



Semántica dinámica de cálculos de sustituciones explícitas a distancia

Pablo Barenbaum

Defensa de tesis doctoral

Directores

Eduardo Bonelli

Stevens Institute of Technology

Delia Kesner

IRIF, Université de Paris; Institut Universitaire de France

Jurado

Nazareno Aguirre

Universidad Nacional de Río Cuarto; CONICET

Zena Ariola

University of Oregon

Alexandre Miquel

IMERL, Fing, Universidad de la República

Outline

Introduction

Strong call-by-need

Lévy labels

Conclusion

*[...] if their use has not become general for large numerical calculations, it is because they have not in fact resolved the double problem which the question presents, that of **correctness in the results**, united with **economy of time**.*

—Ada Lovelace, on the Analytical Engine (**1843**)



*[...] if their use has not become general for large numerical calculations, it is because they have not in fact resolved the double problem which the question presents, that of **correctness in the results**, united with **economy of time**.*



—Ada Lovelace, on the Analytical Engine (1843)

State of the art (2020)

Writing **correct** and **efficient** software is still a difficult problem.

*[...] if their use has not become general for large numerical calculations, it is because they have not in fact resolved the double problem which the question presents, that of **correctness in the results**, united with **economy of time**.*



—Ada Lovelace, on the Analytical Engine (1843)

State of the art (2020)

Writing **correct** and **efficient** software is still a difficult problem.

Ultimate goal

Write declarative programs, execute them efficiently.

*[...] if their use has not become general for large numerical calculations, it is because they have not in fact resolved the double problem which the question presents, that of **correctness in the results**, united with **economy of time**.*



—Ada Lovelace, on the Analytical Engine (1843)

State of the art (2020)

Writing **correct** and **efficient** software is still a difficult problem.

Ultimate goal

Write declarative programs, execute them efficiently.

Techniques used in this thesis

Abstract machines, rewriting theory, type theory, ...

Evaluation strategies — ways to evaluate expressions

$$f(x) = x * x$$

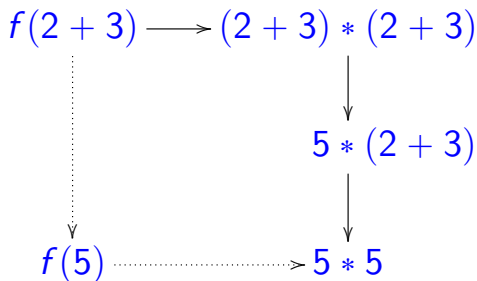
$$\begin{array}{l} f(2 + 3) \longrightarrow (2 + 3) * (2 + 3) \\ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \downarrow \\ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad 5 * (2 + 3) \\ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \downarrow \\ \qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad 5 * 5 \end{array}$$

→ **Call-by-name:** Apply the function first.

Duplication (X)

Evaluation strategies — ways to evaluate expressions

$$f(x) = x * x$$



- > **Call-by-name:** Apply the function first. **Duplication (X)**
-> **Call-by-value:** Evaluate the argument first. **No duplication (✓)**

Evaluation strategies — ways to evaluate expressions

$$g(x) = 0$$

$$g(2 + 3)$$

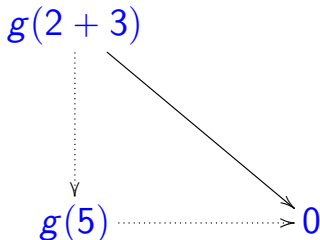


.....> **Call-by-value:** Evaluate the argument first.

Unneeded (X)

Evaluation strategies — ways to evaluate expressions

$$g(x) = 0$$



.....> **Call-by-value:** Evaluate the argument first.

——> **Call-by-name:** Apply the function first.

