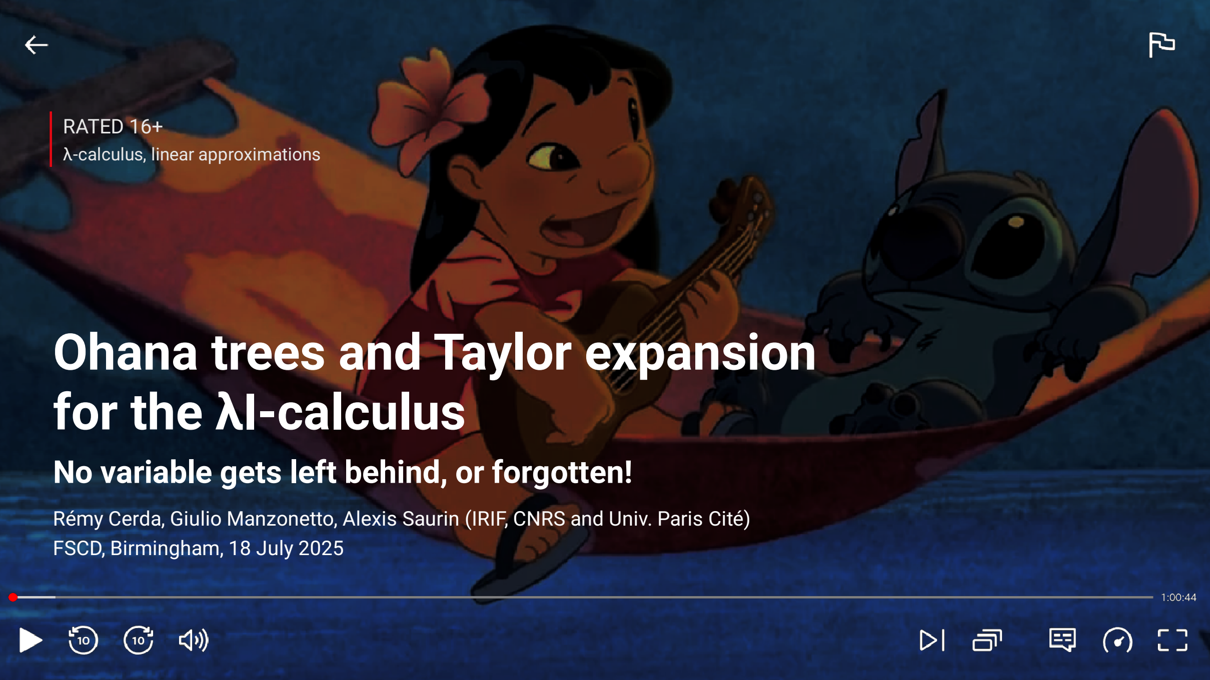


RATED 16+
λ-calculus, linear approximations

Ohana trees and Taylor expansion for the λI-calculus

No variable gets left behind, or forgotten!

Rémy Cerda, Giulio Manzonetto, Alexis Saurin (IRIF, CNRS and Univ. Paris Cité)

FSCD, Birmingham, 18 July 2025

1:00:44

Introducing Ohana trees

From λ-theories...

λ

A model of
the λ-calculus

$\rightsquigarrow$

A λ-theory

$[\![-]\!]$

$M =_{\mathcal{T}} N$ iff $[\![M]\!] = [\![N]\!]$

From λ-theories...

λ

A model of the λ-calculus $\rightsquigarrow$ A λ-theory

$[\![-]\!]$ $\qquad$ $M =_{\mathcal{T}} N$ iff $[\![M]\!] = [\![N]\!]$

A **λ-theory** $\mathcal{T}$ is a set of equalities between λ-terms such that

$$\underbrace{\frac{M =_\beta N}{M =_{\mathcal{T}} N}}_{\text{it contains } \beta\text{-conversion}} \qquad \underbrace{\frac{P =_{\mathcal{T}} P'}{\lambda x.P =_{\mathcal{T}} \lambda x.P'} \qquad \frac{P =_{\mathcal{T}} P' \qquad Q =_{\mathcal{T}} Q'}{PQ =_{\mathcal{T}} P'Q'}}_{\text{it is stable under contexts}}$$

From λ-theories...

