Understanding the λ-calculus via (non-)linearity and rewriting

Giulio Guerrieri

Aix-Marseille Université, LIS UMR 7020, Lirica (Marseille, France)

Lirica seminars

Marseille (France), September 19, 2022

Outline

Outline

From many λ-calculi . . .

The λ-calculus is the model of computation underlying

- functional programming languages (Haskell, OCaml, . . .)
- proof assistants (Coq, Isabelle/Hol, . . .).

Actually, there are many lambda-calculi, depending on

- the evaluation mechanism (e.g., call-by-name, call-by-value, call-by-need);
- computational feature the calculus aims to model (e.g., pure, non-deterministic);
- the type system (e.g. untyped, simply typed, second order).

The existence of N separate paradigms is troubling:

- it makes each calculus appear arbitrary (is there a more canonical language?)
- each time we create a new style of semantics (e.g. operational, denotational, continuations, etc.) we always need to do it N times—once for each paradigm.

From many λ-calculi . . .

The λ-calculus is the model of computation underlying

- functional programming languages (Haskell, OCaml, . . .)
- proof assistants (Coq, Isabelle/Hol, . . .).

Actually, there are many lambda-calculi, depending on

- the evaluation mechanism (e.g., call-by-name, call-by-value, call-by-need);
- computational feature the calculus aims to model (e.g., pure, non-deterministic);
- the type system (e.g. untyped, simply typed, second order).

The existence of N separate paradigms is troubling:

- it makes each calculus appear arbitrary (is there a more canonical language?)
- each time we create a new style of semantics (e.g. operational, denotational, continuations, etc.) we always need to do it N times—once for each paradigm.

From many λ-calculi . . .

The λ-calculus is the model of computation underlying

- functional programming languages (Haskell, OCaml, . . .)
- proof assistants (Coq, Isabelle/Hol, . . .).

Actually, there are many lambda-calculi, depending on

- the evaluation mechanism (e.g., call-by-name, call-by-value, call-by-need);
- computational feature the calculus aims to model (e.g., pure, non-deterministic);
- the type system (e.g. untyped, simply typed, second order).

The existence of N separate paradigms is troubling:

- it makes each calculus appear arbitrary (is there a more canonical language?)
- each time we create a new style of semantics (e.g. operational, denotational, continuations, etc.) we always need to do it N times—once for each paradigm.

. . . to one unifying and robust meta-theory of λ-calculi

Goal: A unifying and robust meta-theory of λ-calculi, rooted on:

1. linear logic $\rightsquigarrow$ unifying setting for different evaluation mechanisms;
2. elementary rewriting $\rightsquigarrow$ modular and robust approach to extensions of λ-calculi.

In this talk: focus on Point 1.

- the role of linear logic;
- different evaluation mechanisms.

. . . to one unifying and robust meta-theory of λ-calculi

Goal: A unifying and robust meta-theory of λ-calculi, rooted on:

1. linear logic $\rightsquigarrow$ unifying setting for different evaluation mechanisms;
2. elementary rewriting $\rightsquigarrow$ modular and robust approach to extensions of λ-calculi.

In this talk: focus on Point 1.

- the role of linear logic;
- different evaluation mechanisms.

The role of linear logic with respect to λ-calculi

Girard's linear logic (1987) provides new concepts and tools to study λ-calculi:

1. denotational models of linear logic provides denotational models for λ-calculi;
2. clear notion of resource and linear consumption
   $f : A \multimap B \;\approx\; f$ consumes a value of type A and transforms it into a value of type B;
3. quantitative analysis of computation
   - semantic tools to study execution time (De Carvalho *et al.*));
   - "compatible" with cost models (Accattoli *et al.*).
4. ...

LL also hints how to modify syntax and dynamics of λ-calculi to have "good properties".

The role of linear logic with respect to λ-calculi

Girard's linear logic (1987) provides new concepts and tools to study λ-calculi:

1. denotational models of linear logic provides denotational models for λ-calculi;
2. clear notion of resource and linear consumption
   $f : A \multimap B \;\approx\; f$ consumes a value of type A and transforms it into a value of type B;
3. quantitative analysis of computation
   - semantic tools to study execution time (De Carvalho *et al.*));
   - "compatible" with cost models (Accattoli *et al.*).
4. ...

LL also hints how to modify syntax and dynamics of λ-calculi to have "good properties".

The role of linear logic with respect to λ-calculi

Girard's linear logic (1987) provides new concepts and tools to study λ-calculi:

1. denotational models of linear logic provides denotational models for λ-calculi;
2. clear notion of resource and linear consumption
   $f : A \multimap B \;\approx\; f$ consumes a value of type A and transforms it into a value of type B;
3. quantitative analysis of computation
   - semantic tools to study execution time (De Carvalho *et al.*));
   - "compatible" with cost models (Accattoli *et al.*).
4. ...

LL also hints how to modify syntax and dynamics of λ-calculi to have "good properties".

The role of linear logic with respect to λ-calculi

Girard's linear logic (1987) provides new concepts and tools to study λ-calculi:

1. denotational models of linear logic provides denotational models for λ-calculi;
2. clear notion of resource and linear consumption
   $f : A \multimap B \;\approx\; f$ consumes a value of type A and transforms it into a value of type B;
3. quantitative analysis of computation
   - semantic tools to study execution time (De Carvalho *et al.*));
   - "compatible" with cost models (Accattoli *et al.*).
4. ...

LL also hints how to modify syntax and dynamics of λ-calculi to have "good properties".

The role of linear logic with respect to λ-calculi

Girard's linear logic (1987) provides new concepts and tools to study λ-calculi:

1. denotational models of linear logic provides denotational models for λ-calculi;
2. clear notion of resource and linear consumption
   $f : A \multimap B \;\approx\; f$ consumes a value of type A and transforms it into a value of type B;
3. quantitative analysis of computation
   - semantic tools to study execution time (De Carvalho *et al.*));
   - "compatible" with cost models (Accattoli *et al.*).
4. ...

LL also hints how to modify syntax and dynamics of λ-calculi to have "good properties".

Call-by-Name vs. Call-by-Value (for dummies)

- Call-by-Name (CbN): pass the argument to the calling function before evaluating it.
- Call-by-Value (CbV): pass the argument to the calling function after evaluating it.

Call-by-Name vs. Call-by-Value (for dummies)

- Call-by-Name (CbN): pass the argument to the calling function before evaluating it.
- Call-by-Value (CbV): pass the argument to the calling function after evaluating it.

$$
\begin{tikzcd}
& (\lambda x.\, x + x)(2 * 3) \arrow[dl, "\mathit{CbN}"'] \arrow[dr, "\mathit{CbV}"] & \\
(2 * 3) + (2 * 3) \arrow[d, "\mathit{CbN}"'] & & (\lambda x.\, x + x)6 \arrow[d, "\mathit{CbV}"] \\
6 + (2 * 3) \arrow[d, "\mathit{CbN}"'] & & 6 + 6 \arrow[d, "\mathit{CbV}"] \\
6 + 6 \arrow[d, "\mathit{CbN}"'] & & 12 \\
12 & &
\end{tikzcd}
$$

Call-by-Name vs. Call-by-Value (for dummies)

- Call-by-Name (CbN): pass the argument to the calling function before evaluating it.
- Call-by-Value (CbV): pass the argument to the calling function after evaluating it.

$$\begin{tikzcd} & (\lambda y.\lambda x.\,x)\Omega \arrow[dl, "CbN"'] \arrow[dr, "CbV"] & \\ \lambda x.x & & (\lambda y.\lambda x.\,x)\Omega \arrow[d, "CbV"] \\ & & \vdots \end{tikzcd}$$

(Ω is a diverging program)

Summing up, CbV is eager, that is,

1. CbV is smarter than CbN when the argument must be duplicated;
2. CbV is sillier than CbN when the argument must be discarded.

The CbN and CbV λ-calculi

λ-terms:	$t, s, r ::= v \mid ts$	(set: Λ)
λ-values:	$v ::= x \mid \lambda x\, t$	(set: Λ_v)

Reductions: $(\lambda x\, t)s \to_\beta t\{s/x\}$ (CbN) $\qquad (\lambda x\, t)v \to_{\beta^v} t\{v/x\}$ (CbV)

CbN and CbV λ-calculi have different operational and denotational semantics
$\rightsquigarrow$ in general, it is impossible to derive a property for CbV from CbN, or vice versa.

Examples, with $I := \lambda z.z$ (identity) and $\delta := \lambda z.zz$ (duplicator):

1. $(\lambda y.I)(\delta\delta)$ β-normalizes but β_v-diverges

$$(\lambda y.I)(\delta\delta) \to_\beta I \qquad (\lambda y.I)(\delta\delta) \to_{\beta_v} (\lambda y.I)(\delta\delta) \to_{\beta_v} \dots$$

2. $(\lambda x.\delta)(xx)\delta$ is β_v-normal but β-divergent: $(\lambda x.\delta)(xx)\delta \to_\beta \delta\delta \to_\beta \delta\delta \to_\beta \dots$

Linear logic allows us to see CbN and CbV in a unifying perspective!

The CbN and CbV λ-calculi

λ-terms:	$t, s, r ::= v \mid ts$	(set: Λ)
λ-values:	$v ::= x \mid \lambda x\, t$	(set: Λ_v)

Reductions: $(\lambda x\, t)s \to_\beta t\{s/x\}$ (CbN) $\quad (\lambda x\, t)v \to_{\beta^v} t\{v/x\}$ (CbV)

CbN and CbV λ-calculi have different operational and denotational semantics
$\rightsquigarrow$ in general, it is impossible to derive a property for CbV from CbN, or vice versa.

Examples, with $I \coloneqq \lambda z.z$ (identity) and $\delta \coloneqq \lambda z.zz$ (duplicator):

1. $(\lambda y.I)(\delta\delta)$ β-normalizes but β_v-diverges

$$(\lambda y.I)(\delta\delta) \to_\beta I \qquad (\lambda y.I)(\delta\delta) \to_{\beta_v} (\lambda y.I)(\delta\delta) \to_{\beta_v} \dots$$

2. $(\lambda x.\delta)(xx)\delta$ is β_v-normal but β-divergent: $(\lambda x.\delta)(xx)\delta \to_\beta \delta\delta \to_\beta \delta\delta \to_\beta \dots$

Linear logic allows us to see CbN and CbV in a unifying perspective!

The CbN and CbV λ-calculi

λ-terms:	$t, s, r ::= v \mid ts$	(set: Λ)
λ-values:	$v ::= x \mid \lambda x\, t$	(set: Λ_v)

Reductions: $(\lambda x\, t)s \to_\beta t\{s/x\}$ (CbN) $\qquad (\lambda x\, t)v \to_{\beta^v} t\{v/x\}$ (CbV)

CbN and CbV λ-calculi have different operational and denotational semantics
$\rightsquigarrow$ in general, it is impossible to derive a property for CbV from CbN, or vice versa.

Examples, with $I := \lambda z.z$ (identity) and $\delta := \lambda z.zz$ (duplicator):

1. $(\lambda y.I)(\delta\delta)$ β-normalizes but β_v-diverges

$$(\lambda y.I)(\delta\delta) \to_\beta I \qquad (\lambda y.I)(\delta\delta) \to_{\beta_v} (\lambda y.I)(\delta\delta) \to_{\beta_v} \dots$$

2. $(\lambda x.\delta)(xx)\delta$ is β_v-normal but β-divergent: $(\lambda x.\delta)(xx)\delta \to_\beta \delta\delta \to_\beta \delta\delta \to_\beta \dots$

Linear logic allows us to see CbN and CbV in a unifying perspective!

The CbN and CbV λ-calculi

λ-terms:	$t, s, r ::= v \mid ts$	(set: Λ)
λ-values:	$v ::= x \mid \lambda x\, t$	(set: Λ_v)

Reductions: $(\lambda x\, t)s \to_\beta t\{s/x\}$ (CbN) $\qquad (\lambda x\, t)v \to_{\beta^v} t\{v/x\}$ (CbV)

CbN and CbV λ-calculi have different operational and denotational semantics
$\rightsquigarrow$ in general, it is impossible to derive a property for CbV from CbN, or vice versa.

