Intersection types for productivity and for non-erasure

Rémy Cerda, Università di Bologna

Paris, IRIF, 26 March 2026

Au menu aujourd'hui...

Tropical grades for productive programs

- A polymorphic recursive calculus of productive programs
- Tropical intersection types for productivity in the λ-calculus

Ohana trees and intersection types for the λI-calculus

- λI-theories, Ohana trees, and the induced theory $\mathcal{O}$
- An intersection type semantics characterising the theory $\mathcal{O}$

Tropical grades for productive programs

Amount of accesses to resources

The meaning of

$$x : !_a A \vdash M : B$$

is “using a copies of x...”

[Girard, Scedrov, Scott ’92; Brunel *et al.* ’14; Ghica, Smith ’14]

Amount of accesses to resources

The meaning of

$$x : !_a A \vdash M : B$$

is "using a copies of x..."

Grades compose accordingly in cuts:

$$\frac{x : !_a A \vdash M : !_b B \multimap C \qquad x : !_{a'} A \vdash M : B}{x : !_{a+b\times a'} A \vdash MN : C}$$

[Girard, Scedrov, Scott '92; Brunel *et al.* '14; Ghica, Smith '14]

Amount of accesses to resources

The meaning of

$$x : !_a A \vdash M : B$$

is “using a copies of x...”

Grades compose accordingly in cuts:

$$\frac{x : !_a A \vdash M : !_b B \multimap C \qquad x : !_{a'} A \vdash M : B}{x : !_{a+b\times a'} A \vdash MN : C}$$

Delay in access to resources

The meaning of

$$x : !_a A \vdash M : B$$

is “if x is available after a ticks...”
or “if I see x under a guards...”

[Girard, Scedrov, Scott ’92; Brunel *et al.* ’14; Ghica, Smith ’14]

Amount of accesses to resources

The meaning of

$$x : !_a A \vdash M : B$$

is "using a copies of x..."

Grades compose accordingly in cuts:

$$\frac{x : !_a A \vdash M : !_b B \multimap C \qquad x : !_{a'} A \vdash M : B}{x : !_{a+b\times a'} A \vdash MN : C}$$

Delay in access to resources

The meaning of

$$x : !_a A \vdash M : B$$

is "if x is available after a ticks..."
or "if I see x under a guards..."

Grades compose accordingly in cuts:

$$\frac{x : !_a A \vdash M : !_b B \multimap C \qquad x : !_{a'} A \vdash M : B}{x : !_{\min(a,b+a')} A \vdash MN : C}$$

[Girard, Scedrov, Scott '92; Brunel *et al.* '14; Ghica, Smith '14]

A BOUNDED/GRADED SYSTEM F WITH RECURSION

Let $(\mathbf{S}, +, 0, \times, 1, \leqslant, \top)$ be an ordered semiring s.t. 0 and $\top$ are the least and greatest elements.

Types are defined by:

$$\mathrm{B}_{\mathbf{S}}\mathrm{F}\mu \quad \ni \quad A, B, \ldots \quad ::= \quad \alpha \mid !_a A \multimap B \mid \forall \alpha.A \mid \mu \alpha.A \qquad (\alpha \in \mathcal{A}, a \in \mathbf{S}).$$

Terms are defined by:

$$\Lambda_{\mathrm{F}\mu} \quad \ni \quad M, N, \ldots \quad ::= \quad x \mid \lambda x^A.M \mid MN \mid \Lambda \alpha.M \mid MA \mid \mathsf{fold}\, M \mid \mathsf{unfold}\, M$$
$$(x \in \mathcal{V}, \alpha \in \mathcal{A}).$$

Typing rules are as follows:

$$\frac{}{x : !_1 A \vdash x : A}\ (\mathrm{ax})$$

$$\frac{\Gamma, x : !_a A \vdash M : B}{\Gamma \vdash \lambda x^A.M : !_a A \multimap B}\ (\multimap_{\mathrm{i}}) \qquad \frac{\Gamma \vdash M : !_a A \multimap B \qquad \Gamma' \vdash N : A}{\Gamma + (\Gamma' \times a) \vdash MN : B}\ (\multimap_{\mathrm{e}})$$

$$\frac{\Gamma \vdash M : B \qquad \alpha \notin \mathrm{fv}(\Gamma)}{\Gamma \vdash \Lambda\alpha.M : \forall\alpha.B}\ (\forall_{\mathrm{i}}) \qquad \frac{\Gamma \vdash M : \forall\alpha.B}{\Gamma \vdash MA : B[A/\alpha]}\ (\forall_{\mathrm{e}})$$

$$\frac{\Gamma \vdash M : A[\mu\alpha.A/\alpha]}{\Gamma \vdash \mathrm{fold}\, M : \mu\alpha.A}\ (\mathrm{fold}) \qquad \frac{\Gamma \vdash M : \mu\alpha.A}{\Gamma \vdash \mathrm{unfold}\, M : A[\mu\alpha.A/\alpha]}\ (\mathrm{unfold})$$