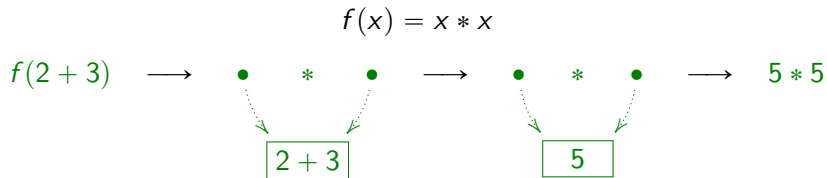
Unneeded (X)

Needed (✓)

Evaluation strategies — ways to evaluate expressions

Call-by-need

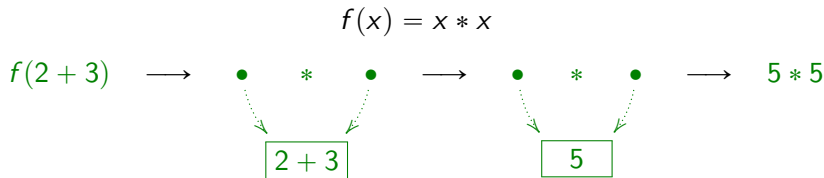
(Wadsworth, 1971)



Evaluation strategies — ways to evaluate expressions

Call-by-need

(Wadsworth, 1971)



Lazy (no unneeded work). (✓)

Complete with respect to call-by-name. (Ariola et al., 1995)

Sharing (no duplication of work). (✓)

Optimal for weak reduction. (Balabonski, 2013)

Syntax

Terms	t, s, \dots	$::=$	x	variable
			$ \ \lambda x.t$	lambda abstraction
			$ \ ts$	application

Semantics

$$(\lambda x.t) s \rightarrow_{\beta} t\{x := s\}$$

Syntax

Terms	t, s, \dots	$::=$	x	variable
			$ \ \lambda x.t$	lambda abstraction
			$ \ ts$	application

Semantics

$$(\lambda x.t) s \rightarrow_{\beta} t\{x := s\}$$

Example

Erasure $(\lambda x.0)(yz) \rightarrow_{\beta} 0$

Syntax

Terms	t, s, \dots	$::=$	x	variable
			$ \ \lambda x.t$	lambda abstraction
			$ \ ts$	application

Semantics

$$(\lambda x.t) s \rightarrow_{\beta} t\{x := s\}$$

Example

Erase	$(\lambda x.0)(yz)$	\rightarrow_{β}	0
Duplication	$(\lambda x.xx)(yz)$	\rightarrow_{β}	$(yz)(yz)$

Syntax

Terms	t, s, \dots	$::=$	x	variable
			$ \ \lambda x.t$	lambda abstraction
			$ \ ts$	application

Semantics

$$(\lambda x.t) s \rightarrow_{\beta} t\{x := s\}$$

Example

Erasure	$(\lambda x.0)(yz)$	\rightarrow_{β}	0
Duplication	$(\lambda x.xx)(yz)$	\rightarrow_{β}	$(yz)(yz)$
Non-termination	$(\lambda x.xx)(\lambda x.xx)$	\rightarrow_{β}	$(\lambda x.xx)(\lambda x.xx)$

Syntax

Terms	t, s, \dots	$::=$	x	variable
			$ \ \lambda x.t$	lambda abstraction
			$ \ t s$	application

Semantics

$$(\lambda x.t) s \rightarrow_{\beta} t\{x := s\}$$

Example

Erasure	$(\lambda x.0)(y z)$	\rightarrow_{β}	0
Duplication	$(\lambda x.x x)(y z)$	\rightarrow_{β}	$(y z) (y z)$
Non-termination	$(\lambda x.x x)(\lambda x.x x)$	\rightarrow_{β}	$(\lambda x.x x)(\lambda x.x x)$

Limitation — It cannot express sharing.

Linear Substitution Calculus (LSC) (Accattoli & Kesner, 2010)

Syntax

Terms	t, s, \dots	$::=$	x	variable
			$\lambda x.t$	lambda abstraction
			ts	application
			$t[x/s]$	explicit substitution
List contexts	L	$::=$	$[x_1/t_1] \dots [x_n/t_n]$	
Term contexts	C	$::=$	$\square \mid \lambda x.C \mid Ct \mid tC \mid C[x/t] \mid t[x/C]$	

Semantics

$$\begin{aligned}(\lambda x.t)Ls &\rightarrow_{\text{db}} t[x/s]L \\ C\langle x \rangle[x/t] &\rightarrow_{\text{ls}} C\langle t \rangle[x/t] \\ t[x/s] &\rightarrow_{\text{gc}} t \quad \text{if } x \notin \text{fv}(t)\end{aligned}$$