Examples, with $I := \lambda z.z$ (identity) and $\delta := \lambda z.zz$ (duplicator):

1. $(\lambda y.I)(\delta\delta)$ β-normalizes but β_v-diverges

$$(\lambda y.I)(\delta\delta) \to_\beta I \qquad (\lambda y.I)(\delta\delta) \to_{\beta_v} (\lambda y.I)(\delta\delta) \to_{\beta_v} \dots$$

2. $(\lambda x.\delta)(xx)\delta$ is β_v-normal but β-divergent: $(\lambda x.\delta)(xx)\delta \to_\beta \delta\delta \to_\beta \delta\delta \to_\beta \dots$

Linear logic allows us to see CbN and CbV in a unifying perspective!

The CbN and CbV λ-calculi

λ-terms:	$t, s, r ::= v \mid ts$	(set: Λ)
λ-values:	$v ::= x \mid \lambda x\, t$	(set: Λ_v)

Reductions: $(\lambda x\, t)s \to_\beta t\{s/x\}$ (CbN) $\qquad (\lambda x\, t)v \to_{\beta^v} t\{v/x\}$ (CbV)

CbN and CbV λ-calculi have different operational and denotational semantics
$\rightsquigarrow$ in general, it is impossible to derive a property for CbV from CbN, or vice versa.

Examples, with $I := \lambda z.z$ (identity) and $\delta := \lambda z.zz$ (duplicator):

1. $(\lambda y.I)(\delta\delta)$ β-normalizes but β_v-diverges

$$(\lambda y.I)(\delta\delta) \to_\beta I \qquad (\lambda y.I)(\delta\delta) \to_{\beta_v} (\lambda y.I)(\delta\delta) \to_{\beta_v} \dots$$

2. $(\lambda x.\delta)(xx)\delta$ is β_v-normal but β-divergent: $(\lambda x.\delta)(xx)\delta \to_\beta \delta\delta \to_\beta \delta\delta \to_\beta \dots$

Linear logic allows us to see CbN and CbV in a unifying perspective!

Outline

The Curry-Howard-Girard correspondence

Logic		Computer Science
formula	$\leftrightsquigarrow$	type
proof	$\leftrightsquigarrow$	program
cut-elimination	$\leftrightsquigarrow$	evaluation
coherence	$\leftrightsquigarrow$	termination
different encodings of intuitionistic arrow in LL	$\leftrightsquigarrow$	different evaluation mechanisms

$\rightsquigarrow$ Tools from intuitionistic linear logic (ILL) can be used to study properties of:

- call-by-name evaluation via Girard's translation $(\cdot)^{\mathtt{N}}$,
- call-by-value evaluation via Girard's translation $(\cdot)^{\mathtt{V}}$.

The Curry-Howard-Girard correspondence

Logic		Computer Science
formula	$\leftrightsquigarrow$	type
proof	$\leftrightsquigarrow$	program
cut-elimination	$\leftrightsquigarrow$	evaluation
coherence	$\leftrightsquigarrow$	termination
different encodings of intuitionistic arrow in LL	$\leftrightsquigarrow$	different evaluation mechanisms

$\rightsquigarrow$ Tools from intuitionistic linear logic (ILL) can be used to study properties of:

- call-by-name evaluation via Girard's translation $(\cdot)^{\mathtt{N}}$,
- call-by-value evaluation via Girard's translation $(\cdot)^{\mathtt{V}}$.

The Curry-Howard-Girard correspondence

Logic		Computer Science
formula	$\leftrightsquigarrow$	type
proof	$\leftrightsquigarrow$	program
cut-elimination	$\leftrightsquigarrow$	evaluation
coherence	$\leftrightsquigarrow$	termination
different encodings of intuitionistic arrow in LL	$\leftrightsquigarrow$	different evaluation mechanisms

$\rightsquigarrow$ Tools from intuitionistic linear logic (ILL) can be used to study properties of:

- call-by-name evaluation via Girard's translation $(\cdot)^{\mathtt{N}}$,
- call-by-value evaluation via Girard's translation $(\cdot)^{\mathtt{V}}$.

The two Girard's translations of IL into ILL (1987)

well-known translation $(\cdot)^{\mathtt{N}}$

$$X^{\mathtt{N}} = X$$
$$(A \to B)^{\mathtt{N}} = {!A^{\mathtt{N}}} \multimap B^{\mathtt{N}}$$
$$(\Gamma \vdash A)^{\mathtt{N}} = {!\Gamma^{\mathtt{N}}} \vdash A^{\mathtt{N}}$$

"boring" translation $(\cdot)^{\mathtt{V}}$

$$X^{\mathtt{V}} = X$$
$$(A \to B)^{\mathtt{V}} = {!A^{\mathtt{V}}} \multimap {!B^{\mathtt{V}}}$$
$$(\Gamma \vdash A)^{\mathtt{V}} = {!\Gamma^{\mathtt{V}}} \vdash {!A^{\mathtt{V}}}$$

$$\begin{tikzcd} \mathrm{IL} \arrow[r, bend left, "(\cdot)^{\mathtt{N}}"] \arrow[r, bend right, "(\cdot)^{\mathtt{V}}"'] & \mathrm{ILL} \end{tikzcd}$$ (and then proof-nets)

simply typed Λ = IL (via Curry-Howard)

(untyped) Λ = IL + unique atomic type o + type identity $o = o \to o$

$$\begin{tikzcd} \text{(untyped) } \Lambda \arrow[r, bend left, "(\cdot)^{\mathtt{N}}", "o = !o \multimap o"'] \arrow[r, bend right, "o = !o \multimap !o", "(\cdot)^{\mathtt{V}}"'] & \text{ILL (and then proof-nets)} \end{tikzcd}$$

The two Girard's translations of IL into ILL (1987)

well-known translation $(\cdot)^{\mathtt{N}}$	"boring" translation $(\cdot)^{\mathtt{V}}$
$X^{\mathtt{N}} = X$	$X^{\mathtt{V}} = X$
$(A \to B)^{\mathtt{N}} = !A^{\mathtt{N}} \multimap B^{\mathtt{N}}$	$(A \to B)^{\mathtt{V}} = !A^{\mathtt{V}} \multimap !B^{\mathtt{V}}$
$(\Gamma \vdash A)^{\mathtt{N}} = !\Gamma^{\mathtt{N}} \vdash A^{\mathtt{N}}$	$(\Gamma \vdash A)^{\mathtt{V}} = !\Gamma^{\mathtt{V}} \vdash !A^{\mathtt{V}}$

$$\begin{tikzcd} \mathrm{IL} \arrow[r, bend left, "(\cdot)^{\mathtt{N}}"] \arrow[r, bend right, "(\cdot)^{\mathtt{V}}"'] & \mathrm{ILL} \end{tikzcd} \quad \text{(and then proof-nets)}$$

simply typed Λ = IL (via Curry-Howard)

(untyped) Λ = IL + unique atomic type o + type identity $o = o \to o$

$$\text{(untyped) } \Lambda \begin{tikzcd} {} \arrow[r, bend left, "(\cdot)^{\mathtt{N}}", "o=!o\multimap o"'] \arrow[r, bend right, "o=!o\multimap !o", "(\cdot)^{\mathtt{V}}"'] & \mathrm{ILL} \end{tikzcd} \text{ (and then proof-nets)}$$

Girard's first translation: $(\cdot)^{\mathtt{N}}$

$$X^{\mathtt{N}} = X$$
$$(A \to B)^{\mathtt{N}} = !A^{\mathtt{N}} \multimap B^{\mathtt{N}}$$
$$(\Gamma \vdash A)^{\mathtt{N}} = !\Gamma^{\mathtt{N}} \vdash A^{\mathtt{N}}$$

(natural deduction for IL) (sequent calculus for ILL)

$$\dfrac{}{x : A \vdash x : A}\ ax \quad \rightsquigarrow \quad \dfrac{\dfrac{}{A^{\mathtt{N}} \vdash A^{\mathtt{N}}}\ ax}{!A^{\mathtt{N}} \vdash A^{\mathtt{N}}}\ der$$

$$\dfrac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x\, M : A \to B}\ \to_i \quad \rightsquigarrow \quad \dfrac{!\Gamma^{\mathtt{N}}, !A^{\mathtt{N}} \vdash B^{\mathtt{N}}}{!\Gamma^{\mathtt{N}} \vdash !A^{\mathtt{N}} \multimap B^{\mathtt{N}}}\ \parr$$

$$\dfrac{\Gamma \vdash M : A \to B \qquad \Delta \vdash N : A}{\Gamma, \Delta \vdash MN : B}\ \to_e \quad \rightsquigarrow \quad \dfrac{!\Gamma^{\mathtt{N}} \vdash !A^{\mathtt{N}} \multimap B^{\mathtt{N}} \qquad \dfrac{\dfrac{!\Delta^{\mathtt{N}} \vdash A^{\mathtt{N}}}{!\Delta^{\mathtt{N}} \vdash !A^{\mathtt{N}}}\ ! \qquad \dfrac{}{B^{\mathtt{N}} \vdash B^{\mathtt{N}}}\ ax}{!\Delta^{\mathtt{N}}, !A^{\mathtt{N}} \multimap B^{\mathtt{N}} \vdash B^{\mathtt{N}}}\ \otimes}{!\Gamma^{\mathtt{N}}, !\Delta^{\mathtt{N}} \vdash B^{\mathtt{N}}}\ cut$$

The translation $(\cdot)^{\mathtt{N}}$ puts a ! in front of every formula on the left-hand side of $\vdash$

$\rightsquigarrow$ the translation of the structural rules is obvious.

Girard's first translation: $(\cdot)^{\mathtt{N}}$

$$X^{\mathtt{N}} = X$$
$$(A \to B)^{\mathtt{N}} = !A^{\mathtt{N}} \multimap B^{\mathtt{N}}$$
$$(\Gamma \vdash A)^{\mathtt{N}} = !\Gamma^{\mathtt{N}} \vdash A^{\mathtt{N}}$$

(natural deduction for IL) (sequent calculus for ILL)

$$\frac{}{x : A \vdash x : A}\ ax \quad \rightsquigarrow \quad \dfrac{\dfrac{}{A^{\mathtt{N}} \vdash A^{\mathtt{N}}}\ ax}{!A^{\mathtt{N}} \vdash A^{\mathtt{N}}}\ der$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x\, M : A \to B}\ \to_i \quad \rightsquigarrow \quad \frac{!\Gamma^{\mathtt{N}}, !A^{\mathtt{N}} \vdash B^{\mathtt{N}}}{!\Gamma^{\mathtt{N}} \vdash !A^{\mathtt{N}} \multimap B^{\mathtt{N}}}\ \parr$$

$$\frac{\Gamma \vdash M : A \to B \qquad \Delta \vdash N : A}{\Gamma, \Delta \vdash MN : B}\ \to_e \quad \rightsquigarrow \quad \dfrac{!\Gamma^{\mathtt{N}} \vdash !A^{\mathtt{N}} \multimap B^{\mathtt{N}} \qquad \dfrac{\dfrac{!\Delta^{\mathtt{N}} \vdash A^{\mathtt{N}}}{!\Delta^{\mathtt{N}} \vdash !A^{\mathtt{N}}}\ ! \qquad \dfrac{}{B^{\mathtt{N}} \vdash B^{\mathtt{N}}}\ ax}{!\Delta^{\mathtt{N}}, !A^{\mathtt{N}} \multimap B^{\mathtt{N}} \vdash B^{\mathtt{N}}}\ \otimes}{!\Gamma^{\mathtt{N}}, !\Delta^{\mathtt{N}} \vdash B^{\mathtt{N}}}\ cut$$

The translation $(\cdot)^{\mathtt{N}}$ puts a ! in front of every formula on the left-hand side of $\vdash$
$\rightsquigarrow$ the translation of the structural rules is obvious.