Typing rules are as follows:

$$\frac{}{x : {!_0}A \vdash x : A}\ (\mathrm{ax})$$

$$\frac{\Gamma, x : {!_a}A \vdash M : B}{\Gamma \vdash \lambda x^A.M : {!_a}A \multimap B}\ (\multimap_{\mathrm{i}}) \qquad \frac{\Gamma \vdash M : {!_a}A \multimap B \qquad \Gamma' \vdash N : A}{\min(\Gamma, \Gamma' + a) \vdash MN : B}\ (\multimap_{\mathrm{e}})$$

$$\frac{\Gamma \vdash M : B \qquad \alpha \notin \mathrm{fv}(\Gamma)}{\Gamma \vdash \Lambda\alpha.M : \forall\alpha.B}\ (\forall_{\mathrm{i}}) \qquad \frac{\Gamma \vdash M : \forall\alpha.B}{\Gamma \vdash MA : B[A/\alpha]}\ (\forall_{\mathrm{e}})$$

$$\frac{\Gamma \vdash M : A[\mu\alpha.A/\alpha]}{\Gamma \vdash \mathrm{fold}\, M : \mu\alpha.A}\ (\mathrm{fold}) \qquad \frac{\Gamma \vdash M : \mu\alpha.A}{\Gamma \vdash \mathrm{unfold}\, M : A[\mu\alpha.A/\alpha]}\ (\mathrm{unfold})$$

where from now on $\mathbf{S}$ is the tropical semiring $\mathbf{T} \coloneqq (\mathbf{R}_+^\infty, \min, \infty, +, 0, \geqslant, 0)$.

Bad things happen

With $A \coloneqq \mu\beta.!_0\beta \multimap \alpha$, we can derive: ($\underline{M} \coloneqq \mathsf{fold}\, M$, $\overline{M} \coloneqq \mathsf{unfold}\, M$)

$$\dfrac{\dfrac{\dfrac{}{x : !_0 A \vdash x : A}}{x : !_0 A \vdash \overline{x} : !_0 A \multimap \alpha} \quad \dfrac{}{x : !_0 A \vdash x : A}}{\dfrac{x : !_0 A \vdash \overline{x}x : \alpha}{\vdash \lambda x^A.\overline{x}x : !_0 A \multimap \alpha}} \quad \dfrac{\vdots}{\dfrac{\vdash \lambda x^A.\overline{x}x : !_0 A \multimap \alpha}{\vdash \underline{\lambda x^A.\overline{x}x} : A}}$$

$$\vdash \Omega \coloneqq (\lambda x^A.\overline{x}x)(\underline{\lambda x^A.\overline{x}x}) : \alpha$$

Bad things happen

With $A \coloneqq \mu\beta.!_0\beta \multimap \alpha$, we can derive: ($\underline{M} \coloneqq \text{fold}\, M$, $\overline{M} \coloneqq \text{unfold}\, M$)

$$\dfrac{\dfrac{\dfrac{}{x : !_0 A \vdash x : A}}{x : !_0 A \vdash \overline{x} : !_0 A \multimap \alpha} \quad \dfrac{}{x : !_0 A \vdash x : A}}{\dfrac{x : !_0 A \vdash \overline{x}x : \alpha}{\vdash \lambda x^A.\overline{x}x : !_0 A \multimap \alpha}} \quad \dfrac{\vdots}{\dfrac{\vdash \lambda x^A.\overline{x}x : !_0 A \multimap \alpha}{\vdash \underline{\lambda x^A.\overline{x}x} : A}}$$

$$\vdash \Omega \coloneqq (\lambda x^A.\overline{x}x)(\underline{\lambda x^A.\overline{x}x}) : \alpha$$

As we want to type only **productive** terms, we introduce the following restriction: Type $\mu\alpha.A$ is defined only if α occurs in A only under some $!_a$, for $a \neq 0$.