Rules are *at a distance*. Justified by linear logic proof nets (✓)

Linear Substitution Calculus (LSC) (Accattoli & Kesner, 2010)

Syntax

Terms	t, s, \dots	$::=$	x	variable
			$\lambda x.t$	lambda abstraction
			ts	application
			$t[x/s]$	explicit substitution
List contexts	L	$::=$	$[x_1/t_1] \dots [x_n/t_n]$	
Term contexts	C	$::=$	$\square \mid \lambda x.C \mid Ct \mid tC \mid C[x/t] \mid t[x/C]$	

Semantics

$$\begin{aligned}(\lambda x.t)Ls &\rightarrow_{\text{db}} t[x/s]L \\ C\langle x \rangle[x/t] &\rightarrow_{\text{ls}} C\langle t \rangle[x/t] \\ t[x/s] &\rightarrow_{\text{gc}} t \quad \text{if } x \notin \text{fv}(t)\end{aligned}$$

Rules are *at a distance*. Justified by linear logic proof nets (✓)

Example

$$\begin{aligned}(\lambda x.xx)(yz) &\rightarrow_{\text{db}} (xx)[x/yz] \\ &\rightarrow_{\text{ls}} ((yz)x)[x/yz] \\ &\rightarrow_{\text{ls}} ((yz)(yz))[x/yz] \\ &\rightarrow_{\text{gc}} (yz)(yz)\end{aligned}$$

This thesis

Overarching theme

Evaluation strategies in the **Linear Substitution Calculus**.

Three lines of work

1. **Abstract machines** (with Accattoli, Mazza)
2. **Strong call-by-need** (with Balabonski, Bonelli, Kesner)
+ Pattern matching and fixed points (with Bonelli, Mohamed)
3. **Lévy labels** (with Bonelli)

This thesis

Overarching theme

Evaluation strategies in the **Linear Substitution Calculus**.

Three lines of work

1. **Abstract machines*** (with Accattoli, Mazza)
2. **Strong call-by-need** (with Balabonski, Bonelli, Kesner)
+ Pattern matching and fixed points* (with Bonelli, Mohamed)
3. **Lévy labels** (with Bonelli)

* Not included in this defense.

Outline

Introduction

Strong call-by-need

Lévy labels

Conclusion

Weak vs. strong reduction

Typical programming languages use **weak** reduction:

- ▶ Bodies of functions are not evaluated (until applied).
- ▶ Programs are *closed* (no free variables).
- ▶ The result is a *weak head normal form*.

Example

$\lambda x. (2 + 3) * \text{id } x$ is already a weak head normal form.

Weak vs. **strong** reduction

Contrast with **strong** reduction:

- ▶ Bodies of functions must be evaluated.
- ▶ Programs may be *open*.
- ▶ The result is a (*strong*) *normal form*.

Example

$\lambda x. (2 + 3) * \text{id } x \longrightarrow \lambda x. 5 * \text{id } x \longrightarrow \lambda x. 5 * x$

Motivation to study strong reduction

Proof assistants based on dependent type theory (**Coq**, **Agda**, ...) include the following typing rule:

$$\frac{\Gamma \vdash t : A \quad A \equiv B}{\Gamma \vdash t : B}$$

- ▶ To decide whether $A \equiv B$, compare their normal forms.
- ▶ This requires **strong** reduction, as types may depend on terms.

Weak call-by-need

Weak call-by-need reduction

(Accattoli, Barenbaum, Mazza, 2014)

Values $v ::= \lambda x.t$
Weak evaluation contexts $E ::= \square \mid E t \mid E[x/t] \mid E\langle x \rangle[x/E]$

$$\begin{array}{l} (\lambda x.t)L s \xrightarrow{W} t[x/s]L \\ E\langle x \rangle[x/v]L \xrightarrow{W} E\langle v \rangle[x/v]L \end{array}$$

Similar to (Ariola et al., 1995), but with rules *at a distance*.

Our goal

Extend **weak** call-by-need ($\overset{W}{\rightsquigarrow}$) to **strong** call-by-need ($\overset{S}{\rightsquigarrow}$).

Technical challenges

Context-dependency

$\lambda x.t$ (no arguments) \implies evaluate t .