Girard's second ("boring") translation: $(\cdot)^{\mathtt{v}}$

$$X^{\mathtt{v}} = X$$
$$(A \to B)^{\mathtt{v}} = !A^{\mathtt{v}} \multimap !B^{\mathtt{v}}$$
$$(\Gamma \vdash A)^{\mathtt{v}} = !\Gamma^{\mathtt{v}} \vdash !A^{\mathtt{v}}$$

(natural deduction for IL) (sequent calculus for ILL)

$$\dfrac{}{x : A \vdash x : A}\,ax \quad \rightsquigarrow \quad \dfrac{}{!A^{\mathtt{v}} \vdash !A^{\mathtt{v}}}\,ax$$

$$\dfrac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x\, M : A \to B}\,\to_i \quad \rightsquigarrow \quad \dfrac{\dfrac{!\Gamma^{\mathtt{v}}, !A^{\mathtt{v}} \vdash !B^{\mathtt{v}}}{!\Gamma^{\mathtt{v}} \vdash !A^{\mathtt{v}} \multimap !B^{\mathtt{v}}}\,\text{⅋}}{!\Gamma^{\mathtt{v}} \vdash !(!A^{\mathtt{v}} \multimap !B^{\mathtt{v}})}\,!$$

$$\dfrac{\Gamma \vdash M : A \to B \qquad \Delta \vdash N : A}{\Gamma, \Delta \vdash MN : B}\,\to_e \quad \rightsquigarrow \quad \dfrac{!\Gamma^{\mathtt{v}} \vdash !(!A^{\mathtt{v}} \multimap !B^{\mathtt{v}}) \qquad \dfrac{\dfrac{!\Delta^{\mathtt{v}} \vdash !A^{\mathtt{v}} \qquad \dfrac{}{!B^{\mathtt{v}} \vdash !B^{\mathtt{v}}}\,ax}{!\Delta^{\mathtt{v}}, !A^{\mathtt{v}} \multimap !B^{\mathtt{v}} \vdash !B^{\mathtt{v}}}\,\otimes}{!\Delta^{\mathtt{v}}, !(!A^{\mathtt{v}} \multimap !B^{\mathtt{v}}) \vdash !B^{\mathtt{v}}}\,der}{!\Gamma^{\mathtt{v}}, !\Delta^{\mathtt{v}} \vdash !B^{\mathtt{v}}}\,cut$$

The translation $(\cdot)^{\mathtt{v}}$ puts a ! in front of every formula on the left-hand side of $\vdash$
$\rightsquigarrow$ the translation of the structural rules is obvious.

Girard's second ("boring") translation: $(\cdot)^{\mathtt{v}}$

$$
\begin{aligned}
X^{\mathtt{v}} &= X \\
(A \to B)^{\mathtt{v}} &= !A^{\mathtt{v}} \multimap !B^{\mathtt{v}} \\
(\Gamma \vdash A)^{\mathtt{v}} &= !\Gamma^{\mathtt{v}} \vdash !A^{\mathtt{v}}
\end{aligned}
$$

(natural deduction for IL) (sequent calculus for ILL)

$$
\dfrac{}{x : A \vdash x : A}\,ax \quad \rightsquigarrow \quad \dfrac{}{!A^{\mathtt{v}} \vdash !A^{\mathtt{v}}}\,ax
$$

$$
\dfrac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x\, M : A \to B}\,\to_i \quad \rightsquigarrow \quad \dfrac{\dfrac{!\Gamma^{\mathtt{v}}, !A^{\mathtt{v}} \vdash !B^{\mathtt{v}}}{!\Gamma^{\mathtt{v}} \vdash !A^{\mathtt{v}} \multimap !B^{\mathtt{v}}}\,\parr}{!\Gamma^{\mathtt{v}} \vdash !(!A^{\mathtt{v}} \multimap !B^{\mathtt{v}})}\,!
$$

$$
\dfrac{\Gamma \vdash M : A \to B \qquad \Delta \vdash N : A}{\Gamma, \Delta \vdash MN : B}\,\to_e \quad \rightsquigarrow \quad \dfrac{!\Gamma^{\mathtt{v}} \vdash !(!A^{\mathtt{v}} \multimap !B^{\mathtt{v}}) \qquad \dfrac{\dfrac{!\Delta^{\mathtt{v}} \vdash !A^{\mathtt{v}} \qquad \dfrac{}{!B^{\mathtt{v}} \vdash !B^{\mathtt{v}}}\,ax}{!\Delta^{\mathtt{v}}, !A^{\mathtt{v}} \multimap !B^{\mathtt{v}} \vdash !B^{\mathtt{v}}}\,\otimes}{!\Delta^{\mathtt{v}}, !(!A^{\mathtt{v}} \multimap !B^{\mathtt{v}}) \vdash !B^{\mathtt{v}}}\,der}{!\Gamma^{\mathtt{v}}, !\Delta^{\mathtt{v}} \vdash !B^{\mathtt{v}}}\,cut
$$

The translation $(\cdot)^{\mathtt{v}}$ puts a ! in front of every formula on the left-hand side of $\vdash$
$\rightsquigarrow$ the translation of the structural rules is obvious.

An example: from IL (natural deduction) . . .

$$\pi \quad = \quad \dfrac{\dfrac{\dfrac{}{a:A \vdash a:A}\,ax}{a:A, c:C \vdash a:A}\,w}{a:A \vdash \lambda c\, a : C \to A}\,\to_i \qquad \dfrac{\dfrac{}{x:B \to C \vdash x:B \to C}\,ax \qquad \dfrac{}{b:B \vdash b:B}\,ax}{b:B, x:B \to C \vdash xb:C}\,\to_e$$

$$\overline{\qquad a:A, b:B, x:B \to C \vdash (\lambda c\, a)(xb):A \qquad}\,\to_e$$

$$\downarrow_{\mathbf{cut}}$$

$$\mathrm{nf}(\pi) \quad = \quad \dfrac{\dfrac{\dfrac{}{a:A \vdash a:A}\,ax}{a:A, b:B, \vdash a:A}\,w}{a:A, b:B, x:B \to C \vdash a:A}\,w$$

An example: from IL (natural deduction)...

$$\pi \quad = \quad \dfrac{\dfrac{\dfrac{}{a\!:\!A \vdash a\!:\!A}\,ax}{a\!:\!A, c\!:\!C \vdash a\!:\!A}\,w}{a\!:\!A \vdash \lambda c\, a\!:\!C \to A}\,\to_i \qquad \dfrac{\dfrac{}{x\!:\!B \to C \vdash x\!:\!B \to C}\,ax \qquad \dfrac{}{b\!:\!B \vdash b\!:\!B}\,ax}{b\!:\!B, x\!:\!B \to C \vdash xb\!:\!C}\,\to_e$$

$$\overline{\qquad a\!:\!A, b\!:\!B, x\!:\!B \to C \vdash (\lambda c\, a)(xb)\!:\!A \qquad}\,\to_e$$

$$\downarrow_{\text{cut}}$$

$$\mathrm{nf}(\pi) \quad = \quad \dfrac{\dfrac{\dfrac{}{a\!:\!A \vdash a\!:\!A}\,ax}{a\!:\!A, b\!:\!B, \vdash a\!:\!A}\,w}{a\!:\!A, b\!:\!B, x\!:\!B \to C \vdash a\!:\!A}\,w$$

Curry-Howard: $(\lambda c\, a)(xb) \to_\beta a$

An example: from IL (natural deduction) . . .

$$
\begin{tikzcd}
\text{intuit. logic (IL):} \arrow[d, "\mathit{Curry-Howard}"] & \pi \arrow[rr, "*"', "\mathsf{cut}"] \arrow[d, "\underline{(\cdot)}"] & & \mathsf{nf}(\pi) \arrow[d, "\underline{(\cdot)}"] \\
\lambda\text{-calculus:} & \underline{\pi} \arrow[r, "*", "\beta"'] & \mathsf{nf}(\underline{\pi}) = & \underline{\mathsf{nf}(\pi)}
\end{tikzcd}
$$

An example: ... to ILL via $(\cdot)^{\mathtt{N}}$

$$
\pi^{\mathtt{N}} =
\begin{prooftree}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$A \vdash A$}
\RightLabel{der}
\UnaryInfC{$!A \vdash A$}
\RightLabel{w}
\UnaryInfC{$!A, !C \vdash A$}
\RightLabel{$\parr$}
\UnaryInfC{$!A \vdash !C \multimap A$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!B \multimap C \vdash !B \multimap C$}
\RightLabel{der}
\UnaryInfC{$!(!B \multimap C) \vdash !B \multimap C$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$B \vdash B$}
\RightLabel{der}
\UnaryInfC{$!B \vdash B$}
\RightLabel{$!$}
\UnaryInfC{$!B \vdash !B$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$C \vdash C$}
\RightLabel{$\otimes$}
\BinaryInfC{$!B, !B \multimap C \vdash C$}
\RightLabel{cut}
\BinaryInfC{$!B, !(!B \multimap C) \vdash C$}
\RightLabel{$!$}
\UnaryInfC{$!B, !(!B \multimap C) \vdash !C$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$A \vdash A$}
\RightLabel{$\otimes$}
\BinaryInfC{$!C \multimap A, !B, !(!B \multimap C) \vdash A$}
\RightLabel{cut}
\BinaryInfC{$a : !A, b : !B, x : !(!B \multimap C) \vdash (\lambda c\, a)(xb) : A$}
\end{prooftree}
$$

$$\mathbf{cut}\downarrow_{+}$$

$$
\mathrm{nf}(\pi^{\mathtt{N}}) =
\begin{prooftree}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$a : A \vdash a : A$}
\RightLabel{der}
\UnaryInfC{$a : !A \vdash a : A$}
\RightLabel{w}
\UnaryInfC{$a : !A, b : !B, \vdash a : A$}
\RightLabel{w}
\UnaryInfC{$a : !A, b : !B, x : !(!B \multimap C) \vdash a : A$}
\end{prooftree}
= (\mathrm{nf}(\pi))^{\mathtt{N}}
$$

An example: ... to ILL via $(\cdot)^{\mathtt{N}}$

$$
\begin{tikzcd}
\text{intuit. logic (IL):} \arrow[d, "(\cdot)^{\mathtt{N}}"] & \pi \arrow[rr, "*", "\mathsf{cut}"'] \arrow[d, "(\cdot)^{\mathtt{N}}"] & & \mathsf{nf}(\pi) \arrow[d, "(\cdot)^{\mathtt{N}}"] \\
\text{intuit. LL (ILL):} & \pi^{\mathtt{N}} \arrow[r, "*", "\mathsf{cut}"'] & \mathsf{nf}(\pi^{\mathtt{N}}) = (\mathsf{nf}(\pi))^{\mathtt{N}} &
\end{tikzcd}
$$

An example: ...to ILL via $(\cdot)^{\mathtt{N}}$

$$
\pi^{\mathtt{N}} = \quad
\begin{prooftree}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$A \vdash A$}
\RightLabel{der}
\UnaryInfC{$!A \vdash A$}
\RightLabel{w}
\UnaryInfC{$!A, !C \vdash A$}
\RightLabel{$⅋$}
\UnaryInfC{$!A \vdash !C \multimap A$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!B \multimap C \vdash !B \multimap C$}
\RightLabel{der}
\UnaryInfC{$!(!B \multimap C) \vdash !B \multimap C$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$B \vdash B$}
\RightLabel{der}
\UnaryInfC{$!B \vdash B$}
\RightLabel{$!$}
\UnaryInfC{$!B \vdash !B$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$C \vdash C$}
\RightLabel{$\otimes$}
\BinaryInfC{$!B, !B \multimap C \vdash C$}
\RightLabel{cut}
\BinaryInfC{$!B, !(!B \multimap C) \vdash C$}
\RightLabel{$!$}
\UnaryInfC{$!B, !(!B \multimap C) \vdash !C$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$A \vdash A$}
\RightLabel{$\otimes$}
\BinaryInfC{$!C \multimap A, !B, !(!B \multimap C) \vdash A$}
\RightLabel{cut}
\BinaryInfC{$a : !A, b : !B, x : !(!B \multimap C) \vdash (\lambda c\, a)(xb) : A$}
\end{prooftree}
$$

$$
{}_{\mathbf{cut}}\!\downarrow_{+}
$$

$$
\mathrm{nf}(\pi^{\mathtt{N}}) \; = \;
\begin{prooftree}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$a : A \vdash a : A$}
\RightLabel{der}
\UnaryInfC{$a : !A \vdash a : A$}
\RightLabel{w}
\UnaryInfC{$a : !A, b : !B, \vdash a : A$}
\RightLabel{w}
\UnaryInfC{$a : !A, b : !B, x : !(!B \multimap C) \vdash a : A$}
\end{prooftree}
\; = \; (\mathrm{nf}(\pi))^{\mathtt{N}}
$$