With $A \coloneqq \mu\beta.!_1\beta \multimap \alpha$ and $F \coloneqq !_1\alpha \multimap \alpha$, we can derive:

$$\dfrac{\dfrac{\dfrac{}{f : !_0F \vdash f : F} \qquad \dfrac{\dfrac{\dfrac{}{x : !_0A \vdash x : A}}{x : !_0A \vdash \overline{x} : !_1A \multimap \alpha} \qquad \dfrac{}{x : !_0A \vdash x : A}}{x : !_0A \vdash \overline{x}x : \alpha}}{f : !_0F, x : !_1A \vdash f(\overline{x}x) : \alpha}}{f : !_0F \vdash \lambda x^A.f(\overline{x}x) : !_1A \multimap \alpha} \qquad \dfrac{\vdots}{f : !_0F \vdash \lambda x^A.f(\overline{x}x) : !_1A \multimap \alpha}}{f : !_0F \vdash \underline{\lambda x^A.f(\overline{x}x)} : A}$$

$$\dfrac{f : !_0F \vdash (\lambda x^A.f(\overline{x}x))(\underline{\lambda x^A.f(\overline{x}x)}) : \alpha}{\vdash \mathsf{Y} : !_0(!_1\alpha \multimap \alpha) \multimap \alpha}$$

The typing enforces that f is a guard! *E.g.* we can't substitute it by $\lambda x.x$...

Thanks to the Scott encoding we have...

Booleans:	$\forall\alpha.!_0\alpha \multimap !_0\alpha \multimap \alpha$
Numerals:	$\mu\alpha.\forall\beta.!_0\beta \multimap !_0(!_1\alpha \to \beta) \to \beta$
Streams:	$\forall\alpha.\mu\beta.\forall\gamma.!_0(!_0\alpha \multimap !_1\beta \multimap \gamma) \multimap \gamma$
...	

and we can program *e.g.* on streams:

$$Y\left(\lambda f^{!_0\mathrm{Str}_A\multimap\mathrm{Str}_A}.\lambda s^{\mathrm{Str}_A}.\Lambda\gamma.\lambda c^{!_0A\multimap!_1\mathrm{Str}_A\to\gamma}.c\left(M(\overline{s}At)(\overline{\overline{s}Af}At)\right)(\overline{\overline{f}sAf}Af)\right) : !_0\mathrm{Str}_A \multimap \mathrm{Str}_A$$

takes the first two elements of a stream s, applies a map $M : !_0A \multimap !_0A \multimap A$ and outputs the result, and goes on with the remainder of s.

Question 1: What exactly is the expressivity of the system?

Only good things happen

A term $M \in \Lambda_{F\mu}$ is *productive* whenever it is head normalising and the direct subterms of its HNF are productive, *i.e.*

- $M \twoheadrightarrow_\beta x$,
- $M \twoheadrightarrow_\beta \lambda x^A.P$ and P is a productive HNF,
- $M \twoheadrightarrow_\beta PQ$, P is a productive HNF, $P \neq \lambda x^A.P'$, and Q is productive,
- $M \twoheadrightarrow_\beta \Lambda\alpha.P$ and P is a productive HNF,
- $M \twoheadrightarrow_\beta PA$, P is a productive HNF and $P \neq \Lambda\alpha.P'$.
- $M \twoheadrightarrow_\beta \mathsf{fold}\, P$ and P is a productive HNF,
- $M \twoheadrightarrow_\beta \mathsf{unfold}\, P$, P is a productive HNF and $P \neq \mathsf{fold}\, P'$.

Only good things happen

A term $M \in \Lambda_{F\mu}$ is *productive* whenever it is head normalising and the direct subterms of its HNF are productive.

Theorem (soundness)

If M is typable then it is productive.

A term $M \in \Lambda_{F\mu}$ is *productive* whenever it is head normalising and the direct subterms of its HNF are productive.

Theorem (soundness)

If M is typable then it is productive.

The (coinductive step of the) proof is by reducibility candidates:

$$
\begin{aligned}
[\![\alpha]\!]_\rho &:= \rho(\alpha), \\
[\![!_0 A \multimap B]\!]_\rho &:= [\![A]\!]_\rho \to [\![B]\!]_\rho, \\
[\![!_a A \multimap B]\!]_\rho &:= \Lambda \to [\![B]\!]_\rho, && \text{for } a \neq 0, \\
[\![\forall \alpha.A]\!]_\rho &:= \bigcap_{\mathcal{R} \text{ a RC}} [\![A]\!]_{\rho[\alpha \mapsto \mathcal{R}]}, \\
[\![\mu \alpha.A]\!]_\rho &:= [\![A[\mu\alpha.A/\alpha]]\!]_\rho.
\end{aligned}
$$

A glimpse at the main lemma:

If $x_1 : !_{a_1} A_1, \dots, x_n : !_{a_n} A_n \vdash M : B$, *$\rho$ is an environment, and*

$$\text{for all } i \in [1,t],\ T_i \in [\![A_i]\!]_\rho^{(a_i)} := \begin{cases} [\![A_i]\!]_\rho & \text{if } a_i = 0, \\ \Lambda & \text{otherwise,} \end{cases}$$

then $|M|[\vec{T}/\vec{x}] := |M|[T_1/x_1] \cdots [T_n/x_n] \in [\![B]\!]_\rho$.