$(\lambda x.t) u$ (with arguments) \implies do not evaluate t yet.

Production $E ::= \dots \mid \lambda x.E$ is too naive.

Technical challenges

Context-dependency

$\lambda x.t$ (no arguments) \implies evaluate t .

$(\lambda x.t) u$ (with arguments) \implies do not evaluate t yet.

Production $E ::= \dots \mid \lambda x.E$ is too naive.

Frozen variables

$\lambda x.x t$ (x is frozen) \implies evaluate t .

$(x t)[x/\lambda y.y]$ (x is not frozen) \implies do not evaluate t yet.

Production $E ::= \dots \mid x E$ is too naive.

Strong call-by-need strategy

Normal forms under frozen variables ϑ :

$$\begin{aligned} N_{\vartheta} &::= S_{\vartheta} & | & \lambda x. N_{\vartheta \cup \{x\}} & | & \underbrace{N_{\vartheta \cup \{x\}}[x/S_{\vartheta}]}_{x \in \text{ngv}(N_{\vartheta})} & | & \underbrace{N_{\vartheta}[x/t]}_{x \notin \text{ngv}(N_{\vartheta})} \\ S_{\vartheta} &::= \underbrace{x}_{x \in \vartheta} & | & S_{\vartheta} N_{\vartheta} & | & \underbrace{S_{\vartheta \cup \{x\}}[x/S'_{\vartheta}]}_{x \in \text{ngv}(S_{\vartheta})} & | & \underbrace{S_{\vartheta}[x/t]}_{x \notin \text{ngv}(S_{\vartheta})} \end{aligned}$$

Evaluation contexts under frozen variables ϑ :

$$\begin{aligned} E_{\vartheta} &::= E_{\vartheta}^{\circ} & | & \lambda x. E_{\vartheta \cup \{x\}} \\ & & | & \underbrace{E_{\vartheta}[x/t]}_{x \notin \vartheta, t \notin S_{\vartheta}} & | & E_{\vartheta \cup \{x\}}[x/S_{\vartheta}] & | & E_{\vartheta}\langle x \rangle[x/E_{\vartheta}^{\circ}] \\ E_{\vartheta}^{\circ} &::= \square & | & E_{\vartheta}^{\circ} t & | & S_{\vartheta} E_{\vartheta} \\ & & | & \underbrace{E_{\vartheta}^{\circ}[x/t]}_{x \notin \vartheta, t \notin S_{\vartheta}} & | & E_{\vartheta \cup \{x\}}^{\circ}[x/S_{\vartheta}] & | & E_{\vartheta}^{\circ}\langle x \rangle[x/E_{\vartheta}^{\circ}] \end{aligned}$$

Strong call-by-need strategy

Reduction rules

$$\begin{array}{l} \mathbf{C}\langle(\lambda x.t)\mathbf{L} s\rangle \xrightarrow{\vartheta} \mathbf{C}\langle t[x/s]\mathbf{L}\rangle \\ \mathbf{C}_1\langle\mathbf{C}_2\langle x\rangle[x/v]\mathbf{L}\rangle \xrightarrow{\vartheta} \mathbf{C}_1\langle\mathbf{C}_2\langle v\rangle[x/v]\mathbf{L}\rangle \end{array}$$

If the **context** is a strong call-by-need ϑ -evaluation context.

I.e., \mathbf{C} and $\mathbf{C}_1\langle\mathbf{C}_2\langle \square \rangle[x/v]\mathbf{L}\rangle$ are generated by the non-terminal symbol \mathbf{E}_ϑ .

Strong call-by-need strategy

Properties

1. **Strong reduction** if $t \in \text{NF}(\overset{S}{\rightsquigarrow})$ then $t^\diamond \in \text{NF}(\rightarrow_\beta)$.
2. **Determinism** $t \overset{S}{\rightsquigarrow} s_1$ and $t \overset{S}{\rightsquigarrow} s_2$ implies $s_1 = s_2$.
3. **Conservativity** $\overset{W}{\rightsquigarrow} \subseteq \overset{S}{\rightsquigarrow}$.
4. **Soundness** If $t \overset{S}{\rightsquigarrow} s$ then $t^\diamond =_\beta s^\diamond$.
5. **Completeness** If $t =_\beta s$ and s is a β -normal form, then $t \overset{S}{\rightsquigarrow}^* u$ with $u^\diamond = s$.