λ

A model of the λ-calculus $\rightsquigarrow$ A λ-theory $\leftsquigarrow$ A notion of evaluation tree

$M =_{\mathcal{B}} N$ iff $\mathrm{BT}(M) = \mathrm{BT}(N)$

$\mathrm{BT}(-)$

From λ-theories...

λ

A model of the λ-calculus ⇝ A λ-theory ⇜ A notion of evaluation tree

$M =_{\mathcal{B}} N$ iff $\mathrm{BT}(M) = \mathrm{BT}(N)$

$\mathrm{BT}(-)$

The **Böhm tree** of a λ-term M is defined **coinductively** by

$$\mathrm{BT}(M) := \begin{cases} \begin{array}{ccc} & \lambda x_1 \ldots x_n . y & \\ \mathrm{BT}(M_1) & \cdots & \mathrm{BT}(M_k) \end{array} & \text{if } M \twoheadrightarrow_h \lambda x_1 \ldots x_n . y M_1 \cdots M_k, \\ \perp & \text{otherwise.} \end{cases}$$

For example: $\mathrm{BT}(\mathrm{I}) := \mathrm{I}$ $\quad \mathrm{BT}(\Omega) := \perp$ $\quad \mathrm{BT}(\mathrm{Y}) := \lambda f . f(f(f(\ldots)))$

From λ-theories… to λI-theories

λ	A model of the λ-calculus	$\rightsquigarrow$	A λ-theory	$\leftsquigarrow$	A notion of evaluation tree
λI					

λ | A model of the λ-calculus $\rightsquigarrow$ A λ-theory $\leftsquigarrow$ A notion of evaluation tree

λI

The **λI-calculus** is the fragment of the λ-calculus **without erasure**. Formally, $\Lambda_\mathsf{I} \coloneqq \bigcup_{X \subseteq_\mathsf{f} \mathcal{V}} \Lambda_\mathsf{I}(X)$, where $\Lambda_\mathsf{I}(X)$ is the set of λI-terms with free variables in X:

$$\frac{}{x \in \Lambda_\mathsf{I}(\{x\})} \qquad \frac{M \in \Lambda_\mathsf{I}(X) \quad x \in X}{\lambda x.M \in \Lambda_\mathsf{I}(X \setminus \{x\})} \qquad \frac{M \in \Lambda_\mathsf{I}(X) \quad N \in \Lambda_\mathsf{I}(Y)}{MN \in \Lambda_\mathsf{I}(X \cup Y)}$$

For example: $\lambda x.x \in \Lambda_\mathsf{I}(\emptyset)$ $\quad \lambda x.xy \in \Lambda_\mathsf{I}(\{y\})$ $\quad \lambda x.y \notin \Lambda_\mathsf{I}$

From λ-theories... to λI-theories

λ	A model of the λ-calculus	$\rightsquigarrow$	A λ-theory	$\leftsquigarrow$	A notion of evaluation tree
λI			???		

λ	A model of the λ-calculus	$\rightsquigarrow$	A λ-theory	$\leftsquigarrow$	A notion of evaluation tree
λI			???		

A **λI-theory** $\mathcal{T}$ is a set of equalities between λ-terms such that

$$\underbrace{\frac{M =_\beta N}{M =_{\mathcal{T}} N}}_{\text{it contains } \beta\text{-conversion}} \qquad \underbrace{\frac{P =_{\mathcal{T}} P' \qquad x \in \mathrm{fv}(P) \cap \mathrm{fv}(P')}{\lambda x.P =_{\mathcal{T}} \lambda x.P'} \qquad \frac{P =_{\mathcal{T}} P' \qquad Q =_{\mathcal{T}} Q'}{PQ =_{\mathcal{T}} P'Q'}}_{\text{it is stable under } \lambda\text{I-contexts}}$$

- Every λ-theory restricted to Λ_I is a λI-theory.
- The converse is false, *e.g.* the λI-theory generated by equating all λI-terms without β-nf.