Curry-Howard: $\quad (\lambda c\, a)(xb) \to_{\beta} a$

An example: ...to ILL via $(\cdot)^{\mathtt{v}}$

$$
\pi^{\mathtt{v}} =
\begin{prooftree}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!A \vdash !A$}
\RightLabel{w}
\UnaryInfC{$!A, !C \vdash !A$}
\RightLabel{$\parr$}
\UnaryInfC{$!A \vdash !C \multimap !A$}
\RightLabel{$!$}
\UnaryInfC{$!A \vdash !(!C \multimap !A)$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!(!B \multimap !C) \vdash !(!B \multimap !C)$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!B \vdash !B$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!C \vdash !C$}
\RightLabel{$\otimes$}
\BinaryInfC{$!B, !B \multimap !C \vdash !C$}
\RightLabel{der}
\UnaryInfC{$!B, !(!B \multimap !C) \vdash C$}
\RightLabel{cut}
\BinaryInfC{$!B, !(!B \multimap !C) \vdash !C$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$A \vdash A$}
\RightLabel{$\otimes$}
\BinaryInfC{$!C \multimap A, !B, !(!B \to C) \vdash A$}
\RightLabel{der}
\UnaryInfC{$!(!C \multimap A), !B, !(!B \multimap !C) \vdash A$}
\RightLabel{cut}
\BinaryInfC{$a : !A, b : !B, x : !(!B \multimap !C) \vdash (\lambda c\, a)(xb) : A$}
\end{prooftree}
$$

$$\downarrow_{\mathbf{cut}}^{+}$$

$$
\mathrm{nf}(\pi^{\mathtt{v}}) =
\begin{prooftree}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!B \vdash !B$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!A \vdash !A$}
\RightLabel{w}
\UnaryInfC{$!C, !A \vdash !A$}
\RightLabel{$\otimes$}
\BinaryInfC{$!A, !B, !B \multimap !C \vdash !A$}
\RightLabel{der}
\UnaryInfC{$a : !A, b : !B, x : !(!B \multimap !C) \vdash a[xb/c] : !A$}
\end{prooftree}
\neq
\begin{prooftree}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!A \vdash !A$}
\RightLabel{w}
\UnaryInfC{$!B, !A \vdash !A$}
\RightLabel{w}
\UnaryInfC{$a : !A, b : !B, x : !(!B \multimap !C) \vdash a : !A$}
\end{prooftree}
= (\mathrm{nf}(\pi))^{\mathtt{v}}
$$

An example: . . . to ILL via $(\cdot)^{\mathtt{V}}$

$$\begin{tikzcd}
\text{intuit. logic (IL):} \arrow[d, "(\cdot)^{\mathtt{V}}"] & \pi \arrow[rr, "*"', "\text{cut}"] \arrow[d, "(\cdot)^{\mathtt{V}}"] & & \mathsf{nf}(\pi) \arrow[d, "(\cdot)^{\mathtt{V}}"] \\
\text{intuit. LL (ILL):} & \pi^{\mathtt{V}} \arrow[r, "*"', "\text{cut}"] & \mathsf{nf}(\pi^{\mathtt{V}}) \neq & (\mathsf{nf}(\pi))^{\mathtt{V}}
\end{tikzcd}$$

An example: ...to ILL via $(\cdot)^{\mathtt{v}}$

$$
\pi^{\mathtt{v}} =
\begin{prooftree}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!A \vdash\, !A$}
\RightLabel{w}
\UnaryInfC{$!A, !C \vdash\, !A$}
\RightLabel{$\parr$}
\UnaryInfC{$!A \vdash\, !C \multimap\, !A$}
\RightLabel{$!$}
\UnaryInfC{$!A \vdash\, !(!C \multimap\, !A)$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!(!B \multimap\, !C) \vdash\, !(!B \multimap\, !C)$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!B \vdash\, !B$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!C \vdash\, !C$}
\RightLabel{$\otimes$}
\BinaryInfC{$!B, !B \multimap\, !C \vdash\, !C$}
\RightLabel{der}
\UnaryInfC{$!B, !(!B \multimap\, !C) \vdash C$}
\RightLabel{cut}
\BinaryInfC{$!B, !(!B \multimap\, !C) \vdash\, !C$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$A \vdash A$}
\RightLabel{$\otimes$}
\BinaryInfC{$!C \multimap A, !B, !(!B \to C) \vdash A$}
\RightLabel{der}
\UnaryInfC{$!(!C \multimap A), !B, !(!B \multimap\, !C) \vdash A$}
\RightLabel{cut}
\BinaryInfC{$\color{blue}{a}:!A, \color{blue}{b}:!B, \color{blue}{x}:!(!B \multimap\, !C) \vdash \color{blue}{(\lambda c\, a)(xb)}:A$}
\end{prooftree}
$$

$$\downarrow_{\mathbf{cut}}^{+}$$

$$
\mathrm{nf}(\pi^{\mathtt{v}}) =
\begin{prooftree}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!B \vdash\, !B$}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!A \vdash\, !A$}
\RightLabel{w}
\UnaryInfC{$!C, !A \vdash\, !A$}
\RightLabel{$\otimes$}
\BinaryInfC{$!A, !B, !B \multimap\, !C \vdash\, !A$}
\RightLabel{der}
\UnaryInfC{$\color{blue}{a}:!A, \color{blue}{b}:!B, \color{blue}{x}:!(!B \multimap\, !C) \vdash \color{blue}{a[xb/c]}:!A$}
\end{prooftree}
\;\color{blue}{\neq}\;
\begin{prooftree}
\AxiomC{}
\RightLabel{ax}
\UnaryInfC{$!A \vdash\, !A$}
\RightLabel{w}
\UnaryInfC{$!B, !A \vdash\, !A$}
\RightLabel{w}
\UnaryInfC{$\color{blue}{a}:!A, \color{blue}{b}:!B, \color{blue}{x}:!(!B \multimap\, !C) \vdash \color{blue}{a}:!A$}
\end{prooftree}
= (\mathrm{nf}(\pi))^{\mathtt{v}}
$$

$\color{blue}{(\lambda c\, a)(xb)} \to_{\beta_v} a[xb/c]$ (i.e. `let` $c := xb$ `in` a) $\approx \color{blue}{(\lambda c\, a)(xb)}$ ...boring (according to Girard).

But $a[xb/c]$ is β_v-normal!

In the λ-calculus there are two evaluation mechanisms:

- *call-by-name* (CbN, β-reduction): no restriction in firing a β-redex;
- *call-by-value* (CbV, β_v-reduction): a β-redex $(\lambda x\, t)s$ can be fired only if s is a value.

ILL (and proof-nets) cut-elimination simulates $\begin{cases} \beta\text{-reduction via the translation } (\cdot)^{\mathtt{N}} \\ \beta_v\text{-reduction via the translation } (\cdot)^{\mathtt{V}} \end{cases}$

- via $(\cdot)^{\mathtt{N}}$ every argument is translated by a box
  $\rightsquigarrow$ every argument can be duplicated or discarded (CbN discipline);
- via $(\cdot)^{\mathtt{V}}$ every (and only) abstraction or variable is translated by a box
  $\rightsquigarrow$ only abstraction or variable can be duplicated or discarded (CbV discipline).

The two Girard's logical translations can explain the two different evaluation mechanisms inside the same setting, bringing them into the scope of the Curry-Howard isomorphism.

Call-by-name vs. call-by-value from a Linear Logic point of view (1 of 2)

In the λ-calculus there are two evaluation mechanisms:

- *call-by-name* (CbN, β-reduction): no restriction in firing a β-redex;
- *call-by-value* (CbV, β_v-reduction): a β-redex $(\lambda x\, t)s$ can be fired only if s is a value.

ILL (and proof-nets) cut-elimination simulates $\begin{cases} \beta\text{-reduction via the translation } (\cdot)^{\mathtt{N}} \\ \beta_v\text{-reduction via the translation } (\cdot)^{\mathtt{V}} \end{cases}$

- via $(\cdot)^{\mathtt{N}}$ every argument is translated by a box
  $\rightsquigarrow$ every argument can be duplicated or discarded (CbN discipline);
- via $(\cdot)^{\mathtt{V}}$ every (and only) abstraction or variable is translated by a box
  $\rightsquigarrow$ only abstraction or variable can be duplicated or discarded (CbV discipline).

The two Girard's logical translations can explain the two different evaluation mechanisms inside the same setting, bringing them into the scope of the Curry-Howard isomorphism.

Call-by-name vs. call-by-value from a Linear Logic point of view

In the λ-calculus there are two evaluation mechanisms:

- *call-by-name* (CbN, β-reduction): no restriction in firing a β-redex;
- *call-by-value* (CbV, β_v-reduction): a β-redex $(\lambda x\, t)s$ can be fired only if s is a value.

ILL (and proof-nets) cut-elimination simulates $\begin{cases} \beta\text{-reduction via the translation } (\cdot)^{\mathtt{N}} \\ \beta_v\text{-reduction via the translation } (\cdot)^{\mathtt{V}} \end{cases}$

- via $(\cdot)^{\mathtt{N}}$ every argument is translated by a box
  $\rightsquigarrow$ every argument can be duplicated or discarded (CbN discipline);
- via $(\cdot)^{\mathtt{V}}$ every (and only) abstraction or variable is translated by a box
  $\rightsquigarrow$ only abstraction or variable can be duplicated or discarded (CbV discipline).

The two Girard's logical translations can explain the two different evaluation mechanisms inside the same setting, bringing them into the scope of the Curry-Howard isomorphism.

ILL (and proof-nets) syntax is extremely expressive and powerful, but it is too general for the computational purpose of representing purely functional programs.

Question: Can we internalize the two Girard's translations in a variant of the λ-calculus?

Yes! There are several examples in the literature, *e.g.*

Call-by-name, call-by-value, call-by-need, and the linear lambda calculus. John Maraist, Martin Odersky, David Turner, and Philip Wadler. MFPS, 1995.

Idea: Let linear logic guide the study and design of models of computation.

We propose here an alternative solution, the bang calculus: it differs from the linear λ-calculi of Maraist *et al.* for some technical aspects.

All the "good" results proved for Maraist *et al.*'s linear λ-calculus hold also for the bang calculus, and in a "better" way.

ILL (and proof-nets) syntax is extremely expressive and powerful, but it is too general for the computational purpose of representing purely functional programs.

Question: Can we internalize the two Girard's translations in a variant of the λ-calculus?

Yes! There are several examples in the literature, *e.g.*

Call-by-name, call-by-value, call-by-need, and the linear lambda calculus. John Maraist, Martin Odersky, David Turner, and Philip Wadler. MFPS, 1995.

Idea: Let linear logic guide the study and design of models of computation.

We propose here an alternative solution, the bang calculus: it differs from the linear λ-calculi of Maraist *et al.* for some technical aspects.

All the "good" results proved for Maraist *et al.*'s linear λ-calculus hold also for the bang calculus, and in a "better" way.

ILL (and proof-nets) syntax is extremely expressive and powerful, but it is too general for the computational purpose of representing purely functional programs.

Question: Can we internalize the two Girard's translations in a variant of the λ-calculus?

Yes! There are several examples in the literature, *e.g.*

Call-by-name, call-by-value, call-by-need, and the linear lambda calculus. John Maraist, Martin Odersky, David Turner, and Philip Wadler. MFPS, 1995.

Idea: Let linear logic guide the study and design of models of computation.

We propose here an alternative solution, the bang calculus: it differs from the linear λ-calculi of Maraist *et al.* for some technical aspects.

All the "good" results proved for Maraist *et al.*'s linear λ-calculus hold also for the bang calculus, and in a "better" way.

Outline

Syntax and reduction rules of the bang calculus

Syntax:

Terms $\qquad T, S, R ::= x \mid T^! \mid \lambda x\, T \mid TS \mid \mathrm{der}\, T \qquad$ (set: $!\Lambda$)

Reduction rules:

$$(\lambda x\, T)S^! \mapsto_\lambda T\{S/x\} \qquad \mathrm{der}(T^!) \mapsto_{\mathsf{d}} T \qquad \mapsto_{\mathsf{b}} := \mapsto_\lambda \cup \mapsto_{\mathsf{d}}$$

For any $\mathsf{r} \in \{\lambda, \mathsf{d}, \mathsf{b}\}$,

- r-*reduction* $\to_{\mathsf{r}}$ is the closure under any contexts of $\mapsto_{\mathsf{r}}$ (it fires any r-redex);
- *ground* r-*reduction* (or $\mathsf{r_g}$-*reduction*) $\to_{\mathsf{r_g}}$ does not fire r-redexes under !.