Strong call-by-need strategy

Properties

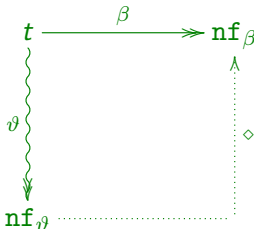
1. **Strong reduction** if $t \in \text{NF}(\overset{S}{\rightsquigarrow})$ then $t^\diamond \in \text{NF}(\rightarrow_\beta)$.
2. **Determinism** $t \overset{S}{\rightsquigarrow} s_1$ and $t \overset{S}{\rightsquigarrow} s_2$ implies $s_1 = s_2$.
3. **Conservativity** $\overset{W}{\rightsquigarrow} \subseteq \overset{S}{\rightsquigarrow}$.
4. **Soundness** If $t \overset{S}{\rightsquigarrow} s$ then $t^\diamond =_\beta s^\diamond$.
5. **Completeness** If $t =_\beta s$ and s is a β -normal form, then $t \overset{S}{\rightsquigarrow}^* u$ with $u^\diamond = s$.
Most interesting/difficult one.
We discuss the proof next.

For example, $(\lambda x. \lambda y. x) a b \overset{S}{\rightsquigarrow}^* x[y/b][x/a]$ and $x[y/b][x/a]^\diamond = a$.

Proof of Completeness of strong call-by-need

Completeness of strong call-by-need

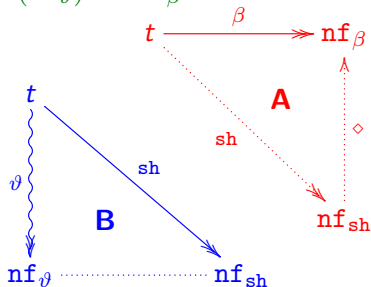
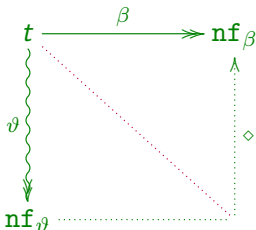
If $t \rightarrow_{\beta}^* \mathbf{nf}_{\beta}$ then $t \rightsquigarrow_{\vartheta}^* \mathbf{nf}_{\vartheta}$ and $(\mathbf{nf}_{\vartheta})^{\diamond} = \mathbf{nf}_{\beta}$.



Proof of Completeness of strong call-by-need

Completeness of strong call-by-need

If $t \rightarrow_{\beta}^* \text{nf}_{\beta}$ then $t \xrightarrow{\vartheta}^* \text{nf}_{\vartheta}$ and $(\text{nf}_{\vartheta})^{\diamond} = \text{nf}_{\beta}$.



A: Completeness of the Theory of Sharing

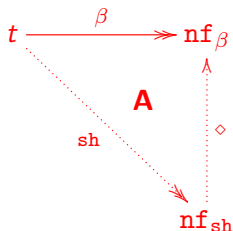
If $t \rightarrow_{\beta}^* \text{nf}_{\beta}$ then $t \rightarrow_{\text{sh}}^* \text{nf}_{\text{sh}}$ and $\text{nf}_{\text{sh}}^{\diamond} = \text{nf}_{\beta}$.

B: Factorization of the Theory of Sharing

If $t \rightarrow_{\text{sh}}^* \text{nf}_{\text{sh}}$ then $t \xrightarrow{\vartheta}^* \text{nf}_{\vartheta}$ and $\text{nf}_{\text{sh}}^{\diamond} = \text{nf}_{\vartheta}^{\diamond}$.

Proof of **A**: Completeness of the Theory of Sharing

If $t \rightarrow_{\beta}^* \text{nf}_{\beta}$ then $t \rightarrow_{\text{sh}}^* \text{nf}_{\text{sh}}$ and $\text{nf}_{\text{sh}}^{\diamond} = \text{nf}_{\beta}$.



An argument based on *non-idempotent intersection types*.
Extending a **similar argument for weak call-by-need**.

(Kesner, 2016)

Proof of **A**: Completeness of the Theory of Sharing

The proof passes through the type system \mathcal{HW} .

