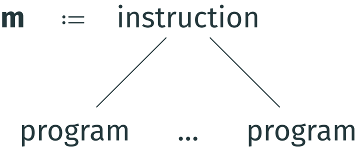
INFINITARY REWRITING FOR THE LINEAR APPROXIMATION OF FUNCTIONAL PROGRAMS

Rémy Cerda
Bologna, 4 February 2026

LINEAR APPROXIMATION OF FUNCTIONAL PROGRAMS

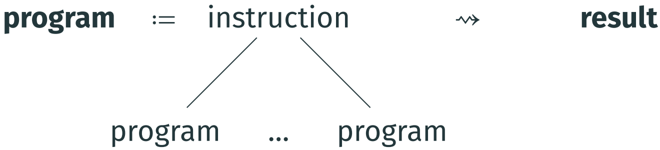
program :=



instruction

program ... program

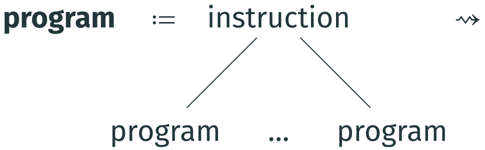
LINEAR APPROXIMATION OF FUNCTIONAL PROGRAMS



program := instruction $\rightsquigarrow$ **result**

program … program

LINEAR APPROXIMATION OF FUNCTIONAL PROGRAMS



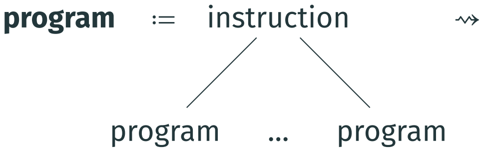
program := instruction ⇝

program … program

result

= program that cannot be further executed (*normal form*)

LINEAR APPROXIMATION OF FUNCTIONAL PROGRAMS

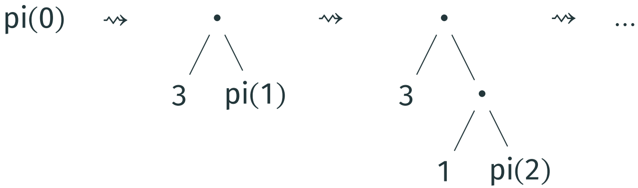


program := instruction ⇝ **result**

program … program

= program that cannot be further executed (*normal form*)

But the result might be *infinite* and *infinitely far*:



pi(0) ⇝ • ⇝ • ⇝ …

3 pi(1)

3 •

1 pi(2)

LINEAR APPROXIMATION OF FUNCTIONAL PROGRAMS

What we can compute in finite time are *partial results*:

LINEAR APPROXIMATION OF FUNCTIONAL PROGRAMS

What we can compute in finite time are *partial results*:

As a summary: [Wadsworth, Hyland, Barendregt, 1970s]

$$\begin{tikzcd}
\text{program} \arrow[r, squiggly, "\text{(infinite) execution}"] \arrow[d, equal] & \text{result} \arrow[d, phantom, "\rotatebox{90}{$\sqsubseteq$}"] \\
\text{program} \arrow[r, squiggly, "\text{(finite) partial execution}"'] & \text{partial result}
\end{tikzcd}$$

and the result is the *limit* of all partial results:
it's a *continuous* approximation.

LINEAR APPROXIMATION OF FUNCTIONAL PROGRAMS

This continuous approximation:

$$\begin{tikzcd}
\text{program} \arrow[r, rightsquigarrow, "\text{(infinite) execution}"] \arrow[d, equal] & \text{result} \arrow[d, phantom, sloped, "\sqsupseteq"] \\
\text{program} \arrow[r, rightsquigarrow, "\text{(finite) partial execution}"'] & \text{partial result}
\end{tikzcd}$$

can be refined into a *linear* one: [Ehrhard-Regnier, 2000s]

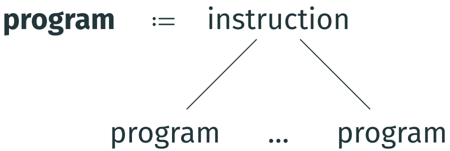
$$\begin{tikzcd}
\text{program} \arrow[r, rightsquigarrow, "\text{(infinite) execution}"] \arrow[d, phantom, sloped, "\sqsupseteq"] & \text{result} \arrow[d, phantom, sloped, "\sqsupseteq"] \\
\text{linear program} \arrow[r, rightsquigarrow, "\text{linear execution (finite!)}"'] & \text{linear partial result}
\end{tikzcd}$$

LINEAR APPROXIMATION OF FUNCTIONAL PROGRAMS

Linear programs: each argument of a function is used *exactly once*.