Idea: only boxes $T^!$ are duplicable and discardable ($\rightsquigarrow$ linear logic).

Proposition (confluence; G. & Ehrhard, 2016)

$\to_{\mathsf{b_g}}$ is diamond and $\to_{\mathsf{b}}$ is confluent.

Syntax and reduction rules of the bang calculus

Syntax:

$$\textit{Terms} \qquad T, S, R ::= x \mid T^! \mid \lambda x\, T \mid TS \mid \mathrm{der}\, T \qquad \text{(set: } !\Lambda)$$

Reduction rules:

$$(\lambda x\, T)S^! \mapsto_\lambda T\{S/x\} \qquad \mathrm{der}(T^!) \mapsto_{\mathsf{d}} T \qquad \mapsto_{\mathsf{b}} := \mapsto_\lambda \cup \mapsto_{\mathsf{d}}$$

For any $\mathsf{r} \in \{\lambda, \mathsf{d}, \mathsf{b}\}$,

- r-*reduction* $\to_{\mathsf{r}}$ is the closure under any contexts of $\mapsto_{\mathsf{r}}$ (it fires any r-redex);
- *ground* r-*reduction* (or r_{g}-*reduction*) $\to_{\mathsf{r}_{\mathsf{g}}}$ does not fire r-redexes under !.

Idea: only boxes $T^!$ are duplicable and discardable ($\rightsquigarrow$ linear logic).

Proposition (confluence; G. & Ehrhard, 2016)

$\to_{\mathsf{b}_{\mathsf{g}}}$ is diamond and $\to_{\mathsf{b}}$ is confluent.

Syntax and reduction rules of the bang calculus

Syntax:

$$\textit{Terms} \qquad T, S, R ::= x \mid T^{!} \mid \lambda x\, T \mid TS \mid \mathrm{der}\, T \qquad \text{(set: } !\Lambda)$$

Reduction rules:

$$(\lambda x\, T)S^{!} \mapsto_{\lambda} T\{S/x\} \qquad \mathrm{der}(T^{!}) \mapsto_{\mathsf{d}} T \qquad \mapsto_{\mathsf{b}} := \mapsto_{\lambda} \cup \mapsto_{\mathsf{d}}$$

For any $\mathsf{r} \in \{\lambda, \mathsf{d}, \mathsf{b}\}$,

- r-*reduction* $\to_{\mathsf{r}}$ is the closure under any contexts of $\mapsto_{\mathsf{r}}$ (it fires any r-redex);
- *ground* r-*reduction* (or r_{g}-*reduction*) $\to_{\mathsf{r}_{\mathsf{g}}}$ does not fire r-redexes under !.

Idea: only boxes $T^{!}$ are duplicable and discardable ($\rightsquigarrow$ linear logic).

Proposition (confluence; G. & Ehrhard, 2016)

$\to_{\mathsf{b}_{\mathsf{g}}}$ is diamond and $\to_{\mathsf{b}}$ is confluent.

The bang calculus and LL proof-nets

The bang calculus gives a nice decomposition of the two Girard's translations $(\cdot)^{\mathtt{N}}$ and $(\cdot)^{\mathtt{V}}$.

$$\begin{tikzcd}[column sep=huge] \text{(untyped) } \Lambda \arrow[r, bend left, "(\cdot)^{\mathtt{N}}", "o = !o \multimap o"'] \arrow[r, bend right, "o = !o \multimap !o", "(\cdot)^{\mathtt{V}}"'] & \text{ILL (and then proof-nets)} \end{tikzcd}$$

Rmk (Ehrhard, 2016): Bang calculus $\approx$ untyped version of Levy's Call-by-Push-Value:

- $(\cdot)^{!} \approx$ thunk
- $\mathrm{der} \approx$ force

The bang calculus and LL proof-nets

The bang calculus gives a nice decomposition of the two Girard's translations $(\cdot)^{\mathtt{N}}$ and $(\cdot)^{\mathtt{V}}$.

$$
\begin{tikzcd}
\text{(untyped) } \Lambda \arrow[rr, bend left, "(\cdot)^{\mathtt{N}}", "o = !o \multimap o"'] \arrow[rr, bend right, "o = !o \multimap !o", "(\cdot)^{\mathtt{V}}"'] \arrow[dr, bend right, "(\cdot)^{\mathtt{v}}"'] \arrow[dr, bend left, "(\cdot)^{\mathtt{n}}"] & & \text{ILL (and then proof-nets)} \\
& !\Lambda \arrow[ur, "(\cdot)^{\bullet}", "o = !o \& (!o \multimap o)"'] &
\end{tikzcd}
$$

Reduction $\to_{\mathsf{b}}$ corresponds to the cut-elimination in LL proof-nets.
Reduction $\to_{\mathsf{b_g}}$ corresponds to the cut-elimination at depth 0 in LL proof-nets.

Rmk (Ehrhard, 2016): Bang calculus $\approx$ untyped version of Levy's Call-by-Push-Value:

- $(\cdot)^{!} \approx$ thunk
- $\mathrm{der} \approx$ force

The bang calculus and LL proof-nets

The bang calculus gives a nice decomposition of the two Girard's translations $(\cdot)^{\mathtt{N}}$ and $(\cdot)^{\mathtt{V}}$.

$$
\begin{tikzcd}
\text{(untyped) }\Lambda \arrow[rr, bend left, "(\cdot)^{\mathtt{N}}", "o=!o\multimap o"'] \arrow[rr, bend right, "o=!o\multimap !o", "(\cdot)^{\mathtt{V}}"'] \arrow[dr, bend left, "(\cdot)^{\mathtt{n}}"] \arrow[dr, bend right, "(\cdot)^{\mathtt{v}}"'] & & \text{ILL (and then proof-nets)} \\
& !\Lambda \arrow[ur, "(\cdot)^{\bullet}", "o=!o\&(!o\multimap o)"'] &
\end{tikzcd}
$$

Reduction $\to_{\mathbf{b}}$ corresponds to the cut-elimination in LL proof-nets.
Reduction $\to_{\mathbf{b_g}}$ corresponds to the cut-elimination at depth 0 in LL proof-nets.

Rmk (Ehrhard, 2016): Bang calculus $\approx$ untyped version of Levy's Call-by-Push-Value:

- $(\cdot)^!$ $\approx$ thunk
- der $\approx$ force

Denotational models for LL

A *denotational model for LL* is given by:

1. a symmetric monoidal closed category $(\mathcal{L}, \otimes, 1, \lambda, \rho, \alpha, \sigma)$; we use $X \multimap Y$ for the object of linear morphisms from X to Y;
2. $\mathcal{L}$ is cartesian with terminal object $\top$ and product $\&$;
   $\mathcal{L}$ is cocartesian with initial object 0 and coproduct $\oplus$;
3. a comonad $!\_ : \mathcal{L} \to \mathcal{L}$ with counit $\mathrm{der}_X \in \mathcal{L}(!X, X)$ (*dereliction*) and comultiplication $\mathrm{dig}_X \in \mathcal{L}(!X, !!X)$ (*digging*);
4. ...

Specific assumption: the unique morphism in $\mathcal{L}(0, \top)$ is an iso; to simplify, $0 = \top$.
(this assumption is fulfilled by many models of LL: relational, coherent, Scott, etc.)

Denotational models for LL

A *denotational model for LL* is given by:

1. a symmetric monoidal closed category $(\mathcal{L}, \otimes, 1, \lambda, \rho, \alpha, \sigma)$; we use $X \multimap Y$ for the object of linear morphisms from X to Y;
2. $\mathcal{L}$ is cartesian with terminal object $\top$ and product &;
   $\mathcal{L}$ is cocartesian with initial object 0 and coproduct $\oplus$;
3. a comonad $!\_ : \mathcal{L} \to \mathcal{L}$ with counit $\mathrm{der}_X \in \mathcal{L}(!X, X)$ (*dereliction*) and comultiplication $\mathrm{dig}_X \in \mathcal{L}(!X, !!X)$ (*digging*);
4. ...

Specific assumption: the unique morphism in $\mathcal{L}(0, \top)$ is an iso; to simplify, $0 = \top$.
(this assumption is fulfilled by many models of LL: relational, coherent, Scott, etc.)

Denotational models for the bang calculus built from LL

To have a model of the–untyped–bang calculus, take a retraction: an object $\mathcal{U}$ such that

$$!\mathcal{U} \mathbin{\&} (!\mathcal{U} \multimap \mathcal{U}) \lhd \mathcal{U}$$

Rmk (retractions): It follows that $!\mathcal{U} \lhd \mathcal{U}$ and

1. $!\mathcal{U} \multimap \mathcal{U} \lhd \mathcal{U}$ (semantic CbN version of $o \to o = o$);
2. $!\mathcal{U} \multimap !\mathcal{U} \lhd \mathcal{U}$ (semantic CbV version of $o \to o = o$).

A term T with $\vec{x} = (x_1, \ldots, x_n) \supseteq \mathrm{fv}(T)$ is interpreted by a morphism $[\![T]\!]_{\vec{x}} \colon (!\mathcal{U})^{\otimes n} \to \mathcal{U}$. The (omitted) definition is by induction on T, using the morphisms in $\mathcal{L}$ sketched before.

Theorem (invariance under reduction; G. & Ehrhard 2016)

Let $T, S \in !\Lambda$ and $\mathrm{fv}(T) \subseteq \vec{x}$. If $T \to_{\mathsf{b}} S$ then $[\![T]\!]_{\vec{x}} = [\![S]\!]_{\vec{x}}$.

Denotational models for the bang calculus built from LL

To have a model of the–untyped–bang calculus, take a retraction: an object $\mathcal{U}$ such that

$$!\mathcal{U} \,\&\, (!\mathcal{U} \multimap \mathcal{U}) \lhd \mathcal{U}$$

Rmk (retractions): It follows that $!\mathcal{U} \lhd \mathcal{U}$ and

1. $!\mathcal{U} \multimap \mathcal{U} \lhd \mathcal{U}$ (semantic CbN version of $o \to o = o$);
2. $!\mathcal{U} \multimap !\mathcal{U} \lhd \mathcal{U}$ (semantic CbV version of $o \to o = o$).

A term T with $\vec{x} = (x_1, \ldots, x_n) \supseteq \mathsf{fv}(T)$ is interpreted by a morphism $[\![T]\!]_{\vec{x}} \colon (!\mathcal{U})^{\otimes n} \to \mathcal{U}$. The (omitted) definition is by induction on T, using the morphisms in $\mathcal{L}$ sketched before.

Theorem (invariance under reduction; G. & Ehrhard 2016)

Let $T, S \in !\Lambda$ and $\mathsf{fv}(T) \subseteq \vec{x}$. If $T \to_{\mathsf{b}} S$ then $[\![T]\!]_{\vec{x}} = [\![S]\!]_{\vec{x}}$.

Outline

Embedding CbN and CbV λ-calculi into the bang calculus

We can internalize the two Girard's translations into the bang calculus:

CbN translation $(\cdot)^{\mathrm{n}} \colon \Lambda \to !\Lambda$

$$x^{\mathrm{n}} = x$$
$$(\lambda x\, t)^{\mathrm{n}} = \lambda x\, t^{\mathrm{n}}$$
$$(ts)^{\mathrm{n}} = t^{\mathrm{n}}(s^{\mathrm{n}})^{!}$$

CbV translation $(\cdot)^{\mathrm{v}} \colon \Lambda \to !\Lambda$

$$x^{\mathrm{v}} = x^{!}$$
$$(\lambda x\, t)^{\mathrm{v}} = (\lambda x\, t^{\mathrm{v}})^{!}$$
$$(ts)^{\mathrm{v}} = (\mathrm{der}\, t^{\mathrm{v}})s^{\mathrm{v}}$$

The difference between $(\cdot)^{\mathrm{n}}$ and $(\cdot)^{\mathrm{v}}$ is only where $(\cdot)^{!}$ and der are placed.

Idea: in CbN any argument is duplicable and discardable, in CbV only λ's and variables.

Lemma (translations commute with substitution)

Let t and s be λ-terms.

1. One has that $t^{\mathrm{n}}\{s^{\mathrm{n}}/x\} = (t\{s/x\})^{\mathrm{n}}$.
2. If s is such that $s^{\mathrm{v}} = R^{!}$ for some $R \in !\Lambda$, then $t^{\mathrm{v}}\{R/x\} = (t\{s/x\})^{\mathrm{v}}$.

