Optimal reduction
in the
Linear Substitution Calculus

(work in progress)

Pablo Barenbaum[1]
joint work with Eduardo Bonelli[2]

[1]Universidad de Buenos Aires / CONICET
[2]Universidad Nacional de Quilmes / CONICET

Optimality

Motivation (in the context of the λ-calculus)

- By the standardization theorem, the leftmost-outermost strategy is correct (normalizing).
- But normal-order evaluation is certainly not "optimal":

$$(\lambda x.x\,x)\,R \to R\,R \qquad \text{two copies of } R!$$

- Reducing needed internal redexes is not optimal:

$$(\lambda x.x\,I)\,(\lambda y.\Delta\,(y\,z)) \to \dots \qquad \text{where } \Delta := \lambda x.x\,x$$

- *Lazy evaluation*, introduced by Wadsworth in 1971, improves the situation by sharing the argument with pointers but is also not optimal:

$$(\lambda x.x\,y\,(x\,z))\,\lambda w.I\,w \to \dots$$

Optimality

Motivation (in the context of the λ-calculus)

- By the standardization theorem, the leftmost-outermost strategy is correct (normalizing).
- But normal-order evaluation is certainly not "optimal":

$$(\lambda x.x\,x)\,R \to R\,R \qquad \text{two copies of } R!$$

- Reducing needed internal redexes is not optimal:

$$(\lambda x.x\,I)\,(\lambda y.\Delta\,(y\,z)) \to \ldots \qquad \text{where } \Delta := \lambda x.x\,x$$

- *Lazy evaluation*, introduced by Wadsworth in 1971, improves the situation by sharing the argument with pointers but is also not optimal:

$$(\lambda x.x\,y\,(x\,z))\,\lambda w.I\,w \to \ldots$$

Some questions, studied by Lévy in his PhD thesis (1978)

What would it mean for a reduction to be optimal?

Are there optimal reduction strategies?

Can optimal reduction be implemented efficiently?

What would it mean for a reduction to be optimal?

Informally

- Avoid doing useless work $\Longrightarrow$ contract needed redexes only.
- Avoid duplicating work $\Longrightarrow$ share multiple copies of redexes.

Less informally

1. Characterize the notion of redex family.
   - Redexes in the same family are "copies" of the same redex.
   - *Example.* $\Delta\,(\lambda x.\Delta\,(x\,y)) \to \ldots$ with $\Delta := \lambda x.x\,x$.
   - *Non-examples.* redexes that happen to coincide: $z\,(I\,x)\,(I\,x)$; syntactic accidents: $I\,(I\,I) \to \ldots$.
   - We care about *redexes with history* ρR rather than redexes. **Why?**
   - Define a family equivalence relation: $\rho R \simeq \sigma S$.
   - Lévy gives three equivalent characterizations. We'll focus on this later.

2. Let $[\rho R]$ denote the family class of ρR:

$$[\rho R] \stackrel{\text{def}}{=} \{\sigma S \mid \rho R \simeq \sigma S\}$$

3. If $\rho = R_1 \dots R_n$ is a reduction sequence, let $\mathsf{FAM}(\rho)$ denote the families contracted along ρ:

$$\mathsf{FAM}(\rho) \stackrel{\text{def}}{=} \{[R_1 R_2 \dots R_i] \mid i \in \{1, \dots, n\}\}$$

4. A redex ρR is needed iff any extension $\rho\sigma$ to normal form contracts a residual of R.
5. A derivation $\rho = \mathcal{F}_1 \mathcal{F}_2 \dots \mathcal{F}_n$ is call-by-need iff each $\mathcal{F}_i$ contains at least one needed redex R_i.
6. A derivation $\rho = \mathcal{F}_1 \mathcal{F}_2 \dots \mathcal{F}_n$ is complete iff $\mathcal{F}_i \neq \emptyset$ and $\mathcal{F}_i$ is a maximal set of redexes such that:

$$\forall R, S \in \mathcal{F}_i. \qquad \mathcal{F}_1 \dots \mathcal{F}_{i-1} R \simeq \mathcal{F}_1 \dots \mathcal{F}_{i-1} S$$