LINEAR APPROXIMATION OF FUNCTIONAL PROGRAMS

Linear programs: each argument of a function is used *exactly once*.

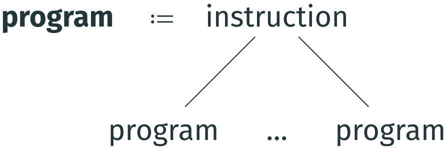


program := instruction

program … program

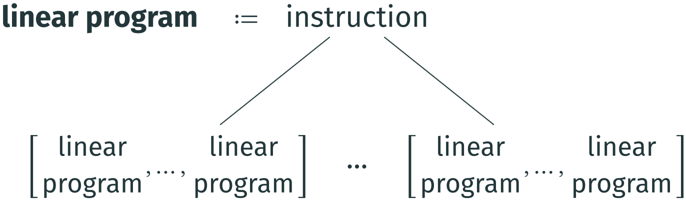
LINEAR APPROXIMATION OF FUNCTIONAL PROGRAMS

Linear programs: each argument of a function is used *exactly once*.



program := instruction

program … program



linear program := instruction

$\left[\begin{array}{c}\text{linear}\\ \text{program}\end{array}, \dots, \begin{array}{c}\text{linear}\\ \text{program}\end{array}\right]$ … $\left[\begin{array}{c}\text{linear}\\ \text{program}\end{array}, \dots, \begin{array}{c}\text{linear}\\ \text{program}\end{array}\right]$

LINEAR APPROXIMATION OF FUNCTIONAL PROGRAMS

LINEAR APPROXIMATION OF THE λ-CALCULUS

λ-terms:

$$M, N, \dots \quad := \quad x \quad | \quad \lambda x.M \quad | \quad MN$$

equipped with β-reduction:

$$\frac{}{(\lambda x.M)N \longrightarrow_{\beta} M[N/x]}$$

(+ lifting to contexts)

LINEAR APPROXIMATION OF THE λ-CALCULUS

λ⊥-terms:

$$M, N, \ldots \quad := \quad x \quad | \quad \lambda x.M \quad | \quad MN \quad | \quad \bot$$

equipped with β-reduction:

$$\frac{}{(\lambda x.M)N \longrightarrow_{\beta} M[N/x]}$$

(+ lifting to contexts)

LINEAR APPROXIMATION OF THE λ-CALCULUS

λ⊥-terms:

$$M, N, \ldots \quad := \quad x \quad | \quad \lambda x.M \quad | \quad MN \quad | \quad \bot$$

equipped with β⊥-reduction:

$$\frac{}{(\lambda x.M)N \longrightarrow_{\beta} M[N/x]} \qquad \frac{M \text{ produces no information}}{M \longrightarrow_{\bot} \bot}$$

(+ lifting to contexts)

Let's go more technical

APPROXIMATIONS OF THE λ-CALCULUS

The **continuous** approximation:

$$\begin{tikzcd}
\text{program} \arrow[r, squiggly, "\text{(infinite) execution}"] \arrow[d, equal] & \text{result} \\
\text{program} \arrow[r, squiggly, "\text{(finite) partial execution}"'] & \text{partial result} \arrow[u, phantom, sloped, "\sqsubseteq"]
\end{tikzcd}$$

The **linear** approximation:

$$\begin{tikzcd}
\text{program} \arrow[r, squiggly, "\text{(infinite) execution}"] & \text{result} \\
\text{linear program} \arrow[u, phantom, sloped, "\sqsubseteq"] \arrow[r, squiggly, "\text{linear execution (finite!)}"'] & \text{linear partial result} \arrow[u, phantom, sloped, "\sqsubseteq"]
\end{tikzcd}$$

THE BÖHM SEMANTICS

What is finite prefix of stable information?
A **head normal form** $\lambda x_1 \ldots \lambda x._m.(y)M_1 \ldots M_n$.