In the key case of the proof:

$$\frac{x : !_a A \vdash M : !_b B \multimap C \qquad x : !_{a'} A \vdash M : B}{x : !_{\min(a, b+a')} A \vdash MN : C}$$

the tropical operations yield exactly the induction hypotheses we need!

Only good things happen

A term $M \in \Lambda_{F\mu}$ is *productive* whenever it is head normalising and the direct subterms of its HNF are productive.

Theorem (soundness)

If M is typable then it is productive.

Question 2: Are there complete fragments?

The meaning of grades (continued)

Amount of accesses to resources

The meaning of

$$x : !_a A \vdash M : B$$

is “using a copies of x...”

Delay in access to resources

The meaning of

$$x : !_a A \vdash M : B$$

is “if x is available after a ticks...”
or “if I see x under a guards...”

The meaning of grades (continued)

Amount of accesses to resources

The meaning of

$$x : !_a A \vdash M : B$$

is "using a copies of x..."

If each copy bears a different type, we obtain intersection types:

$$x : [\sigma_1, \dots, \sigma_a] \vdash M : \tau$$

Delay in access to resources

The meaning of

$$x : !_a A \vdash M : B$$

is "if x is available after a ticks..." or "if I see x under a guards..."

Can we do the same?

The meaning of grades (continued)

Amount of accesses to resources

The meaning of

$$x : !_a A \vdash M : B$$

is "using a copies of x..."

If each copy bears a different type, we obtain intersection types:

$$x : \left[\begin{array}{ccc} \text{types} & \to & \mathbf{N} \\ \sigma_i & \mapsto & a_i \end{array}\right] \vdash M : \tau$$

Delay in access to resources

The meaning of

$$x : !_a A \vdash M : B$$

is "if x is available after a ticks..." or "if I see x under a guards..."

Can we do the same?

The meaning of grades (continued)

Amount of accesses to resources

The meaning of

$$x : !_a A \vdash M : B$$

is "using a copies of x..."

If each copy bears a different type, we obtain intersection types:

$$x : \begin{bmatrix} \text{types} & \to & \mathbf{N} \\ \sigma_i & \mapsto & a_i \end{bmatrix} \vdash M : \tau$$

Delay in access to resources

The meaning of

$$x : !_a A \vdash M : B$$

is "if x is available after a ticks..."
or "if I see x under a guards..."

Can we do the same?
Let's try:

$$x : \begin{bmatrix} \text{types} & \to & \mathbf{T} \\ \sigma_i & \mapsto & a_i \end{bmatrix} \vdash M : \tau$$

Types are defined as follows:

$$\frac{\alpha \in \mathcal{A}}{\alpha \in \mathsf{I}_{\mathbf{S}}} \qquad \frac{\bar{\sigma} \in \mathsf{I}^{!}_{\mathbf{S}} \qquad \tau \in \mathsf{I}_{\mathbf{S}}}{\bar{\sigma} \multimap \tau \in \mathsf{I}_{\mathbf{S}}} \qquad \frac{(\sigma_i \in \mathsf{I}_{\mathbf{S}})_{i<\omega} \qquad \lim_{i\to\omega} a_i = \infty}{[\sigma_i^{a_i}]_{i<\omega} \in \mathsf{I}^{!}_{\mathbf{S}}}$$

where $[\sigma_i^{a_i}]_{i<\omega}$ is a notation for $\sigma_i \mapsto a_i$.

Types are defined as follows:

$$\frac{\alpha \in \mathcal{A}}{\alpha \in \mathsf{I}_{\mathbf{S}}} \qquad \frac{\bar{\sigma} \in \mathsf{I}_{\mathbf{S}}^{!} \qquad \tau \in \mathsf{I}_{\mathbf{S}}}{\bar{\sigma} \multimap \tau \in \mathsf{I}_{\mathbf{S}}} \qquad \frac{(\sigma_i \in \mathsf{I}_{\mathbf{S}})_{i<\omega} \qquad \lim_{i\to\omega} a_i = \infty}{[\sigma_i^{a_i}]_{i<\omega} \in \mathsf{I}_{\mathbf{S}}^{!}}$$

where $[\sigma_i^{a_i}]_{i<\omega}$ is a notation for $\sigma_i \mapsto a_i$.