From λ-theories... to λI-theories

λ	A model of the λ-calculus	$\rightsquigarrow$	A λ-theory	$\leftsquigarrow$	A notion of evaluation tree
λI			A λI-theory	$\leftsquigarrow$	**???**

λ	A model of the λ-calculus	$\rightsquigarrow$	A λ-theory	$\leftsquigarrow$	A notion of evaluation tree
λI			A λI-theory	$\leftsquigarrow$	**???**

Böhm trees still generate a λI-theory $\mathcal{B}$... but behave poorly wrt. Λ_{I}:

$$M \text{ is a } \lambda\text{I-term} \quad \not\Rightarrow \quad \mathrm{BT}(M) \text{ is an “infinitary } \lambda\text{I-term”}$$

Indeed, abstracted variables may be:

- **left behind** an unsolvable subterm: $\mathrm{BT}(\lambda xy.x(\Omega y)) = \lambda xy.x\bot$.
- **forgotten** along infinite computations:
  if $Mxf \twoheadrightarrow_\beta f(Mxf)$, then $\mathrm{BT}(M) = \lambda xf.f(f(f(\dots)))$.

The **Ohana tree** of a λI-term M is defined coinductively by

$$\mathrm{OT}(M) := \begin{cases} \begin{tikzcd}[column sep=small] & \lambda x_1 \ldots x_n.y \arrow[dl, dash, "\mathrm{fv}(M_1)"'] \arrow[dr, dash, "\mathrm{fv}(M_k)"] & \\ \mathrm{OT}(M_1) & \cdots & \mathrm{OT}(M_k) \end{tikzcd} & \text{if } M \twoheadrightarrow_h \lambda x_1 \ldots x_n.yM_1 \cdots M_k, \\ \perp_{\mathrm{fv}(M)} & \text{otherwise.} \end{cases}$$

No variable left behind!

$$\mathrm{OT}(M) := \begin{cases} \begin{tikzcd}[column sep=small] & \lambda x_1 \ldots x_n.y \arrow[dl, dash, "\mathrm{fv}(M_1)"'] \arrow[dr, dash, "\mathrm{fv}(M_k)"] & \\ \mathrm{OT}(M_1) & \cdots & \mathrm{OT}(M_k) \end{tikzcd} & \text{if } M \twoheadrightarrow_h \lambda x_1 \ldots x_n.yM_1 \cdots M_k, \\ \bot_{\mathrm{fv}(M)} & \text{otherwise.} \end{cases}$$

Whereas Böhm trees equate $\lambda x.x(\Omega y)(\Omega z)$ and $\lambda x.x(\Omega z)(\Omega y)$:

$$\mathrm{BT}(\lambda x.x(\Omega x)(\Omega y)) \;=\; \lambda x.x\bot\bot \;=\; \mathrm{BT}(\lambda x.x(\Omega y)(\Omega z))$$

Ohana trees do separate them:

$$\mathrm{OT}(\lambda x.x(\Omega x)(\Omega y)) \;=\; \lambda x.x\bot_{\{y\}}\bot_{\{z\}} \;\neq\; \lambda x.x\bot_{\{z\}}\bot_{\{y\}} \;=\; \mathrm{BT}(\lambda x.x(\Omega y)(\Omega z))$$

$$\mathrm{OT}(M) := \begin{cases} \lambda x_1 \ldots x_n.y \\ \quad {\scriptstyle \mathrm{fv}(M_1)} \quad\quad {\scriptstyle \mathrm{fv}(M_k)} \\ \mathrm{OT}(M_1) \quad \cdots \quad \mathrm{OT}(M_k) & \text{if } M \twoheadrightarrow_h \lambda x_1 \ldots x_n.yM_1 \cdots M_k, \\ \perp_{\mathrm{fv}(M)} & \text{otherwise.} \end{cases}$$

Klop's **Bible fixed-point combinator** is $🕮_l := \lambda e.\mathrm{BYBe}l =_\beta \lambda e.e(\mathrm{BYBe}l)$.

Whereas Böhm trees equate Y and $🕮_l$: $\mathrm{BT}(\mathrm{Y}) = \lambda f.f(f(\ldots)) = \mathrm{BT}(🕮_l)$,

Ohana trees do separate them: $\mathrm{OT}(\mathrm{Y}) = \lambda f.f \xrightarrow{\{f\}} f \xrightarrow{\{f\}} f \cdots \neq \lambda f.f \xrightarrow{\{f,l\}} f \xrightarrow{\{f,l\}} f \cdots = \mathrm{OT}(🕮_l)$.