What is the total information a term can output?
Its **Böhm tree**:

$$\mathrm{BT}(M) := \begin{cases} \lambda\vec{x}.(y)\,\mathrm{BT}(M_1)\ \ldots\ \mathrm{BT}(M_n) & \text{if } M \longrightarrow_\beta^* \lambda\vec{x}.(y)M_1 \ldots M_n \\ \bot & \text{otherwise.} \end{cases}$$

This is a coinductive definition: $\mathrm{BT}(Y) = \lambda f.f\,f\,f \ldots$

APPROXIMATIONS OF THE λ-CALCULUS

The **continuous** approximation:

$$\begin{tikzcd}
M \arrow[r, rightsquigarrow, "\text{(infinite) execution}"] \arrow[d, equal] & \mathrm{BT}(M) \\
M \arrow[r, rightsquigarrow, "\text{(finite) partial execution}"'] & \text{partial result} \arrow[u, phantom, sloped, "\sqsubseteq"]
\end{tikzcd}$$

The **linear** approximation:

$$\begin{tikzcd}
M \arrow[r, rightsquigarrow, "\text{(infinite) execution}"] & \mathrm{BT}(M) \\
\text{linear program} \arrow[u, phantom, sloped, "\sqsubseteq"] \arrow[r, rightsquigarrow, "\text{linear execution (finite!)}"'] & \text{linear partial result} \arrow[u, phantom, sloped, "\sqsubseteq"]
\end{tikzcd}$$

AN INFINITARY λ-CALCULUS

- Infinitary terms
  (*via* coinduction, metric completion, ideal completion)

$$\frac{}{x \in \Lambda^{\infty}_{\perp}} \qquad \frac{P \in \Lambda^{\infty}_{\perp}}{\lambda x.P \in \Lambda^{\infty}_{\perp}} \qquad \frac{P \in \Lambda^{\infty}_{\perp} \qquad \underline{Q \in \Lambda^{\infty}_{\perp}}}{PQ \in \Lambda^{\infty}_{\perp}}$$

AN INFINITARY λ-CALCULUS

- Infinitary terms
  (*via* coinduction, metric completion, ideal completion)
- Infinitary reductions
  (*via* coinduction, transfinite sequences of reductions)

$$\frac{M \longrightarrow^{*}_{\beta} x}{M \longrightarrow^{\infty}_{\beta} x} \qquad \frac{M \longrightarrow^{*}_{\beta} \lambda x.P \qquad P \longrightarrow^{\infty}_{\beta} P'}{M \longrightarrow^{\infty}_{\beta} \lambda x.P'} \qquad \frac{M \longrightarrow^{*}_{\beta} PQ \qquad P \longrightarrow^{\infty}_{\beta} P' \qquad \underline{Q \longrightarrow^{\infty}_{\beta} Q'}}{M \longrightarrow^{\infty}_{\beta} P'Q'}$$

AN INFINITARY λ-CALCULUS

- Infinitary terms
  (*via* coinduction, metric completion, ideal completion)
- Infinitary reductions
  (*via* coinduction, transfinite sequences of reductions)

Theorem [Kennaway *et al.* 1997]

$\longrightarrow_{\beta\perp}^{\infty}$ is confluent.

Corollary

BT(M) is the unique β⊥-normal form of M through $\longrightarrow_{\beta\perp}^{\infty}$.

APPROXIMATIONS OF THE λ-CALCULUS

The **continuous** approximation:

$$\begin{tikzcd}
M \arrow[r, "\infty", "\beta\perp"'] \arrow[d, equal] & \mathrm{BT}(M) \arrow[d, phantom, "\rotatebox{90}{$\sqsubseteq$}"] \\
M \arrow[r, squiggly, "\text{(finite) partial execution}"'] & \text{partial result}
\end{tikzcd}$$

The **linear** approximation:

$$\begin{tikzcd}
M \arrow[r, "\infty", "\beta\perp"'] \arrow[d, phantom, "\rotatebox{90}{$\sqsubseteq$}"] & \mathrm{BT}(M) \arrow[d, phantom, "\rotatebox{90}{$\sqsubseteq$}"] \\
\text{linear program} \arrow[r, squiggly, "\text{linear execution (finite!)}"'] & \text{linear partial result}
\end{tikzcd}$$

THE CONTINUOUS APPROXIMATION

The **continuous** approximation:

$$\begin{tikzcd}
M \arrow[rr, "\infty", "\beta\perp"'] \arrow[d, equal] & & \mathrm{BT}(M) \arrow[d, phantom, "\sqsupseteq"] \\
M \arrow[r, "*", "\beta"'] & M' \arrow[r, phantom, "\sqsupseteq"] & P
\end{tikzcd}$$

Continous approximation theorem

- $\mathscr{A}(M) := \left\{ P \text{ in } \beta\perp\text{-nf} \;\middle|\; \exists M',\ M \longrightarrow^*_\beta M' \sqsupseteq P \right\}$ is directed.
- $\bigsqcup \mathscr{A}(M) = \mathrm{BT}(M)$.