Typing rules are as follows:

$$\frac{}{\Gamma, x : [\sigma^0] \vdash x : \sigma}\ (\mathrm{ax}) \qquad \frac{\Gamma, x : \bar{\sigma} \vdash M : \tau}{\Gamma \vdash \lambda x.M : \bar{\sigma} \multimap \tau}\ (\multimap_{\mathrm{i}}) \qquad \frac{\Gamma \vdash M : \bar{\sigma} \multimap \tau \qquad \Gamma' \vdash N : \bar{\sigma}}{\min(\Gamma, \Gamma') \vdash MN : \tau}\ (\multimap_{\mathrm{e}})$$

$$\frac{\left(\Gamma_i \vdash^{(a_i)} N : \sigma_i\right)_{i<\omega}}{\min_{i<\omega}(a_i + \Gamma_i) \vdash N : [\sigma_i^{a_i}]_{i<\omega}}\ (!) \qquad \frac{\Gamma \vdash N : \sigma \qquad a \neq \infty}{\Gamma \vdash^{(a)} N : \sigma}\ (\mathrm{c}) \qquad \left(\frac{}{\Gamma \vdash^{(\infty)} N : \sigma}\ (\mathrm{w})\right)$$

But this system doesn't tell us anything we didn't already know...

- M is typable with (w) iff. it is head-normalising.
- M is typable without (w) iff. it is normalising.

But this system doesn't tell us anything we didn't already know...

- M is typable with (w) iff. it is head-normalising.
- M is typable without (w) iff. it is normalising.

Actually types are defined as follows, **coinductively**:

$$\frac{\alpha \in \mathcal{A}}{\alpha \in \mathsf{I}_{\mathbf{S}}} \qquad \frac{\bar{\sigma} \in \mathsf{I}^{!}_{\mathbf{S}} \qquad \tau \in \mathsf{I}_{\mathbf{S}}}{\bar{\sigma} \multimap \tau \in \mathsf{I}_{\mathbf{S}}} \qquad \frac{(\triangleright_{a_i}\, \sigma_i \in \mathsf{I}_{\mathbf{S}})_{i<\omega} \qquad \lim_{i\to\omega} a_i = \infty}{[\sigma_i^{a_i}]_{i<\omega} \in \mathsf{I}^{!}_{\mathbf{S}}}$$

$$\frac{\sigma \in \mathsf{I}_{\mathbf{S}} \qquad a \neq 0}{\triangleright_a\, \sigma \in \mathsf{I}_{\mathbf{S}}} \qquad \frac{\sigma \in \mathsf{I}_{\mathbf{S}}}{\triangleright_0\, \sigma \in \mathsf{I}_{\mathbf{S}}}$$

Remark 3: Recursion on types should be enough...

Good things happen again!

With $\bar{\sigma} \coloneqq [[*^1] \multimap *^0]$, $\bar{\tau} \coloneqq [\bar{\tau}' \multimap *^0]$ and $\bar{\tau}' \coloneqq [[[\dots] \multimap *^1] \multimap *^1]$ we can derive:

$$\dfrac{\dfrac{\dfrac{}{f : \bar{\sigma} \vdash f : [*^1] \multimap *} \quad \dfrac{\dfrac{\dfrac{}{x : \bar{\tau} \vdash x : \bar{\tau}' \multimap *} \quad \dfrac{}{x : \bar{\tau}' \vdash x : \bar{\tau}'}}{x : \bar{\tau} \vdash xx : *}}{x : \bar{\tau}' \vdash xx : [*^1]}}{f : \bar{\sigma}, x : \bar{\tau}' \vdash f(xx) : *}}{f : \bar{\sigma} \vdash \lambda x.f(xx) : \bar{\tau}' \multimap *} \quad \dfrac{\vdots}{f : \bar{\sigma} + 1 \vdash \lambda x.f(xx) : \bar{\tau}'}}{f : \bar{\sigma} \vdash (\lambda x.f(xx))(\lambda x.f(xx)) : *}}{\vdash \mathsf{Y} : [[*^1] \multimap *^0] \multimap *}$$

Theorem (as usual...). SR and SE hold.

Theorem (soundness). If M is typable, then it is productive.