Approximation theories for Ohana evaluation

The continuous approximation: It works as usual

- Add constant $\perp$ to the syntax.
  $\perp$ is "an undefined term".

The continuous approximation: It works as usual

- Add constants $\perp_X$ (for $X \subseteq_f \mathcal{V}$) to the syntax.
  $\perp_X$ is "an undefined term whose set of free variables is X".

The continuous approximation: It works as usual

- Add constants $\perp_{X}$ (for $X \subseteq_{\mathrm{f}} \mathcal{V}$) to the syntax.
  $\perp_X$ is "an undefined term whose set of free variables is X".
- Define a **(head) approximation ordering** by

$$\frac{}{\perp \sqsubseteq M} \qquad \frac{\forall i, \qquad M_i \sqsubseteq N_i}{\lambda \vec{x}.yM_1 \cdots M_k \sqsubseteq \lambda \vec{x}.yN_1 \cdots N_k}$$

The continuous approximation: It works as usual

- Add constants $\perp_X$ (for $X \subseteq_f \mathcal{V}$) to the syntax.
  $\perp_X$ is "an undefined term whose set of free variables is X".
- Define a **(head) approximation ordering** by

$$\frac{}{\perp_X \sqsubseteq M} \qquad \frac{\forall i, \quad M_i \sqsubseteq N_i \quad \mathrm{fv}(M_i) = \mathrm{fv}(N_i)}{\lambda \vec{x}.yM_1 \cdots M_k \sqsubseteq \lambda \vec{x}.yN_1 \cdots N_k}$$

- Add constants $\bot_X$ (for $X \subseteq_f \mathcal{V}$) to the syntax.
  $\bot_X$ is "an undefined term whose set of free variables is X".
- Define a **(head) approximation ordering** by

$$\frac{}{\bot_X \sqsubseteq M} \qquad \frac{\forall i, \quad M_i \sqsubseteq N_i \qquad \mathrm{fv}(M_i) = \mathrm{fv}(N_i)}{\lambda\vec{x}.yM_1 \cdots M_k \sqsubseteq \lambda\vec{x}.yN_1 \cdots N_k}$$

- Define **(head) approximants**:

$$\mathcal{A} \quad \ni \quad A, B, \ldots \quad := \quad \bot \quad | \quad \lambda x_1 \ldots x_n.yA_1 \cdots A_k$$

The continuous approximation: It works as usual

- Add constants $\bot_X$ (for $X \subseteq_f \mathcal{V}$) to the syntax.
  $\bot_X$ is "an undefined term whose set of free variables is X".
- Define a **(head) approximation ordering** by

$$\frac{}{\bot_X \sqsubseteq M} \qquad \frac{\forall i, \quad M_i \sqsubseteq N_i \quad \mathrm{fv}(M_i) = \mathrm{fv}(N_i)}{\lambda\vec{x}.yM_1 \cdots M_k \sqsubseteq \lambda\vec{x}.yN_1 \cdots N_k}$$

- Define **(head) approximants**:

$$\mathcal{A}_{\mathrm{m}}(X) \quad \ni \quad A, B, \ldots \quad := \quad \bot_X \quad | \quad \lambda x_1 \ldots x_n.yA_1 \cdots A_k \quad \text{s.t. } (\ldots)$$

- Add constants $\bot_X$ (for $X \subseteq_f \mathcal{V}$) to the syntax.
  $\bot_X$ is "an undefined term whose set of free variables is X".
- Define a **(head) approximation ordering** by

$$\frac{}{\bot_X \sqsubseteq M} \qquad \frac{\forall i, \quad M_i \sqsubseteq N_i \quad \mathrm{fv}(M_i) = \mathrm{fv}(N_i)}{\lambda\vec{x}.yM_1 \cdots M_k \sqsubseteq \lambda\vec{x}.yN_1 \cdots N_k}$$