(Kesner & Ventura, 2014)

- ▶ \mathcal{HW} is an non-idempotent intersection type system.

(Coppo & Dezani-Ciancaglini, 1978)

(Gardner, 1994; Kfoury, 2004; de Carvalho, 2007)

- ▶ Intersection type systems characterize **normalization properties**.

Proof of **A**: Completeness of the Theory of Sharing

The proof passes through the type system \mathcal{HW} .

(Kesner & Ventura, 2014)

- ▶ \mathcal{HW} is an **non-idempotent intersection** type system.

(Coppo & Dezani-Ciancaglini, 1978)

(Gardner, 1994; Kfoury, 2004; de Carvalho, 2007)

- ▶ Intersection type systems characterize **normalization properties**.

t is weakly normalizing with respect to \rightarrow_{β}



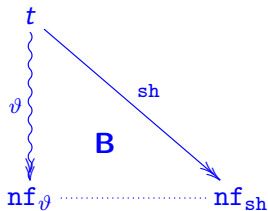
$\Gamma \vdash t : \tau$ in system \mathcal{HW} , without positive occurrences of $[]$



t is weakly normalizing with respect to \rightarrow_{sh}

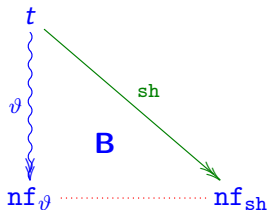
Proof of **B**: Factorization of the Theory of Sharing

If $t \rightarrow_{\text{sh}}^* \text{nf}_{\text{sh}}$ then $t \xrightarrow{\vartheta}^* \text{nf}_{\vartheta}$ and $\text{nf}_{\text{sh}}^{\diamond} = \text{nf}_{\vartheta}^{\diamond}$.



Proof of **B**: Factorization of the Theory of Sharing

If $t \xrightarrow{*}_{\text{sh}} \text{nf}_{\text{sh}}$ then $t \xrightarrow{\vartheta} \text{nf}_{\vartheta}$ and $\text{nf}_{\text{sh}}^{\diamond} = \text{nf}_{\vartheta}^{\diamond}$.



Core of the argument: **internal/external** commutation.

$$\xrightarrow{\text{sh}} = \xrightarrow{\vartheta} \uplus \xrightarrow{\neg\vartheta}_{\text{sh}}$$

We prove $(\xrightarrow{\neg\vartheta}_{\text{sh}} \xrightarrow{\vartheta}) \subseteq (\xrightarrow{\vartheta} \xrightarrow{*}_{\text{sh}})$ by exhaustive case analysis.
Very long/intricate proof.

It relies on an **abstract factorization result**. (Accattoli, 2012).

Outline

Introduction

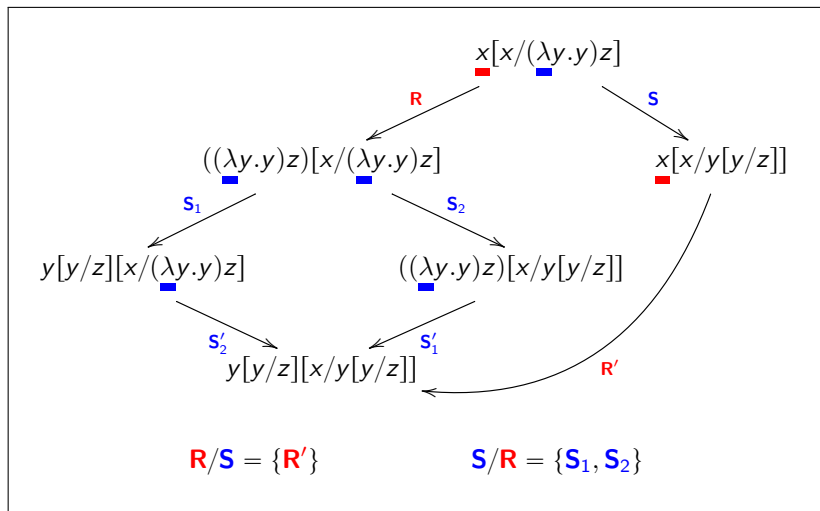
Strong call-by-need

Lévy labels

Conclusion

Residuals

S/R denotes the set of **residuals** of S after R



Residuals

R creates **S'** if there is no **S** such that $S' \in S/R$.

$$\underline{\lambda x}.xx(\lambda y.y) \xrightarrow{R} \underline{xx}[x/\lambda y.y] \xrightarrow{S} ((\lambda y.y)x)[x/\lambda y.y]$$

R creates **S**

Our goal

Develop the residual theory of reduction in the LSC.