Thanks to the same kind of reducibility candidates:

$$[\![[\sigma_i^{a_i}] \multimap \tau]\!]_\rho := \left(\bigcap_{\substack{i<\omega \\ a_i=0}} [\![\sigma_i]\!]_\rho \right) \to [\![\tau]\!]_\rho.$$

Conjecture (completeness). If M is productive, then it is typable.

We're working on it...

Question 2': can the system be encoded as a complete fragment of our system F?

Questions

- Are there similar uses of tropical semirings?
- Is this just an instance of [insert your favourite very complicated categorical framework]?
- Does this give rise to interesting “filter models”? “relational” semantics?
- What does the associated Taylor expansion look like?

Ohana trees and intersection types for the λI-calculus

From λ-theories...

λ

A model of the λ-calculus $\llbracket - \rrbracket$

$\rightsquigarrow$

A λ-theory

$M =_{\mathcal{T}} N$ iff $\llbracket M \rrbracket = \llbracket N \rrbracket$

λ

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

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

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

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

From λ-theories...

λ

A model of the λ-calculus ⇝ A λ-theory ⇜ 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

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

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

From λ-theories… to λI-theories

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

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

λI

The **λI-calculus** is the fragment of the λ-calculus **without erasure**. Formally, $\Lambda_{\mathsf{I}} \coloneqq \bigcup_{X \subseteq_{\mathrm{f}} \mathcal{V}} \Lambda_{\mathsf{I}}(X)$, 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: $\lambda x.x \in \Lambda_{\mathsf{I}}(\emptyset)$ $\lambda x.xy \in \Lambda_{\mathsf{I}}(\{y\})$ $\lambda x.y \notin \Lambda_{\mathsf{I}}$

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

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

- 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	⇝	A λ-theory	⇜ A notion of evaluation tree
λI			A λI-theory	⇜ ???

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	⇜ ???

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

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

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

$$\mathrm{OT}(M) := \begin{cases} \lambda x_1 \dots x_n.y \;\text{ with children } \mathrm{fv}(M_1), \dots, \mathrm{fv}(M_k) \text{ over } \mathrm{OT}(M_1), \dots, \mathrm{OT}(M_k) & \text{if } M \twoheadrightarrow_h \lambda x_1 \dots x_n.yM_1 \cdots M_k, \\ \bot_{\mathrm{fv}(M)} & \text{otherwise.} \end{cases}$$

No variable left behind!

$$\mathrm{OT}(M) \coloneqq \begin{cases} \lambda x_1 \ldots x_n.y \;\; \text{(children: } \mathrm{fv}(M_1), \ldots, \mathrm{fv}(M_k) \text{; } \mathrm{OT}(M_1), \ldots, \mathrm{OT}(M_k)) & \text{if } M \twoheadrightarrow_h \lambda x_1 \ldots x_n.yM_1 \cdots M_k, \\ \bot_{\mathrm{fv}(M)} & \text{otherwise.} \end{cases}$$

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:

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

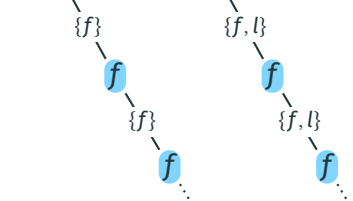
No variable forgotten!

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

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

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

Ohana trees do separate them: $\mathrm{OT}(\mathrm{Y}) = \lambda f.f \neq \lambda f.f = \mathrm{OT}(\boxplus_l)$.



The equational theory $\mathcal{O}$ is defined by saying that for all $M, N \in \Lambda_I$, $M =_{\mathcal{O}} N$ whenever $\mathrm{OT}(M) = \mathrm{OT}(N)$.

We want to prove the following

Theorem

$\mathcal{O}$ is a λI-theory.

The continuous approximation, as usual...

Add constants $\bot_X$ (for $X \subseteq_{\mathrm{f}} \mathcal{V}$) to the syntax.
$\bot_X$ is "an undefined term whose set of free variables is X".

Define a **(head) approximation ordering** by

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

Define **(head) approximants**:

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

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. $\mathrm{OT}(M) = \bigsqcup \mathrm{App}_{\mathrm{m}}(M)$.

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 the multilinear substitution looks like:

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

Also the Taylor expansion is "fibered": $\mathcal{T}_{\mathsf{m}}(M) := (\mathrm{fv}(M), \mathcal{T}^{\bullet}_{\mathsf{m}}(M))$,
where $\mathcal{T}^{\bullet}_{\mathsf{m}}(M)$ is the set of all resource approximants of M.

Linear approximation theorem (aka Commutation). $\mathrm{nf}(\mathcal{T}_{\mathsf{m}}(M)) = \mathcal{T}_{\mathsf{m}}(\mathrm{OT}(M))$.