7. Define $\mathbf{cost}(\rho)$ to measure the number of steps in a derivation, assigning unitary cost to the reduction of a set of shared copies.
8. **Theorem (Lévy)** Complete call-by-need derivations compute the normal form in optimal cost, *i.e.*:
   - if ρ is a complete call-by-need derivation: $\mathbf{cost}(\rho) = \#\mathsf{FAM}(\rho)$
   - if σ is a terminating reduction starting from the same term, $\mathbf{cost}(\sigma) \geq \#\mathsf{FAM}(\sigma) \geq \#\mathsf{FAM}(\rho) = \mathbf{cost}(\rho)$

Are there optimal reduction strategies?
Can optimal reduction be implemented efficiently?

Optimal reduction: *it comes with a free frogurt!*

☺ Lévy (1978): optimal derivations have optimal cost.

Are there optimal reduction strategies?
Can optimal reduction be implemented efficiently?

Optimal reduction: *it comes with a free frogurt!*

- ☺ Lévy (1978): optimal derivations have optimal cost.
- ☹ There cannot exist an optimal reduction *strategy*:

$$(\lambda x.x\, I\, x)\,(\lambda y.\Delta\,(y\, z)) \quad \text{with } \Delta := \lambda x.x\, x$$

Are there optimal reduction strategies?
Can optimal reduction be implemented efficiently?

Optimal reduction: *it comes with a free frogurt!*

- ☻ Lévy (1978): optimal derivations have optimal cost.
- ☹ There cannot exist an optimal reduction *strategy*:

$$(\lambda x.x\, I\, x)\,(\lambda y.\Delta\,(y\,z)) \quad \text{with } \Delta := \lambda x.x\,x$$

- ☻ Lamping's reduction algorithm (1989) based on sharing graphs implements optimal reduction.

Are there optimal reduction strategies?
Can optimal reduction be implemented efficiently?

Optimal reduction: *it comes with a free frogurt!*

☻ Lévy (1978): optimal derivations have optimal cost.

☹ There cannot exist an optimal reduction *strategy*:

$$(\lambda x.x\, I\, x)\,(\lambda y.\Delta\,(y\, z)) \quad \text{with } \Delta := \lambda x.x\, x$$

☻ Lamping's reduction algorithm (1989) based on sharing graphs implements optimal reduction.

??? Asperti and Mairson (1998) show that optimal reduction cannot be implemented efficiently: $n = \mathbf{cost}(\rho)$ parallel beta steps is not bounded by $O(2^n)$, $O(2^{2^n})$, $O(2^{2^{2^n}})$, etc.

Are there optimal reduction strategies?
Can optimal reduction be implemented efficiently?

Optimal reduction: *it comes with a free frogurt!*

☻ Lévy (1978): optimal derivations have optimal cost.

☹ There cannot exist an optimal reduction *strategy*:

$$(\lambda x.x\, I\, x)\,(\lambda y.\Delta\,(y\,z)) \quad \text{with } \Delta := \lambda x.x\,x$$

☻ Lamping's reduction algorithm (1989) based on sharing graphs implements optimal reduction.

??? Asperti and Mairson (1998) show that optimal reduction cannot be implemented efficiently: $n = \mathbf{cost}(\rho)$ parallel beta steps is not bounded by $O(2^n)$, $O(2^{2^n})$, $O(2^{2^{2^n}})$, etc.

??? Can a realistic notion of optimal reduction be devised?