THE LINEAR APPROXIMATION

$$\begin{tikzcd}
M \arrow[r, "\infty", "\beta\perp"'] & \mathrm{BT}(M) \\
\text{linear program} \arrow[u, phantom, sloped, "\sqsubseteq"] \arrow[r, squiggly, "\text{linear execution (finite!)}"'] & \text{linear partial result} \arrow[u, phantom, sloped, "\sqsubseteq"]
\end{tikzcd}$$

THE RESOURCE CALCULUS

Linear programs are **resource λ-terms**:

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

A λ-term is **Taylor expanded** into a formal sum of resource approximants:

$$\begin{aligned}
\mathscr{T}(x) &:= x & \{x\} \\
\mathscr{T}(\lambda x.M) &:= \lambda x.\mathscr{T}(M) & \{\, \lambda x.s \mid s \in \mathscr{T}(M) \,\} \\
\mathscr{T}(MN) &:= \mathscr{T}(M) \sum_{n \in \mathbb{N}} \frac{1}{n!} \mathscr{T}(N)^n & \{\, s[t_1, \ldots, t_n] \mid \ldots \,\} \\
\mathscr{T}(\bot) &:= 0 & \emptyset
\end{aligned}$$

… and this also works for infinitary terms (kind of).

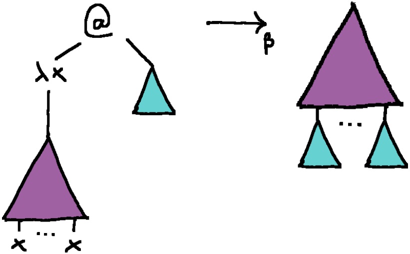
The linear approximation

$$\begin{tikzcd}
M \arrow[r, "\infty", "\beta\perp"'] \arrow[d, phantom, "\wr|"] & \mathrm{BT}(M) \arrow[d, phantom, "\wr|"] \\
\mathscr{T}(M) \arrow[r, squiggly, "\text{linear execution (finite!)}"'] & \mathscr{T}(\mathrm{BT}(M))
\end{tikzcd}$$

LINEAR APPROXIMATION OF THE λ-CALCULUS

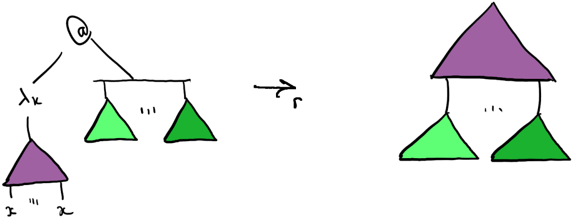
Program execution is the **β-reduction** on λ-terms:

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



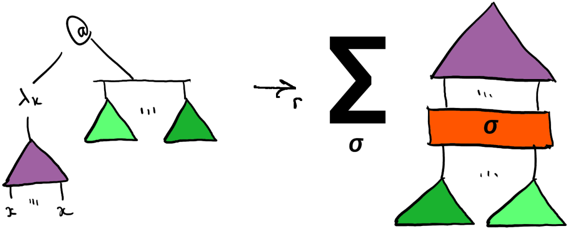
LINEAR APPROXIMATION OF THE λ-CALCULUS

Linear execution is the **resource reduction** on resource λ-terms:



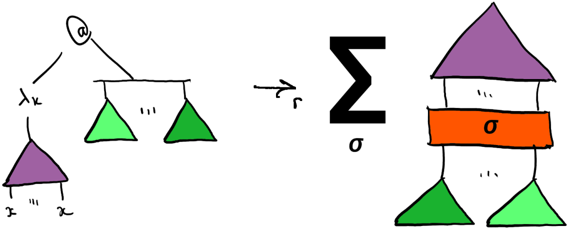
LINEAR APPROXIMATION OF THE λ-CALCULUS

Linear execution is the **resource reduction** on resource λ-terms:



LINEAR APPROXIMATION OF THE λ-CALCULUS

Linear execution is the **resource reduction** on resource λ-terms:



It is confluent and strongly normalising!