Embedding CbN and CbV λ-calculi into the bang calculus

We can internalize the two Girard's translations into the bang calculus:

CbN translation $(\cdot)^{\mathrm{n}} \colon \Lambda \to !\Lambda$

$$x^{\mathrm{n}} = x$$
$$(\lambda x\, t)^{\mathrm{n}} = \lambda x\, t^{\mathrm{n}}$$
$$(ts)^{\mathrm{n}} = t^{\mathrm{n}}(s^{\mathrm{n}})^{!}$$

CbV translation $(\cdot)^{\mathrm{v}} \colon \Lambda \to !\Lambda$

$$x^{\mathrm{v}} = x^{!}$$
$$(\lambda x\, t)^{\mathrm{v}} = (\lambda x\, t^{\mathrm{v}})^{!}$$
$$(ts)^{\mathrm{v}} = (\mathrm{der}\, t^{\mathrm{v}})s^{\mathrm{v}}$$

The difference between $(\cdot)^{\mathrm{n}}$ and $(\cdot)^{\mathrm{v}}$ is only where $(\cdot)^{!}$ and der are placed.

Idea: in CbN any argument is duplicable and discardable, in CbV only λ's and variables.

Lemma (translations commute with substitution)

Let t and s be λ-terms.

1. One has that $t^{\mathrm{n}}\{s^{\mathrm{n}}/x\} = (t\{s/x\})^{\mathrm{n}}$.
2. If s is such that $s^{\mathrm{v}} = R^{!}$ for some $R \in !\Lambda$, then $t^{\mathrm{v}}\{R/x\} = (t\{s/x\})^{\mathrm{v}}$.

The bang calculus is a conservative extension of CbN and CbV λ-cacluli

Theorem (Sound and complete simulations; G. & Manzonetto, 2018)

Let t be a λ-term.

1. *Conservative extension of CbN λ-calculus:*
   Soundness: If $t \to_\beta t'$ then $t^{\mathtt{n}} \to_\lambda t'^{\mathtt{n}}$ (and $t^{\mathtt{n}} \to_{\mathsf{b}} t'^{\mathtt{n}}$);
   Completeness: If $t^{\mathtt{n}} \to_{\mathsf{b}} S$ then $t^{\mathtt{n}} \to_\lambda S = t'^{\mathtt{n}}$ and $t \to_\beta t'$ for some λ-term t'.
2. *Conservative extension of CbV λ-calculus:*
   Soundness: If $t \to_{\beta^v} t'$ then $t^{\mathtt{v}} \to_{\mathsf{d}}\to_\lambda t'^{\mathtt{v}}$ (and hence $t^{\mathtt{v}} \to_{\mathsf{b}}\to_{\mathsf{b}} t'^{\mathtt{v}}$);
   Completeness: If $t^{\mathtt{v}} \to_{\mathsf{d}}\to_\lambda S$ then $S = t'^{\mathtt{v}}$ and $t \to_{\beta^v} t'$ for some λ-term t'.

These simulations are:

- modular: the ground CbX λ-calculus is simulated in the ground bang calculus (ground CbN = head reduction, ground CbV = weak reduction);
- quantitative sensitive: one β-step is simulated by one λ-step, and conversely.

In other linear λ-calculi, completeness of CBV translation fails!
$\rightsquigarrow$ A step in the CbV fragment of those calculi need not correspond to a β^v-step.

The bang calculus is a conservative extension of CbN and CbV λ-cacluli

Theorem (Sound and complete simulations; G. & Manzonetto, 2018)

Let t be a λ-term.

1. *Conservative extension of CbN λ-calculus:*
   Soundness: If $t \to_\beta t'$ then $t^{\mathtt{n}} \to_\lambda t'^{\mathtt{n}}$ (and $t^{\mathtt{n}} \to_{\mathsf{b}} t'^{\mathtt{n}}$);
   Completeness: If $t^{\mathtt{n}} \to_{\mathsf{b}} S$ then $t^{\mathtt{n}} \to_\lambda S = t'^{\mathtt{n}}$ and $t \to_\beta t'$ for some λ-term t'.

2. *Conservative extension of CbV λ-calculus:*
   Soundness: If $t \to_{\beta^v} t'$ then $t^{\mathtt{v}} \to_{\mathsf{d}}\to_\lambda t'^{\mathtt{v}}$ (and hence $t^{\mathtt{v}} \to_{\mathsf{b}}\to_{\mathsf{b}} t'^{\mathtt{v}}$);
   Completeness: If $t^{\mathtt{v}} \to_{\mathsf{d}}\to_\lambda S$ then $S = t'^{\mathtt{v}}$ and $t \to_{\beta^v} t'$ for some λ-term t'.

These simulations are:

- modular: the ground CbX λ-calculus is simulated in the ground bang calculus (ground CbN = head reduction, ground CbV = weak reduction);
- quantitative sensitive: one β-step is simulated by one λ-step, and conversely.

In other linear λ-calculi, completeness of CBV translation fails!
$\rightsquigarrow$ A step in the CbV fragment of those calculi need not correspond to a β^v-step.

The bang calculus is a conservative extension of CbN and CbV λ-cacluli

Theorem (Sound and complete simulations; G. & Manzonetto, 2018)

Let t be a λ-term.

1. *Conservative extension of CbN λ-calculus:*
   Soundness: If $t \to_{\beta} t'$ then $t^{\mathtt{n}} \to_{\lambda} t'^{\mathtt{n}}$ (and $t^{\mathtt{n}} \to_{\mathsf{b}} t'^{\mathtt{n}}$);
   Completeness: If $t^{\mathtt{n}} \to_{\mathsf{b}} S$ then $t^{\mathtt{n}} \to_{\lambda} S = t'^{\mathtt{n}}$ and $t \to_{\beta} t'$ for some λ-term t'.

2. *Conservative extension of CbV λ-calculus:*
   Soundness: If $t \to_{\beta^v} t'$ then $t^{\mathtt{v}} \to_{\mathsf{d}}\to_{\lambda} t'^{\mathtt{v}}$ (and hence $t^{\mathtt{v}} \to_{\mathsf{b}}\to_{\mathsf{b}} t'^{\mathtt{v}}$);
   Completeness: If $t^{\mathtt{v}} \to_{\mathsf{d}}\to_{\lambda} S$ then $S = t'^{\mathtt{v}}$ and $t \to_{\beta^v} t'$ for some λ-term t'.

These simulations are:

- modular: the ground CbX λ-calculus is simulated in the ground bang calculus (ground CbN = head reduction, ground CbV = weak reduction);
- quantitative sensitive: one β-step is simulated by one λ-step, and conversely.

In other linear λ-calculi, completeness of CBV translation fails!
$\leadsto$ A step in the CbV fragment of those calculi need not correspond to a β^v-step.

The bang calculus is a conservative extension of CbN and CbV λ-cacluli

Theorem (Sound and complete simulations; G. & Manzonetto, 2018)

Let t be a λ-term.

1. *Conservative extension of CbN λ-calculus:*
   Soundness: If $t \to_\beta t'$ then $t^{\mathtt{n}} \to_\lambda t'^{\mathtt{n}}$ (and $t^{\mathtt{n}} \to_{\mathsf{b}} t'^{\mathtt{n}}$);
   Completeness: If $t^{\mathtt{n}} \to_{\mathsf{b}} S$ then $t^{\mathtt{n}} \to_\lambda S = t'^{\mathtt{n}}$ and $t \to_\beta t'$ for some λ-term t'.
2. *Conservative extension of CbV λ-calculus:*
   Soundness: If $t \to_{\beta^v} t'$ then $t^{\mathtt{v}} \to_{\mathsf{d}}\to_\lambda t'^{\mathtt{v}}$ (and hence $t^{\mathtt{v}} \to_{\mathsf{b}}\to_{\mathsf{b}} t'^{\mathtt{v}}$);
   Completeness: If $t^{\mathtt{v}} \to_{\mathsf{d}}\to_\lambda S$ then $S = t'^{\mathtt{v}}$ and $t \to_{\beta^v} t'$ for some λ-term t'.

These simulations are:

- modular: the ground CbX λ-calculus is simulated in the ground bang calculus (ground CbN $=$ head reduction, ground CbV $=$ weak reduction);
- quantitative sensitive: one β-step is simulated by one λ-step, and conversely.

In other linear λ-calculi, completeness of CBV translation fails!
$\rightsquigarrow$ A step in the CbV fragment of those calculi need not correspond to a β^v-step.

The target of the CbN translation in the bang calculus

target of CbN translation into $!\Lambda$: $\quad T, S ::= x \mid TS^{!} \mid \lambda x\, T \quad$ (set: $!\Lambda^{\mathrm{n}}$)

Rmk: $t^{\mathrm{n}} \in\, !\Lambda^{\mathrm{n}}$ for any $t \in \Lambda$, and conversely, for any $T \in\, !\Lambda^{\mathrm{n}}$, $T^{\mathrm{n}} = t$ for some $t \in \Lambda$.

Rmk: In $!\Lambda^{\mathrm{n}}$, the construct der never occurs $\rightsquigarrow$ in $!\Lambda^{\mathrm{n}}$, hence $\to_\lambda = \to_{\mathsf{b}}$.

$\rightsquigarrow$ The CbN λ-calculus is isomorphic to the fragment $!\Lambda^{\mathrm{n}}$ of the bang calculus.

$$\begin{tikzcd}
\Lambda \ni t \arrow[rr, "\beta"] \arrow[d, "(\cdot)^{\mathrm{n}}"] & & s \in \Lambda \arrow[d, "(\cdot)^{\mathrm{n}}"] \\
!\Lambda \ni t^{\mathrm{n}} \arrow[rr, "\mathsf{b}"] & & s^{\mathrm{n}} \in\, !\Lambda
\end{tikzcd}$$

Corollary (Preservations with respect to CbN λ-calculus; G. & Manzonetto, 2018)

1. (*CbN equational theory*) Let $t, s \in \Lambda$: $t \simeq_\beta s$ iff $t^{\mathrm{n}} \simeq_{\mathsf{b}} s^{\mathrm{n}}$.
2. (*CbN normal forms*) Let $t \in \Lambda$: t is β-normal iff t^{n} is b-normal.

The target of the CbN translation in the bang calculus

target of CbN translation into $!\Lambda$: $\quad T, S ::= x \mid TS^! \mid \lambda x\, T \qquad$ (set: $!\Lambda^{\mathtt{n}}$)

Rmk: $t^{\mathtt{n}} \in\, !\Lambda^{\mathtt{n}}$ for any $t \in \Lambda$, and conversely, for any $T \in\, !\Lambda^{\mathtt{n}}$, $T^{\mathtt{n}} = t$ for some $t \in \Lambda$.

Rmk: In $!\Lambda^{\mathtt{n}}$, the construct der never occurs $\rightsquigarrow$ in $!\Lambda^{\mathtt{n}}$, hence $\to_\lambda = \to_{\mathsf{b}}$.

$\rightsquigarrow$ The CbN λ-calculus is isomorphic to the fragment $!\Lambda^{\mathtt{n}}$ of the bang calculus.

$$\begin{tikzcd}
\Lambda \ni t \arrow[r, "\beta"] \arrow[d, "(\cdot)^{\mathtt{n}}"] & s \in \Lambda \arrow[d, "(\cdot)^{\mathtt{n}}"] \\
!\Lambda \ni t^{\mathtt{n}} \arrow[r, "\mathsf{b}"] & s^{\mathtt{n}} \in\, !\Lambda
\end{tikzcd}$$

Corollary (Preservations with respect to CbN λ-calculus; G. & Manzonetto, 2018)

1. (*CbN equational theory*) Let $t, s \in \Lambda$: $t \simeq_\beta s$ iff $t^{\mathtt{n}} \simeq_{\mathsf{b}} s^{\mathtt{n}}$.
2. (*CbN normal forms*) Let $t \in \Lambda$: t is β-normal iff $t^{\mathtt{n}}$ is b-normal.

The target of the CbN translation in the bang calculus

target of CbN translation into $!\Lambda$: $T, S ::= x \mid TS^! \mid \lambda x\, T$ (set: $!\Lambda^{\mathtt{n}}$)

Rmk: $t^{\mathtt{n}} \in !\Lambda^{\mathtt{n}}$ for any $t \in \Lambda$, and conversely, for any $T \in !\Lambda^{\mathtt{n}}$, $T^{\mathtt{n}} = t$ for some $t \in \Lambda$.