Theorem

$\mathcal{O}$ is a λI-theory.

Proof sektch. We use the fact that:

$$\begin{aligned} M =_{\mathcal{O}} N &\Leftrightarrow \mathrm{OT}(M) = \mathrm{OT}(N) \\ &\Leftrightarrow \mathcal{T}_{\mathrm{m}}(\mathrm{OT}(M)) = \mathcal{T}_{\mathrm{m}}(\mathrm{OT}(N)) \\ &\Leftrightarrow \mathrm{nf}(\mathcal{T}_{\mathrm{m}}(M)) = \mathrm{nf}(\mathcal{T}_{\mathrm{m}}(N)). \end{aligned}$$

- $\mathcal{O}$ contains $=_\beta$.
- Compatibility with abstraction: if $M =_{\mathcal{O}} N$ and $x \in \mathrm{fv}(M) \cap \mathrm{fv}(N)$ then

$$\mathrm{nf}(\mathcal{T}_{\mathrm{m}}(\lambda x.M)) = \lambda x.\mathrm{nf}(\mathcal{T}_{\mathrm{m}}(M)) = \lambda x.\mathrm{nf}(\mathcal{T}_{\mathrm{m}}(N)) = \mathrm{nf}(\mathcal{T}_{\mathrm{m}}(\lambda x.N)).$$

- Compatibility with application: same... □

What happened so far...

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

Well, we don't even know what a λI-model is...

... but we can try to keep applying the usual recipe:

- turn our linear approximation into a multi-type system,
- build the corresponding relational model.

Types:

$$
\begin{array}{rcll}
?\mathbf{T} & \ni & \sigma, \tau, \ldots & \coloneqq \quad \mathbf{0}_X \mid \alpha \qquad\qquad X \subseteq_{\mathsf{f}} \mathcal{V} \\
\mathbf{T} & \ni & \alpha, \beta, \ldots & \coloneqq \quad * \mid \bar{\alpha} \multimap \beta \qquad\qquad * \in \mathcal{A} \\
!\mathbf{T} & \ni & \bar{\alpha}, \bar{\beta}, \ldots & \coloneqq \quad [\,]_X \mid [\alpha_0, \ldots, \alpha_n] \qquad\qquad X \subseteq_{\mathsf{f}} \mathcal{V},\ n \in \mathbf{N}
\end{array}
$$

Contexts:

- Γ maps (all) variables to multisets of types (it's the usual one),
- Δ maps (only) the free variables of the typed term to sets of variables; intuitively, x that will be substituted with N is mapped to $\mathrm{fv}(N)$.

Judgements:

$$\Gamma \,;\, \Delta \vdash M : \sigma \qquad\qquad \Gamma \,;\, \Delta \vdash^{!} M : \bar{\alpha}$$

$$\frac{\mathrm{dom}(\Delta) = \mathrm{fv}(M)}{;\, \Delta \vdash M : \mathbf{0}_{\cup \mathrm{im}(\Delta)}}\ (\mathbf{0})$$

$$\frac{}{x : [\alpha]\ ;\ x : X \vdash x : \alpha}\ (\mathrm{ax}) \qquad \frac{\Gamma_0\ ;\ \Delta_0 \vdash M : \bar{\alpha} \multimap \beta \qquad \Gamma_1\ ;\ \Delta_1 \vdash^{!} N : \bar{\alpha} \qquad \Delta_0 \frown \Delta_1}{\Gamma_0 + \Gamma_1\ ;\ \Delta_0 \vee \Delta_1 \vdash MN : \beta}\ (@)$$

$$\frac{\Gamma, x : []\ ;\ \Delta, x : X \vdash M : \beta}{\Gamma\ ;\ \Delta \vdash \lambda x.M : []_X \multimap \beta}\ (\lambda^0) \qquad \frac{\Gamma, x : [\alpha_0, \ldots, \alpha_n]\ ;\ \Delta, x : X \vdash M : \beta}{\Gamma\ ;\ \Delta \vdash \lambda x.M : [\alpha_0, \ldots, \alpha_n] \multimap \beta}\ (\lambda^+)$$

$$\frac{\mathrm{dom}(\Delta) = \mathrm{fv}(M)}{;\, \Delta \vdash^{!} M : []_{\cup \mathrm{im}(\Delta)}}\ (!^0) \qquad \frac{\Gamma_0\ ;\ \Delta \vdash M : \alpha_0 \quad \ldots \quad \Gamma_n\ ;\ \Delta \vdash M : \alpha_n}{\Gamma_0 + \cdots + \Gamma_n\ ;\ \Delta \vdash^{!} M : [\alpha_0, \ldots, \alpha_n]}\ (!^+)$$

The typing system enjoys **subject reduction** and **subject expansion**. Good news!