Adapt the notions of **redex families**, **Lévy labels**, **FFD**, **extraction**, **optimality**, etc..

(Vuillemin, Lévy, Lamping, Laneve,
van Oostrom, Asperti, Guerrini,
Glauert, Khasidashvili, ...)

Motivations

- ▶ **Conceptual** Causality in the presence of sharing.
- ▶ **Pragmatical** Build technology to prove theorems.

Our goal

Develop the residual theory of reduction in the LSC.

Adapt the notions of **redex families**, **Lévy labels**, **FFD**, **extraction**, **optimality**, etc..

(Vuillemin, Lévy, Lamping, Laneve,
van Oostrom, Asperti, Guerrini,
Glauert, Khasidashvili, ...)

Motivations

- ▶ **Conceptual** Causality in the presence of sharing.
- ▶ **Pragmatical** Build technology to prove theorems.

Two parts

- ▶ **The LSC with Lévy labels.**
- ▶ **Applications of the LSC with Lévy labels.**

The LSC with Lévy labels (LLSC)

Syntax

Labeled terms $t ::= x^\alpha \mid \lambda^\alpha x.t \mid @^\alpha(t, t) \mid t[x/t]$
Labels $\alpha ::= \bullet \mid \mathbf{a} \mid [\alpha] \mid \lfloor \alpha \rfloor \mid \mathbf{db}(\alpha) \mid \alpha\alpha$

Operations

$\uparrow(t)$ denotes the outermost sublabel of t .
 $\downarrow(\alpha)$ denotes the innermost sublabel of α .
 $\alpha : t$ adds a label α to a term t .

Reduction rules

$$@^\alpha((\lambda^\beta x.t)L, s) \xrightarrow{\mathbf{db}(\beta)} \alpha[\mathbf{db}(\beta)] : t[x/[\mathbf{db}(\beta)] : s]L$$

$$C\langle x^\alpha \rangle[x/t] \xrightarrow{\downarrow(\alpha) \bullet \uparrow(t)} C\langle \alpha \bullet : t \rangle[x/t]$$

Properties of LLSC

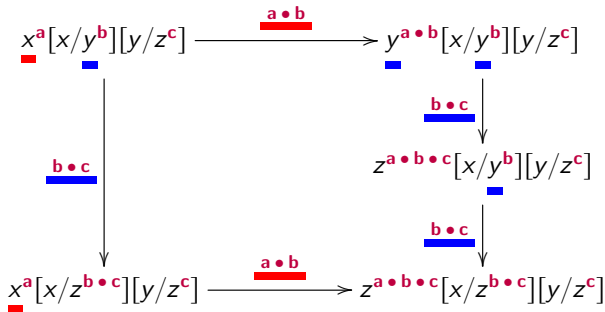
Confluence

$$(\rightarrow_{\ell}^*)^{-1} \rightarrow_{\ell}^* \subseteq \rightarrow_{\ell}^* (\rightarrow_{\ell}^*)^{-1}$$

Copy

If $S' \in S/R$ then S and S' have the same name.

Example



Properties of LLSC

Creation

If **R** creates **S**, the name of **R** is a sublabel of the name of **S**.

Example (1s creates db)

$$\textcircled{a}(x^{\underline{b}}, t)[x/\lambda^c y.z^d]$$

$$\xrightarrow{\underline{b \bullet c}} \textcircled{a}((\lambda^{\underline{b \bullet c}} y.z^d), t)[x/\lambda^c y.z^d]$$

$$\xrightarrow{\underline{\text{db}(b \bullet c)}} z^{a[\text{db}(b \bullet c)]^d}[y/[\text{db}(b \bullet c)] : t][x/\lambda^c y.z^d]$$

Properties of LLSC

Finite Family Developments

Reduction in the LLSC is **strongly normalizing** if restricted to names of bounded height.