Rmk: In $!\Lambda^{\mathtt{n}}$, the construct der never occurs $\rightsquigarrow$ in $!\Lambda^{\mathtt{n}}$, hence $\to_\lambda = \to_{\mathsf{b}}$.

$\rightsquigarrow$ The CbN λ-calculus is isomorphic to the fragment $!\Lambda^{\mathtt{n}}$ of the bang calculus.

$$\begin{tikzcd}
\Lambda \ni t \arrow[r, "\beta"] \arrow[d, "(\cdot)^{\mathtt{n}}"] & s \in \Lambda \arrow[d, "(\cdot)^{\mathtt{n}}"] \\
!\Lambda \ni t^{\mathtt{n}} \arrow[r, "\mathsf{b}"] & s^{\mathtt{n}} \in !\Lambda
\end{tikzcd}$$

Corollary (Preservations with respect to CbN λ-calculus; G. & Manzonetto, 2018)

1. (*CbN equational theory*) Let $t, s \in \Lambda$: $t \simeq_\beta s$ iff $t^{\mathtt{n}} \simeq_{\mathsf{b}} s^{\mathtt{n}}$.
2. (*CbN normal forms*) Let $t \in \Lambda$: t is β-normal iff $t^{\mathtt{n}}$ is b-normal.

The target of the CbV translation in the bang calculus

target of CbV translation into $!\Lambda$: $\quad M, N ::= U^! \mid \mathrm{der}\, MN \mid UM \qquad$ (set: $!\Lambda^{\mathrm{v}}$)

$\qquad U ::= x \mid \lambda x\, M \qquad$ (set: $!\Lambda^{\mathrm{v}}_v$).

Rmk: For any $t \in \Lambda$, $t^{\mathrm{v}} \in\, !\Lambda^{\mathrm{v}}$; in particular, for any $v \in \Lambda_v$, $v^{\mathrm{v}} = U^!$ for some $U \in\, !\Lambda^{\mathrm{v}}_v$.

The converse fails: $(\lambda x\, xx)^{\mathrm{v}} = \Delta' \to_{\mathsf{d}} \Delta$, where Δ is b-normal and $\nexists$ λ-term $t : t^{\mathrm{v}} = \Delta$.
Note that $\lambda x\, xx$ is β^v-normal but $(\lambda x\, xx)^{\mathrm{v}} = \Delta'$ is not b-normal.

$\rightsquigarrow$ The CbV λ-calculus is "morally" isomorphic to the fragment $!\Lambda^{\mathrm{v}}$ of the bang calculus.

$$\begin{tikzcd}
\Lambda \ni t \arrow[r, "\beta^v"] \arrow[d, "(\cdot)^{\mathrm{v}}"] & s \in \Lambda \arrow[d, "(\cdot)^{\mathrm{v}}"] \\
!\Lambda \ni t^{\mathrm{v}} \arrow[r, "\mathsf{b}"] & s^{\mathrm{v}} \in\, !\Lambda
\end{tikzcd}$$

Corollary (Preservations with respect to CbV λ-calculus; G & Manzonetto, 2018)

1. (*CbV equational theory*) Let $t, s \in \Lambda$: $t \simeq_{\beta^v} s$ iff $t^{\mathrm{v}} \simeq_{\mathsf{b}} s^{\mathrm{v}}$.

This is false in other linear λ-calculi!

The target of the CbV translation in the bang calculus

target of CbV translation into $!\Lambda$: $\quad M, N ::= U^! \mid \mathrm{der}\, MN \mid UM \quad$ (set: $!\Lambda^{\mathrm{v}}$)
$\quad U ::= x \mid \lambda x\, M \quad$ (set: $!\Lambda^{\mathrm{v}}_v$).

Rmk: For any $t \in \Lambda$, $t^{\mathrm{v}} \in {!\Lambda^{\mathrm{v}}}$; in particular, for any $v \in \Lambda_v$, $v^{\mathrm{v}} = U^!$ for some $U \in {!\Lambda^{\mathrm{v}}_v}$.

The converse fails: $(\lambda x\, xx)^{\mathrm{v}} = \Delta' \to_{\mathsf{d}} \Delta$, where Δ is b-normal and $\nexists$ λ-term $t : t^{\mathrm{v}} = \Delta$.
Note that $\lambda x\, xx$ is β^v-normal but $(\lambda x\, xx)^{\mathrm{v}} = \Delta'$ is not b-normal.

$\rightsquigarrow$ The CbV λ-calculus is "morally" isomorphic to the fragment $!\Lambda^{\mathrm{v}}$ of the bang calculus.

$$\begin{tikzcd}
\Lambda \ni t \arrow[r, "\beta^v"] \arrow[d, "(\cdot)^{\mathrm{v}}"] & s \in \Lambda \arrow[d, "(\cdot)^{\mathrm{v}}"] \\
!\Lambda \ni t^{\mathrm{v}} \arrow[r, "\mathsf{b}"] & s^{\mathrm{v}} \in !\Lambda
\end{tikzcd}$$

Corollary (Preservations with respect to CbV λ-calculus; G & Manzonetto, 2018)

1. (*CbV equational theory*) Let $t, s \in \Lambda$: $t \simeq_{\beta^v} s$ iff $t^{\mathrm{v}} \simeq_{\mathsf{b}} s^{\mathrm{v}}$.

This is false in other linear λ-calculi!

The target of the CbV translation in the bang calculus

target of CbV translation into $!\Lambda$: $\quad M, N ::= U^! \mid \mathrm{der}\, MN \mid UM \quad$ (set: $!\Lambda^{\mathrm{v}}$)

$U ::= x \mid \lambda x\, M \quad$ (set: $!\Lambda^{\mathrm{v}}_v$).

Rmk: For any $t \in \Lambda$, $t^{\mathrm{v}} \in !\Lambda^{\mathrm{v}}$; in particular, for any $v \in \Lambda_v$, $v^{\mathrm{v}} = U^!$ for some $U \in !\Lambda^{\mathrm{v}}_v$.

The converse fails: $(\lambda x\, xx)^{\mathrm{v}} = \Delta' \to_{\mathsf{d}} \Delta$, where Δ is b-normal and $\nexists$ λ-term $t : t^{\mathrm{v}} = \Delta$.
Note that $\lambda x\, xx$ is β^v-normal but $(\lambda x\, xx)^{\mathrm{v}} = \Delta'$ is not b-normal.

$\leadsto$ The CbV λ-calculus is "morally" isomorphic to the fragment $!\Lambda^{\mathrm{v}}$ of the bang calculus.

$$\begin{array}{ccc} \Lambda \ni t & \xrightarrow{\quad \beta^v \quad} & s \in \Lambda \\ \Big\downarrow {\scriptstyle (\cdot)^{\mathrm{v}}} & & \Big\downarrow {\scriptstyle (\cdot)^{\mathrm{v}}} \\ !\Lambda \ni t^{\mathrm{v}} & \xrightarrow{\quad \mathsf{b} \quad} & s^{\mathrm{v}} \in !\Lambda \end{array}$$

Corollary (Preservations with respect to CbV λ-calculus; G & Manzonetto, 2018)

1. (*CbV equational theory*) Let $t, s \in \Lambda$: $t \simeq_{\beta^v} s$ iff $t^{\mathrm{v}} \simeq_{\mathsf{b}} s^{\mathrm{v}}$.

This is false in other linear λ-calculi!

Outline

Factorization of the semantics of CBN λ-calculus (G. & Manzonetto, 2018)

Retraction $!\mathcal{U} \multimap \mathcal{U} \lhd \mathcal{U}$ in $\mathcal{L}$ lifts to a retraction $!\mathcal{U} \multimap \mathcal{U} \lhd \mathcal{U}$ in $\mathcal{L}_!$ (co-Kleisli of $\mathcal{L}$ via !)
$\rightsquigarrow$ any denotational model $\mathcal{U}$ of the bang calculus is a denotat. model of CbN λ-calculus.

$$[t]^{\mathtt{n}}_{\vec{x}} = \text{usual CbN interpretation of a } \lambda\text{-term } t \text{ in } \mathcal{U}.$$
$$[\![t^{\mathtt{n}}]\!]_{\vec{x}} = \text{bang interpretation of the CbN translation of a } \lambda\text{-term } t \text{ in } \mathcal{U}.$$

Question: For a λ-term t, what is the relationship between $[t]^{\mathtt{n}}$ and $[\![t^{\mathtt{n}}]\!]$?

Theorem (Factorization of *any* denotational semantics of CBN λ-calculus)

For every λ-term t, $[t]^{\mathtt{n}}_{\vec{x}} = [\![t^{\mathtt{n}}]\!]_{\vec{x}}$ (up to Seely's isos).

$$\begin{tikzcd}
\Lambda \ni t \arrow[rr, "{[\cdot]^{\mathtt{n}}}"] \arrow[dr, "(\cdot)^{\mathtt{n}}"'] & \circlearrowright & {[t]^{\mathtt{n}} = [\![t^{\mathtt{n}}]\!] \in \mathcal{U}} \\
& t^{\mathtt{n}} \in\ !\Lambda \arrow[ur, "{[\![\cdot]\!]}"'] &
\end{tikzcd}$$

Factorization of the semantics of CBN λ-calculus (G. & Manzonetto, 2018)

Retraction $!\mathcal{U} \multimap \mathcal{U} \lhd \mathcal{U}$ in $\mathcal{L}$ lifts to a retraction $!\mathcal{U} \multimap \mathcal{U} \lhd \mathcal{U}$ in $\mathcal{L}_!$ (co-Kleisli of $\mathcal{L}$ via !)
$\rightsquigarrow$ any denotational model $\mathcal{U}$ of the bang calculus is a denotat. model of CbN λ-calculus.

$$[t]^{\mathbf{n}}_{\vec{x}} = \text{usual CbN interpretation of a } \lambda\text{-term } t \text{ in } \mathcal{U}.$$
$$[\![t^{\mathtt{n}}]\!]_{\vec{x}} = \text{bang interpretation of the CbN translation of a } \lambda\text{-term } t \text{ in } \mathcal{U}.$$

Question: For a λ-term t, what is the relationship between $[t]^{\mathbf{n}}$ and $[\![t^{\mathtt{n}}]\!]$?

Theorem (Factorization of *any* denotational semantics of CBN λ-calculus)

For every λ-term t, $[t]^{\mathbf{n}}_{\vec{x}} = [\![t^{\mathtt{n}}]\!]_{\vec{x}}$ (up to Seely's isos).

$$\begin{tikzcd}
\Lambda \ni t \arrow[rr, "{[\cdot]^{\mathbf{n}}}"] \arrow[dr, "(\cdot)^{\mathtt{n}}"'] & \circlearrowright & {[t]^{\mathbf{n}} = [\![t^{\mathtt{n}}]\!] \in \mathcal{U}} \\
& t^{\mathtt{n}} \in !\Lambda \arrow[ur, "{[\![\cdot]\!]}"'] &
\end{tikzcd}$$

Factorization of the semantics of CbV λ-calculus? (G. & Manzonetto, 2018)

The functor ! is a strong monad on the Kleisli category $\mathcal{L}_!$ $\rightsquigarrow$ retraction $!\mathcal{U} \multimap !\mathcal{U} \lhd \mathcal{U}$
$\rightsquigarrow$ any denotational model $\mathcal{U}$ of the bang calculus is a denotat. model of CbV λ-calculus.

$$[t]^{\mathsf{v}}_{\vec{x}} = \text{usual CbV interpretation of a } \lambda\text{-term } t \text{ in } \mathcal{U}.$$
$$[\![t^{\mathsf{v}}]\!]_{\vec{x}} = \text{bang interpretation of the CbV translation of a } \lambda\text{-term } t \text{ in } \mathcal{U}.$$

Question: For a λ-term t, what is the relationship between $[t]^{\mathsf{v}}$ and $[\![t^{\mathsf{v}}]\!]$?