The typing system enjoys **subject reduction** and **subject expansion**. Good news!

In addition, if one defines

$$[\![M]\!] := \{(\Gamma, \Delta, \sigma) \mid \Gamma \,;\, \Delta \vdash M \,:\, \sigma\}$$

we obtain the following

Theorem

For all $M, N \in \Lambda_{\mathsf{I}}$, $[\![M]\!] = [\![N]\!]$ iff. $M =_{\mathcal{O}} N$.

What about a λI-model?

It is not completely obvious how to extract a λI-model from this typing system.

What about a λI-model?

It is not completely obvious how to extract a λI-model from this typing system.

Neither is it obvious what a λI-model is!

- There's a proposition [Jacobs '93] with two instances (strict cpo's and some relational model with non-empty multisets),
  but it doesn't seem to be general enough for a model with theory $\mathcal{O}$ to exist.

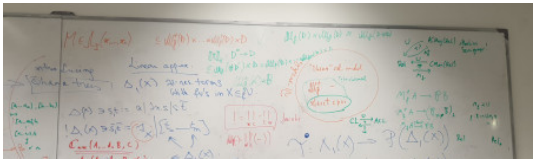
What about a λI-model?

It is not completely obvious how to extract a λI-model from this typing system.

Neither is it obvious what a λI-model is!

- There's a proposition [Jacobs '93] with two instances (strict cpo's and some relational model with non-empty multisets),
  but it doesn't seem to be general enough for a model with theory $\mathcal{O}$ to exist.
- It should be something more general than a λ-model,
  our candidate: a monoidal closed multicategory with contractions.

But don't ask me more, all the notes we have look like this:



Ohana trees for the full λ-calculus?

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

$$\mathrm{OT}(M) := \begin{cases} \lambda x_1 \dots x_n.y \quad \text{(pfv}(M_1), \dots, \text{pfv}(M_k)\text{ children: } \mathrm{OT}(M_1), \dots, \mathrm{OT}(M_k)) & \text{if } M \twoheadrightarrow_h \lambda x_1 \dots x_n.yM_1 \cdots M_k, \\ \bot_{\mathrm{pfv}(M)} & \text{otherwise.} \end{cases}$$

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?

Why should we care?

- **It's fun.** (Right?)

Why should we care?

- **It's fun.** (Right?)
- Since the λ-calculus is an (equationally) conservative extension of the λI-calculus, **λI-theories allow to β-separate λ-terms** (in the λ-calculus).

The double fixed-point combinator conjecture

Sometimes in λ-calculus we want to **separate** two λ-terms M and N,
i.e. prove that they are not β(η)-equivalent.

An example (open) problem.

- Take $\delta \coloneqq \lambda yx.x(yx)$. Then: [Böhm, van der Mey]
  - Y is a fpc $\iff$ $\delta Y =_\beta Y$ (hence is a fpc too)
  - Y is a fpc $\implies$ $Y\delta$ is a fpc
- Starting with $Y_0 \coloneqq$ your preferred fpc, define $Y_{n+1} \coloneqq Y_n\delta$.
  For $Y_0 \coloneqq$ Curry's fpc, all the terms of the sequence are β-distinct. [Klop '07]
  And in general? It's an open question.
- More generally, **is there a fpc Y s.t. $Y =_\beta Y\delta$?** [Statman '93]
  Conjecture: such a "double fixed-point combinator" doesn't exist.

Attacking a separation problem

Standard ideas:

- Find a separating context (as in Böhm's theorem),
  i.e. $C[\ ]$ such that $C[M] \to^* x$ and $C[N] \to^* y$.
- (...)
- Find a separating λ-theory (or λ-model),
  i.e. $=_{\mathcal{T}}$ such that $M \neq_{\mathcal{T}} N$ can be proved.

As for the double fpc problem:

- The Ohana theory doesn't help...
- ... but suggest some ideas! One may design evaluation trees such that in

$$Y\delta f =_\beta \delta(Y\delta)f =_\beta f(Y\delta f) =_\beta \dots$$

  something like the subterm $Y\delta f$ is remembered at infinity.

Why should we care?

- **It's fun.** (Right?)
- Since the λ-calculus is an (equationally) conservative extension of the λI-calculus, **λI-theories allow to β-separate λ-terms** (in the λ-calculus).
- **Long-term goal: to infinity and beyond.**
  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!