- Define **(head) approximants**:

$$\mathcal{A}_{\mathrm{m}}(X) \quad \ni \quad A, B, \ldots \quad := \quad \bot_X \quad | \quad \lambda x_1 \ldots x_n.yA_1 \cdots A_k \quad \text{s.t. } (\ldots)$$

  and obtain the approximants of a λI-term:

$$\mathrm{App}_{\mathrm{m}}(M) := \{A \in \mathcal{A} \mid \exists M \twoheadrightarrow_\beta N,\ A \sqsubseteq N\}.$$

- Add constants $\bot_X$ (for $X \subseteq_f \mathcal{V}$) to the syntax.
  $\bot_X$ is "an undefined term whose set of free variables is X".
- Define a **(head) approximation ordering** by

$$\frac{}{\bot_X \sqsubseteq M} \qquad \frac{\forall i, \quad M_i \sqsubseteq N_i \quad \mathsf{fv}(M_i) = \mathsf{fv}(N_i)}{\lambda \vec{x}.yM_1 \cdots M_k \sqsubseteq \lambda \vec{x}.yN_1 \cdots N_k}$$

- Define **(head) approximants**:

$$\mathcal{A}_{\mathrm{m}}(X) \quad \ni \quad A, B, \ldots \quad := \quad \bot_X \quad | \quad \lambda x_1 \ldots x_n.yA_1 \cdots A_k \quad \text{s.t. } (\ldots)$$

  and obtain the approximants of a λI-term:

$$\mathrm{App}_{\mathrm{m}}(M) := \{A \in \mathcal{A} \mid \exists M \twoheadrightarrow_\beta N,\ A \sqsubseteq N\}.$$

- **Continuous approximation theorem:** $\quad \mathrm{BT}(M) = \bigsqcup \mathrm{App}_{\mathrm{m}}(M)$

- Add constants $\bot_X$ (for $X \subseteq_f \mathcal{V}$) to the syntax.
  $\bot_X$ is "an undefined term whose set of free variables is X".
- Define a **(head) approximation ordering** by

$$\frac{}{\bot_X \sqsubseteq M} \qquad \frac{\forall i, \quad M_i \sqsubseteq N_i \quad \mathsf{fv}(M_i) = \mathsf{fv}(N_i)}{\lambda\vec{x}.yM_1 \cdots M_k \sqsubseteq \lambda\vec{x}.yN_1 \cdots N_k}$$

- Define **(head) approximants**:

$$\mathcal{A}_{\mathrm{m}}(X) \quad \ni \quad A, B, \ldots \quad := \quad \bot_X \quad | \quad \lambda x_1 \ldots x_n.yA_1 \cdots A_k \quad \text{s.t. } (\ldots)$$

  and obtain the approximants of a λI-term:

$$\mathrm{App}_{\mathrm{m}}(M) := \{A \in \mathcal{A}_{\mathrm{m}} \mid \exists M \twoheadrightarrow_\beta N,\ A \sqsubseteq N\}.$$

- **Continuous approximation theorem:** $\mathrm{OT}(M) = \bigsqcup \mathrm{App}_{\mathrm{m}}(M)$

What you may usually call (multi)linear approximation:

Take a function:

$$f(x)$$

to a formal sum of n-linear approximations:

$$\sum_{n \in \mathbb{N}} \frac{1}{n!} \times \frac{\partial f(t)}{\partial t^n} \cdot x^n$$

via an operation of **Taylor expansion**.

What you will now call (multi)linear approximation:

Take a λ-term:

$$x \quad | \quad \lambda x.M \quad | \quad MN$$

to a formal sum of "multilinear λ-terms" (aka **resource λ-terms**):

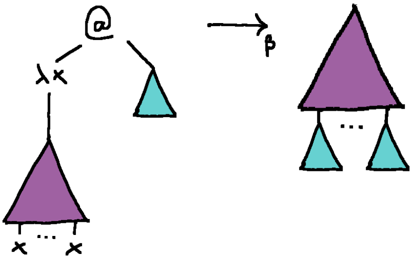
$$x \quad | \quad \lambda x.s \quad | \quad s[t_1, \ldots, t_n]$$

via an operation of **Taylor expansion**. [Ehrhard & Regnier '08]

The linear approximation (original ed.)