THE LINEAR APPROXIMATION

$$\begin{tikzcd}[column sep=huge]
M \arrow[r, "\infty", "\beta\perp"'] \arrow[d, phantom, sloped, "\simeq"] & \mathrm{BT}(M) \arrow[d, phantom, sloped, "\simeq"] \\
\mathscr{T}(M) \arrow[r, two heads, "\mathsf{r}"'] & \mathscr{T}(\mathrm{BT}(M))
\end{tikzcd}$$

THE LINEAR APPROXIMATION

$$\begin{tikzcd}
M \arrow[rr, "\infty", "\beta\perp"'] \arrow[d, phantom, sloped, "\simeq"] & & N \arrow[d, phantom, sloped, "\simeq"] \\
\mathscr{T}(M) \arrow[rr, twoheadrightarrow, "\mathsf{r}"'] & & \mathscr{T}(N)
\end{tikzcd}$$

Theorem (simulation) [C. and Vaux 2023, C. 2024]

If $M \longrightarrow^{\infty}_{\beta\perp} N$ then $\mathscr{T}(M) \twoheadrightarrow_{\mathsf{r}} \mathscr{T}(N)$.

Corollary (commutation)

$\mathrm{nf}(\mathscr{T}(M)) = \mathscr{T}(\mathrm{BT}(M))$.

WHY SHOULD WE CARE

EASIER PROOFS, IN A UNIFIED SETTING

Corollaries (seen before)

Confluence of $\longrightarrow_{\beta\bot}^{\infty}$

Continuous approximation theorem

EASIER PROOFS, IN A UNIFIED SETTING

Corollaries (seen before)

Confluence of $\longrightarrow_{\beta\bot}^{\infty}$

Continuous approximation theorem

Corollary

M has a HNF through $\longrightarrow_{\beta}^{*}$ or $\longrightarrow_{\beta}^{\infty}$

iff the head reduction strategy terminates on M

iff $\mathrm{nf}(\mathscr{T}(M)) \neq 0$

iff M is typable with non-idempotent intersection types:

$$\frac{}{\Gamma, x : [a] \vdash x : a} \qquad \frac{\Gamma, x : [a_1, \dots, a_n] \vdash M : b}{\Gamma \vdash \lambda x.M : [a_1, \dots, a_n] \to b}$$

$$\frac{\Gamma \vdash P : [a_1, \dots, a_n] \to b \qquad \Gamma' \vdash Q : [a_1, \dots, a_n]}{\Gamma + \Gamma' \vdash PQ : b} \qquad \frac{\Gamma_1 \vdash Q : a_1 \qquad \dots \qquad \Gamma_n \vdash Q : a_n}{\sum_{i=1}^{n} \Gamma_i \vdash Q : [a_1, \dots, a_n]}$$

EASIER PROOFS, IN A UNIFIED SETTING

Some more:

Corollary [Barbarossa-Manzonetto, 2020]

$\{ M = N \mid \mathrm{BT}(M) = \mathrm{BT}(N) \}$ is a λ-theory.

Corollary (Genericity lemma)

If M has no HNF and $C[M]$ has a normal form C' then forall N, $C[N] \longrightarrow_{\beta}^{*} C'$.

Corollary

$\mathrm{BT} : \Lambda_{\perp}^{\infty} \to \Lambda_{\perp}^{\infty}$ is Scott-continuous.

A GENERAL METHOD

Linear approximation and **infinitary rewriting** are useful tools allowing to **"speak of semantics dynamically"**,

and they can be exported:

- to the lazy λ-calculus (easy) [Cerda'24]
- to $\beta\eta$-normalisation (feasible)
- to probabilistic λ-calculi (hmm...)
- and?

SOME SATELLITE QUESTIONS

SOME SATELLITE QUESTIONS

- The converse of simulation, **conservativity**:
  if $\mathscr{T}(M) \twoheadrightarrow_{\mathsf{r}} \mathscr{T}(N)$, is it true that $M \longrightarrow_{\beta}^{\infty} N$?
- How to design a program approximation suited to the **λI-calculus** (the λ-calculus without erasure)?
- How does the **coinductive infinitary rewriting** work?