Theorem (Non-factorization of *some* denotational semantics of CbV λ-calculus)

In the relational semantics, there is a λ-term t such that $[t]^{\mathsf{v}}_{\vec{x}} \subsetneq [\![t^{\mathsf{v}}]\!]_{\vec{x}}$.

$$\begin{tikzcd}
\Lambda \ni t \arrow[rr, "{[\cdot]^{\mathsf{v}}}"] \arrow[dr, "(\cdot)^{\mathsf{v}}"'] & \not\circlearrowleft & {[t]^{\mathsf{v}} \neq [\![t^{\mathsf{v}}]\!] \in \mathcal{U}} \\
& t^{\mathsf{v}} \in !\Lambda \arrow[ur, "{[\![\cdot]\!]}"'] &
\end{tikzcd}$$

Conjecture: There still exists a relationship in CbV between $[t]^{\mathsf{v}}$ and $[\![t^{\mathsf{v}}]\!]$, but it should be more sophisticated than in CbN. Maybe we should use logical relations between the two.

Factorization of the semantics of CbV λ-calculus? (G. & Manzonetto, 2018)

The functor ! is a strong monad on the Kleisli category $\mathcal{L}_!$ $\rightsquigarrow$ retraction $!\mathcal{U} \multimap !\mathcal{U} \lhd \mathcal{U}$
$\rightsquigarrow$ any denotational model $\mathcal{U}$ of the bang calculus is a denotat. model of CbV λ-calculus.

$$[t]^{\mathsf{v}}_{\vec{x}} = \text{usual CbV interpretation of a } \lambda\text{-term } t \text{ in } \mathcal{U}.$$
$$[\![t^{\mathsf{v}}]\!]_{\vec{x}} = \text{bang interpretation of the CbV translation of a } \lambda\text{-term } t \text{ in } \mathcal{U}.$$

Question: For a λ-term t, what is the relationship between $[t]^{\mathsf{v}}$ and $[\![t^{\mathsf{v}}]\!]$?

Theorem (Non-factorization of *some* denotational semantics of CbV λ-calculus)

In the relational semantics, there is a λ-term t such that $[t]^{\mathsf{v}}_{\vec{x}} \subsetneq [\![t^{\mathsf{v}}]\!]_{\vec{x}}$.

$$\begin{tikzcd}
\Lambda \ni t \arrow[rr, "{[\cdot]^{\mathsf{v}}}"] \arrow[dr, "(\cdot)^{\mathsf{v}}"'] & & {[t]^{\mathsf{v}} \neq [\![t^{\mathsf{v}}]\!] \in \mathcal{U}} \\
& t^{\mathsf{v}} \in {!\Lambda} \arrow[ur, "{[\![\cdot]\!]}"'] \arrow[u, phantom, "\not\circlearrowleft"] &
\end{tikzcd}$$

Conjecture: There still exists a relationship in CbV between $[t]^{\mathsf{v}}$ and $[\![t^{\mathsf{v}}]\!]$, but it should be more sophisticated than in CbN. Maybe we should use logical relations between the two.

Outline

Conclusions (and motivations)

1. The existence of two separate paradigms (CbN and CbV λ-calculi) is troubling:
   - it makes each language appear arbitrary (a unified language might be more canonical);
   - each time we create a new style of semantics (e.g. operational semantics, continuation semantics, Scott semantics, game semantics, etc.) we always need to do it twice.

2. The bang calculus is a general setting to compare CbN and CbV λ-calculi in the same rewriting system and with the same denotational semantics.
   - CbN λ-calculus has a rich and refined theory featuring advanced concepts such as separability, solvability, Böhm trees, classification of λ-theories, full-abstraction, *etc.*
   - This is not the case for CbV λ-calculus: in the CbV counterpart of these theoretical notions there are only partial and not satisfactory results (or do not exist at all!).
   - These theoretical notions and results well studied for the CbN λ-calculus might be adapted and studied in the more general setting of the bang calculus $\rightsquigarrow$ compelling point of view to analyze the corresponding notions for CbV λ-calculus.

Conclusions (and motivations)

1. The existence of two separate paradigms (CbN and CbV λ-calculi) is troubling:
   - it makes each language appear arbitrary (a unified language might be more canonical);
   - each time we create a new style of semantics (e.g. operational semantics, continuation semantics, Scott semantics, game semantics, etc.) we always need to do it twice.

2. The bang calculus is a general setting to compare CbN and CbV λ-calculi in the same rewriting system and with the same denotational semantics.
   - CbN λ-calculus has a rich and refined theory featuring advanced concepts such as separability, solvability, Böhm trees, classification of λ-theories, full-abstraction, *etc.*
   - This is not the case for CbV λ-calculus: in the CbV counterpart of these theoretical notions there are only partial and not satisfactory results (or do not exist at all!).
   - These theoretical notions and results well studied for the CbN λ-calculus might be adapted and studied in the more general setting of the bang calculus $\rightsquigarrow$ compelling point of view to analyze the corresponding notions for CbV λ-calculus.

Conclusions (and motivations)

1. The existence of two separate paradigms (CbN and CbV λ-calculi) is troubling:
   - it makes each language appear arbitrary (a unified language might be more canonical);
   - each time we create a new style of semantics (e.g. operational semantics, continuation semantics, Scott semantics, game semantics, etc.) we always need to do it twice.
2. The bang calculus is a general setting to compare CbN and CbV λ-calculi in the same rewriting system and with the same denotational semantics.
   - CbN λ-calculus has a rich and refined theory featuring advanced concepts such as separability, solvability, Böhm trees, classification of λ-theories, full-abstraction, *etc.*
   - This is not the case for CbV λ-calculus: in the CbV counterpart of these theoretical notions there are only partial and not satisfactory results (or do not exist at all!).
   - These theoretical notions and results well studied for the CbN λ-calculus might be adapted and studied in the more general setting of the bang calculus $\rightsquigarrow$ compelling point of view to analyze the corresponding notions for CbV λ-calculus.

1. Factorization theorem proved once and for all in the bang calculus:

$$T \to_{\mathsf{b}}^{*} S \implies T \to_{\mathsf{b_g}}^{*} \to_{\neg\mathsf{b_g}}^{*} S$$

$\rightsquigarrow$ By translation in CbN and CbV λ-calculi:

$$t \to_{\beta}^{*} s \implies t \to_{\beta_{\mathbf{g}}}^{*} \to_{\neg\beta_{\mathbf{g}}}^{*} s \qquad\qquad t \to_{\beta^{v}}^{*} s \implies t \to_{\beta_{\mathbf{g}}^{v}}^{*} \to_{\neg\beta_{\mathbf{g}}^{v}}^{*} s$$

(Faggian, G., FoSSaCS 2021)

2. Normalization theorem proved once and for all in the bang calculus:

$$T \to_{\mathsf{b}}^{*} S \text{ with } S \text{ normal} \implies T \to_{\ell\ell}^{*} S$$

$\rightsquigarrow$ By translation in the CbN and CbV λ-calculi:

$$t \to_{\beta}^{*} s \text{ with } s \text{ normal} \implies t \to_{\ell\ell}^{*} s \qquad\qquad t \to_{\beta^{v}}^{*} s \text{ with } s \text{ normal} \implies t \to_{\ell\ell}^{*} s$$

(Faggian, G., FoSSaCS 2021)

3. Characterization of ground normalization proved once and for all in bang calculus:

$$[\![T]\!]_{\vec{x}} \text{ is non-trivial} \iff T \text{ is } \mathrm{b_g}\text{-normalizable}$$

$\rightsquigarrow$ By translation in CbN and CbV λ-calculi:

$$[t]^{\mathsf{n}}_{\vec{x}} \text{ is non-trivial} \iff t \text{ is } \beta_{\mathrm{g}}\text{-normalizable}$$
$$[t]^{\mathsf{v}}_{\vec{x}} \text{ is non-trivial} \iff t \text{ is } \beta^{v}_{\mathrm{g}}\text{-normalizable}$$

(Bucciarelli, Kesner, Rios, Viso, FLOPS 2020)

4. Denotational semantics of the bang calculus via intersection distributors.
$\rightsquigarrow$ bi-categorical setting for a proof-relevant semantics.

$$[\![T]\!]_{\vec{x}} = \left\{ \widetilde{\pi} \;\middle|\; \begin{array}{c} \vdots\, \pi \\ \Gamma \vdash T : a \end{array} \right\}$$

(G., Olimpieri, CSL 2021)

3. Characterization of ground normalization proved once and for all in bang calculus:

$$[\![T]\!]_{\vec{x}} \text{ is non-trivial } \iff T \text{ is } \mathrm{b_g}\text{-normalizable}$$

$\rightsquigarrow$ By translation in CbN and CbV λ-calculi:

$$[t]^{\mathsf{n}}_{\vec{x}} \text{ is non-trivial } \iff t \text{ is } \beta_{\mathrm{g}}\text{-normalizable}$$
$$[t]^{\mathsf{v}}_{\vec{x}} \text{ is non-trivial } \iff t \text{ is } \beta^{v}_{\mathrm{g}}\text{-normalizable}$$

(Bucciarelli, Kesner, Rios, Viso, FLOPS 2020)

4. Denotational semantics of the bang calculus via intersection distributors.
$\rightsquigarrow$ bi-categorical setting for a proof-relevant semantics.

$$[\![T]\!]_{\vec{x}} = \left\{ \widetilde{\pi} \;\middle|\; \begin{array}{c} \vdots\, \pi \\ \Gamma \vdash T : a \end{array} \right\}$$

(G., Olimpieri, CSL 2021)

In non-idempotent intersection type systems for λ-calculi, typability is undecidable.

Question: Is the inabitation problem decidable in the bang calculus?

Given an typing context Γ and a multi type M, is there a term t such that

$$\Gamma \vdash t : M \quad \text{is derivable?}$$

Question bis: Same question, but in the CbN and CbV λ-calculi.

Answer [ArrKesGue23]: Yes, it is decidable an we can find all the inhabitants!
And if we restrict the search space of our algorithm to:

- the CbN fragment of the bang calculus $\rightsquigarrow$ we decide inhabitation in CbN;
- the CbV fragment of the bang calculus $\rightsquigarrow$ we decide inhabitation in CbV.

In non-idempotent intersection type systems for λ-calculi, typability is undecidable.

Question: Is the inabitation problem decidable in the bang calculus?

Given an typing context Γ and a multi type M, is there a term t such that

$$\Gamma \vdash t : M \quad \text{is derivable?}$$

Question bis: Same question, but in the CbN and CbV λ-calculi.

Answer [ArrKesGue23]: Yes, it is decidable an we can find all the inhabitants!
And if we restrict the search space of our algorithm to:

- the CbN fragment of the bang calculus $\rightsquigarrow$ we decide inhabitation in CbN;
- the CbV fragment of the bang calculus $\rightsquigarrow$ we decide inhabitation in CbV.

In non-idempotent intersection type systems for λ-calculi, typability is undecidable.

Question: Is the inabitation problem decidable in the bang calculus?

Given an typing context Γ and a multi type M, is there a term t such that

$$\Gamma \vdash t : M \quad \text{is derivable?}$$

Question bis: Same question, but in the CbN and CbV λ-calculi.

Answer [ArrKesGue23]: Yes, it is decidable an we can find all the inhabitants!
And if we restrict the search space of our algorithm to:

- the CbN fragment of the bang calculus $\rightsquigarrow$ we decide inhabitation in CbN;
- the CbV fragment of the bang calculus $\rightsquigarrow$ we decide inhabitation in CbV.

The big open question

Call-by-Need is another evaluation mechanism (e.g., used by Haskell):

- as smart as CbV for duplication,
- as smart as CbN for erasure.

Question: What about Call-by-Need? Can we use LL to understand Call-by-Need?

Question bis: Is there a general framework subsuming CbV, CbN and CbNeed?

Idea: We should split the ! comonad into two:

- one for duplication;
- one for erasure.

The big open question

Call-by-Need is another evaluation mechanism (e.g., used by Haskell):

- as smart as CbV for duplication,
- as smart as CbN for erasure.

Question: What about Call-by-Need? Can we use LL to understand Call-by-Need?

Question bis: Is there a general framework subsuming CbV, CbN and CbNeed?

Idea: We should split the ! comonad into two:

- one for duplication;
- one for erasure.

The big open question

Call-by-Need is another evaluation mechanism (e.g., used by Haskell):

- as smart as CbV for duplication,
- as smart as CbN for erasure.

Question: What about Call-by-Need? Can we use LL to understand Call-by-Need?

Question bis: Is there a general framework subsuming CbV, CbN and CbNeed?

Idea: We should split the ! comonad into two:

- one for duplication;
- one for erasure.

Thank you!

Questions?