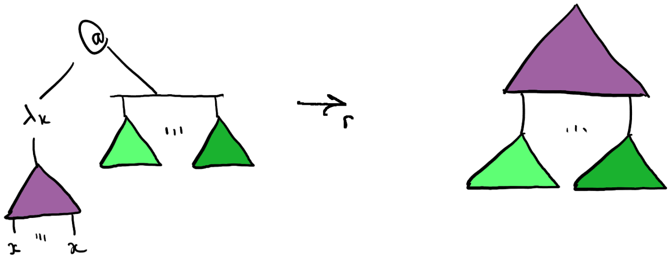
We are also able to simulate β-reduction of λ-terms:

$$(\lambda x.M)N \quad \to_\beta \quad M[N/x]$$



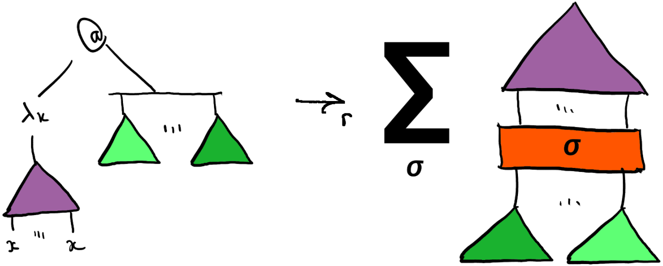
We are also able to simulate β-reduction of λ-terms...

using multilinear β-reduction of resource λ-terms:



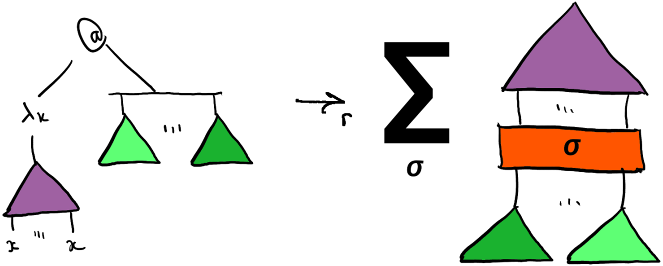
We are also able to simulate β-reduction of λ-terms...

using multilinear β-reduction of resource λ-terms:



We are also able to simulate β-reduction of λ-terms...

using multilinear β-reduction of resource λ-terms:



And in the usual setting we obtain the celebrated

Commutation theorem: $\mathrm{nf}(\mathcal{T}(M)) = \mathcal{T}(\mathrm{BT}(M))$. [Ehrhard & Regnier ’06]

In our setting, the resource λ-calculus is "fibered" over finite sets of variables:

$$\frac{}{x \in \Delta_{\mathsf{l}}(\{x\})} \qquad \frac{s \in \Delta_{\mathsf{l}}(X) \qquad x \in X}{\lambda x.s \in \Delta_{\mathsf{l}}(X - x)} \qquad \frac{s \in \Delta_{\mathsf{l}}(X)}{s[\,]_Y \in \Delta_{\mathsf{l}}(X \cup Y)} \qquad \frac{s \in \Delta_{\mathsf{l}}(X) \qquad t_1, \dots, t_{n+1} \in \Delta_{\mathsf{l}}(Y)}{s[t_1, \dots, t_{n+1}] \in \Delta_{\mathsf{l}}(X \cup Y)}$$

and (for Taylor expansion nerds only!) here's what changes in the substitution:

$$x\,\langle \bar{u}/x \rangle := \begin{cases} u & \text{if } \bar{u} = [u] \\ \mathbf{0}_{\mathrm{fv}(\bar{u})} & \text{otherwise} \end{cases} \qquad y\,\langle \bar{u}/x \rangle := \begin{cases} y & \text{if } \bar{u} = [\,]_{\mathrm{fv}(\bar{u})} \\ \mathbf{0}_{\{y\}} & \text{otherwise} \end{cases}$$

$$[\,]_X\,\langle \bar{u}/x \rangle := \begin{cases} [\,]_{X\{\mathrm{fv}(\bar{u})/x\}} & \text{if } \bar{u} = [\,]_{\mathrm{fv}(\bar{u})} \\ \mathbf{0}_{X\{\mathrm{fv}(\bar{u})/x\}} & \text{otherwise} \end{cases}$$

We denote by $\mathcal{T}_{\mathrm{m}}(M)$ the **Taylor expansion with memory** of a λI-term M.
We extend the construction to Ohana trees.