CONSERVATIVITY OF THE LINEAR APPROXIMATION

Fact 1 (simulation, finite)

Let M, N be finite λ-terms.

If $M \longrightarrow_{\beta}^{*} N$ then $\mathscr{T}(M) \twoheadrightarrow_{\mathrm{r}} \mathscr{T}(N)$.

Problem 1 (conservativity, finite)

Is the converse true?

Theorem 1 [C. and Vaux 2025]

Yes it is! If $\mathscr{T}(M) \twoheadrightarrow_{\mathrm{r}} \mathscr{T}(N)$ then $M \longrightarrow_{\beta}^{*} N$.

CONSERVATIVITY OF THE LINEAR APPROXIMATION

Fact 2 (simulation, infinitary)

Let M, N be infinitary λ-terms.
If $M \longrightarrow_{\beta}^{\infty} N$ then $\mathscr{T}(M) \longrightarrow\!\!\!\!\rightarrow_{\mathsf{r}} \mathscr{T}(N)$.

Problem 2 (conservativity, infinitary)

Is the converse true?

Theorem 2 [C. and Vaux 2025]

No, it isn't!
There are terms $\mathsf{A}, \bar{\mathsf{A}}$ such that $\mathscr{T}(\mathsf{A}) \longrightarrow\!\!\!\!\rightarrow_{\mathsf{r}} \mathscr{T}(\bar{\mathsf{A}})$ but there is no reduction $\mathsf{A} \longrightarrow_{\beta}^{\infty} \bar{\mathsf{A}}$.

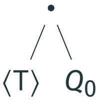
A is the *Accordion* λ-term.

The Accordion

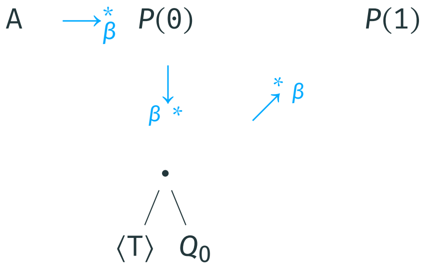
$$\mathtt{A} \quad \longrightarrow_{\beta}^{*} \quad P(\mathtt{0})$$

THE ACCORDION

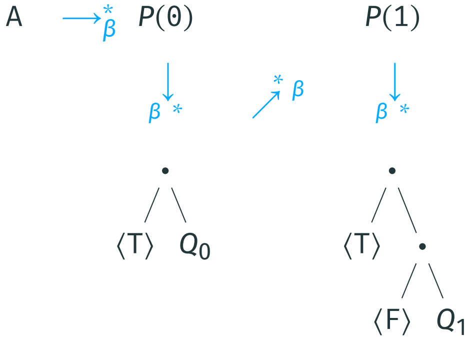
$$\begin{tikzcd}
A \arrow[r, "*", "\beta"'] & P(0) \arrow[d, "\beta\ *"'] \\
& {}
\end{tikzcd}$$



THE ACCORDION



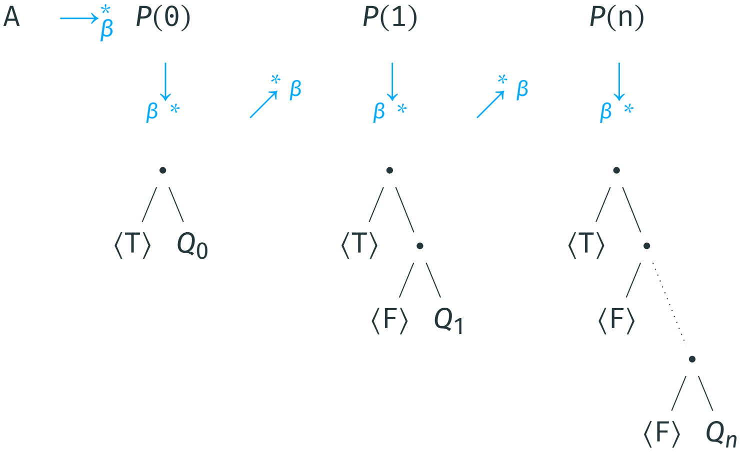
THE ACCORDION



$A \xrightarrow{*}_{\beta} P(0)$ $\quad P(1)$

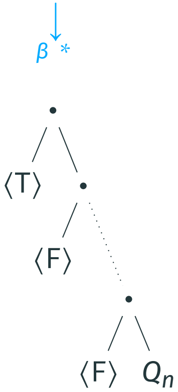
$\downarrow \beta^{*} \qquad \nearrow^{*}\beta \qquad \downarrow \beta^{*}$

