarrow t : B}{\Gamma \vdash e^\dagger(x.\alpha) : u \Rightarrow \lambda x.t : A \Rightarrow B} e^\dagger(x.\alpha)\text{-intro}$$

+ three equational rules

$\rightsquigarrow$   $\eta$ -rule derivable