Commutation theorem

$\mathrm{nf}(\mathcal{T}_{\mathrm{m}}(M)) = \mathcal{T}_{\mathrm{m}}(\mathrm{OT}(M))$.

The corollary we wanted

The set of equations $\mathcal{O} := \{M = N \mid \mathrm{OT}(M) = \mathrm{OT}(N)\}$ is a λI-theory!

Conclusion and further work

What happened so far…

λ	A model of the λ-calculus	$\rightsquigarrow$	A λ-theory	$\leftsquigarrow$	A notion of evaluation tree
			$M =_{\mathcal{B}} N$ iff $\mathrm{BT}(M) = \mathrm{BT}(N)$		$\mathrm{BT}(-)$
$\lambda\mathrm{I}$			A $\lambda\mathrm{I}$-theory	$\leftsquigarrow$	A notion of evaluation tree
			$M =_{\mathcal{O}} N$ iff $\mathrm{OT}(M) = \mathrm{OT}(N)$		$\mathrm{OT}(-)$

What happened so far… and what's going on

λ	A model of the λ-calculus	$\rightsquigarrow$	A λ-theory	$\leftsquigarrow$	A notion of evaluation tree
			$M =_{\mathcal{B}} N$ iff $\mathrm{BT}(M) = \mathrm{BT}(N)$		$\mathrm{BT}(-)$
λI	???	$\rightsquigarrow$	A λI-theory	$\leftsquigarrow$	A notion of evaluation tree
			$M =_{\mathcal{O}} N$ iff $\mathrm{OT}(M) = \mathrm{OT}(N)$		$\mathrm{OT}(-)$

- Is there a λI-model whose theory is $\mathcal{O}$?
  - No straightforward way to turn our Taylor expansion into a relational model ☹
- Actually, what is a λI-model?
  - Should be something more general than a λ-model
  - Our candidate: a cartesian closed multicategory with contractions

Ohana trees for the full λ-calculus?

By the way, the Ohana tree of a λ-term M can be defined coinductively by

$$\mathrm{OT}(M) := \begin{cases} \begin{array}{ccc} & \lambda x_1 \dots x_n.y & \\ \swarrow \mathrm{pfv}(M_1) & & \mathrm{pfv}(M_k) \searrow \\ \mathrm{OT}(M_1) & \cdots & \mathrm{OT}(M_k) \end{array} & \text{if } M \twoheadrightarrow_h \lambda x_1 \dots x_n.yM_1 \cdots M_k, \\ \perp_{\mathrm{pfv}(M)} & \text{otherwise.} \end{cases}$$

where

$$\mathrm{pfv}(M) := \{x \in \mathcal{V} \mid \forall M \twoheadrightarrow_\beta N,\ x \in \mathrm{fv}(N)\}.$$

But Ohana tree equality does **not** induce a λ-theory. ☹

May Ohana trees induce an interesting observational equivalence?

Long-term goal: To infinity and beyond

Why do we care about all that?

- It's fun. (Right?)

Long-term goal: To infinity and beyond

Why do we care about all that?

- It's fun. (Right?)
- Finer models/theories allow to separate more non-β-convertible λ-terms.
    - Example application: the famous **double fixed-point combinator** open problem (is there a FPC Y s.t. $Y =_\beta Y\delta$, for $\delta := \lambda yx.x(yx)$?).

Why do we care about all that?

- It's fun. (Right?)
- Finer models/theories allow to separate more non-β-convertible λ-terms.
    - Example application: the famous **double fixed-point combinator** open problem (is there a FPC Y s.t. $Y =_\beta Y\delta$, for $\delta := \lambda yx.x(yx)$?).
- Our formalism accounts for free variables pushed to infinity. It is a first step: what about pushing whole subterms to infinity?
    - This suggests working with **transfinite terms** and investigate the associated rewriting.
    - It is not clear what evaluation trees and semantics may look like in such a jungle. (But we have some preliminary ideas.)

Thanks for your attention!