THE ACCORDION



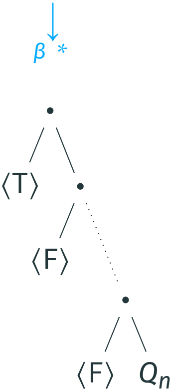
The Accordion

$$A \quad \overset{*}{\longrightarrow_{\beta}} \quad P(\mathrm{n})$$

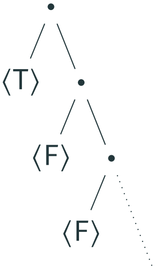


THE ACCORDION

$A \longrightarrow_{\beta}^{*} P(n)$



$\bar{A} =$



CONSERVATIVITY OF THE LINEAR APPROXIMATION

Problem 3 (restoring conservativity)

Can we restrict $\twoheadrightarrow_{\mathrm{r}}$ to obtain a conservative approximation?

Theorem 3 [C. and Vaux 2025]

Yes, thanks to the *uniform* lifting of the resource reduction $\overset{\frown}{\longrightarrow}_{\mathrm{r}}^{\infty}$!

If $\mathscr{T}(M) \overset{\frown}{\longrightarrow}_{\mathrm{r}}^{\infty} \mathscr{T}(N)$ then $M \longrightarrow_{\beta}^{\infty} N$.

In particular, there is no reduction $\mathscr{T}(\mathrm{A}) \overset{\frown}{\longrightarrow}_{\mathrm{r}}^{\infty} \mathscr{T}(\bar{\mathrm{A}})$.

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 ⇝ 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...

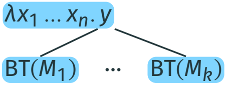
λ

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



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

For example: $\mathrm{BT}(\mathrm{I}) := \mathrm{I}$ $\mathrm{BT}(\Omega) := \bot$ $\mathrm{BT}(\mathrm{Y}) := \lambda f . f(f(f(\dots)))$

FROM λ-THEORIES... TO λI-THEORIES

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

λI

FROM λ-THEORIES… TO λI-THEORIES

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, $\Lambda_{\mathsf{I}} := \bigcup_{X \subseteq_{\mathsf{f}} \mathcal{V}} \Lambda_{\mathsf{I}}(X)$, 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	⇝	A λ-theory	⇜	A notion of evaluation tree
λI			???		

FROM λ-THEORIES… TO λI-THEORIES

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

λI **???**

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

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

- 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	$\leftsquigarrow$	**???**

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	$\leftsquigarrow$	**???**

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

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

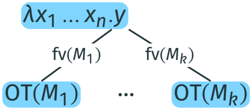
Indeed, abstracted variables may be:

- **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 $\mathrm{BT}(M) = \lambda xf.f(f(f(\dots)))$.

INTRODUCING OHANA TREES

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

$\mathrm{OT}(M) :=$



$\lambda x_1 \ldots x_n.y$

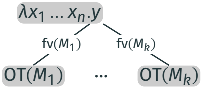
$\mathrm{fv}(M_1)$ $\mathrm{fv}(M_k)$

$\mathrm{OT}(M_1)$ $\cdots$ $\mathrm{OT}(M_k)$

if $M \twoheadrightarrow_h \lambda x_1 \ldots x_n.yM_1 \cdots M_k$,

$\perp_{\mathrm{fv}(M)}$ otherwise.

NO VARIABLE LEFT BEHIND!



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

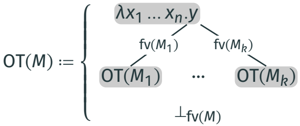
Whereas Böhm trees equate $\lambda x.x(\Omega y)(\Omega z)$ and $\lambda x.x(\Omega z)(\Omega y)$:

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

Ohana trees do separate them:

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

No variable forgotten!



if $M \twoheadrightarrow_h \lambda x_1 \dots x_n.yM_1 \cdots M_k$,

otherwise.

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

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

Ohana trees do separate them: $\mathrm{OT}(\mathrm{Y}) \;=$ $\neq$ $=\; \mathrm{OT}(\text{🕮}_l)$.