Properties of LLSC

Contribution

Let M and N be names of steps. The following are equivalent:

1. **Name contribution.** (Concrete/syntactic notion)
 M is a sublabel of N .
2. **History contribution.** (Abstract/semantic notion)
For every sequence $R_1 \dots R_n$ such that $\text{name}(R_n) = N$
there exists an $i \leq n$ such that $\text{name}(R_i) = M$.

The proof relies on the **Finite Family Developments** property.

Properties of LLSC

The properties we just mentioned:

1. **Confluence**
2. **Copy**
3. **Creation**
4. **Finite Family Developments**
5. **Contribution**

make the LSC a **Deterministic Family Structure (DFS)**.

(Glauert & Khasidashvili, 1996)

Applications (1)

A **family reduction** $\mathcal{M}_1 \dots \mathcal{M}_n$ to normal form is **optimal** if there is no shorter family reduction to normal form.

Optimal reduction for the LSC

Any **complete** and **needed** family reduction $\mathcal{M}_1 \dots \mathcal{M}_n$ to normal form is optimal.

A corollary of Glauert & Khasidashvili, 1996.

Itself extending work by Lévy (1978).

Applications (2)

Two reduction sequences are **permutation equivalent** if they perform the same computational work (swapping steps).

Standardization for the LSC

For each reduction ρ , we can compute a canonical representative $\mathbb{M}(\rho)$ of its permutation equivalence class.

Generalizes a theorem by Klop (1980) to DFSs.

Canonical representatives are known as **standard reductions**.

Applications (3)

A deterministic reduction strategy \mathbb{S} is:

- ▶ **invariant** if steps in \mathbb{S} have residuals in \mathbb{S} ,
- ▶ **strongly invariant** if moreover $\text{NF}(\mathbb{S})$ is stable by reduction,
- ▶ **normalizing** if given a term with normal form, \mathbb{S} finds it.

Normalization for the LSC

Any **strongly invariant** strategy in the LSC is normalizing.
In particular, call-by-name and a variant of weak call-by-need are normalizing.

Outline

Introduction

Strong call-by-need

Lévy labels

Conclusion

Contributions

1. Abstract machines*

Evaluation strategies in the LSC **distill abstract machines**.

They are **reasonable** in terms of time complexity.

(ICFP'14, APLAS'15)

2. Strong call-by-need

We designed a **strong call-by-need strategy**. The main result is **completeness**.

(ICFP'17)

+ Pattern matching and fixed points*

We extended these results to allow **recursion** and **pattern matching**.

(PPDP'18)

3. Lévy labels

We endowed the LSC with **Lévy labels**. We applied it to obtain **optimality**, **standardization**, and **normalization** results.

(FSCD'17)

* Not included in this defense.

Conclusions

Explicit substitutions are adequate to study strategies with subterm sharing.

Conclusions

Explicit substitutions are adequate to study strategies with subterm sharing.

The **efficient implementation of strong reduction** is still poorly understood.

Conclusions

Explicit substitutions are adequate to study strategies with subterm sharing.

The **efficient implementation of strong reduction** is still poorly understood.

Some of the **technology developed in the 1970s/1980s** remains underexplored:

- ▶ Labeled λ -calculus.
- ▶ Finite Family Developments.
- ▶ Optimal reduction.

Future work

- ▶ Use Lévy labels to capture **dynamic properties** of programs.
E.g.: information flow, measuring partial evaluation.
Related: [Blanc's PhD thesis, 2006](#)

Future work

- ▶ Use Lévy labels to capture **dynamic properties** of programs.
E.g.: information flow, measuring partial evaluation.
Related: Blanc's PhD thesis, 2006
- ▶ Design a **reasonable** strong call-by-need strategy/machine.
Related: Biernacka & Charatonik, 2019

Future work

- ▶ Use Lévy labels to capture **dynamic properties** of programs.
E.g.: information flow, measuring partial evaluation.
Related: Blanc's PhD thesis, 2006
- ▶ Design a **reasonable** strong call-by-need strategy/machine.
Related: Biernacka & Charatonik, 2019
- ▶ Characterize *redex families* by means of **extraction**.