Characterizing redex families

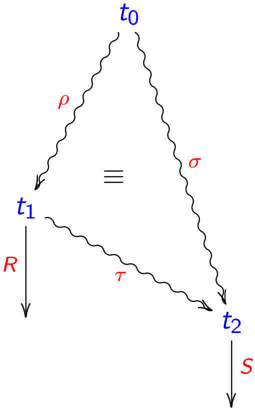
Lévy gives three equivalent characterizations of the family relation:

$$\rho R \simeq \sigma S$$

in the λ-calculus using various tools:

1. Zig-zag.
2. Extraction.
3. Labelling.

Families by zig-zag



Definition (copy).

$$\rho R \leq \sigma S$$

A redex with history σS is a *copy* of a redex with history ρR iff there is a derivation τ such that $\rho\tau \equiv \sigma$ and $S \in R/\tau$.

Definition (family).

$$\rho R \simeq \sigma S$$

The symmetric and transitive closure of $\leq$ defines the *family* relation.

Families by zig-zag: example

Let $I := \lambda x.x$ and $\Delta := \lambda x.x\,x$:

$$
\begin{tikzcd}
 & (\lambda x.x\,I)\,(\lambda y.\Delta\,(y\,z)) \arrow[dl, "\mathbf{A}"'] \arrow[dr, "\mathbf{D}"] & \\
(\lambda y.\Delta\,(y\,z))\,I \arrow[d, "\mathbf{B}"'] & & (\lambda x.x\,I)\,(\lambda y.y\,z\,(y\,z)) \arrow[d, "\mathbf{E}"] \\
\Delta\,(I\,z) \arrow[d, "\mathbf{C}"'] & & (\lambda y.y\,z\,(y\,z))\,I \arrow[d, "\mathbf{F}"] \\
\Delta\,z & & I\,z\,(I\,z) \arrow[d, "\mathbf{G}"] \\
 & & I\,z\,z
\end{tikzcd}
$$

$$\mathbf{A\,B\,C} \simeq \mathbf{D\,E\,F\,G} \quad ??$$

Families by zig-zag: example contd.

Let $I := \lambda x.x$ and $\Delta := \lambda x.x\,x$:

$$
\begin{tikzcd}
 & & (\lambda x.x\,I)\,(\lambda y.\Delta\,(y\,z)) \arrow[dll, "\mathbf{A}"'] \arrow[drr, "\mathbf{D}"] & & \\
(\lambda y.\Delta\,(y\,z))\,I \arrow[d, "\mathbf{B}"'] & & & & (\lambda x.x\,I)\,(\lambda y.y\,z\,(y\,z)) \arrow[d, "\mathbf{E}"] \\
\Delta\,(I\,z) \arrow[d, "\mathbf{C}"'] \arrow[drrrr, dotted, "\mathbf{H}"] & & \equiv & & (\lambda y.y\,z\,(y\,z))\,I \arrow[d, "\mathbf{F}"] \\
\Delta\,z & & & & I\,z\,(I\,z) \arrow[d, "\mathbf{G}"] \\
 & & & & I\,z\,z
\end{tikzcd}
$$

$$\mathbf{A\,B\,H} \equiv \mathbf{D\,E\,F} \qquad \mathbf{G} \in \mathbf{C}/\mathbf{H}$$

$$\Longrightarrow \ \mathbf{A\,B\,C} \leq \mathbf{D\,E\,F\,G} \ \Longrightarrow \ \mathbf{A\,B\,C} \simeq \mathbf{D\,E\,F\,G}$$

Families by extraction

Define a rewriting relation between reduction sequences, the extraction relation:

$$\rho R \;\rhd\; \sigma S$$

Informally. $\rhd$ erases the steps of ρ that do **not** contribute to the creation of the redex R.

Less informally. Let σ stand for any non-empty reduction sequence. Then $\rhd$ is defined by:

(1)	$\rho\, R\, S$	$\rhd$	$\rho\, S'$	if $S \in S'/R$
(2)	$\rho\,(R \sqcup \sigma)$	$\rhd$	$\rho\,\sigma$	if R and σ are disjoint
(3)	$\rho\,(R \sqcup \sigma)$	$\rhd$	$\rho\,\sigma$	if σ is internal to the function part of R
(4)	$\rho\, R\, \tau$	$\rhd$	$\rho\,\sigma$	if τ is internal to the i-th copy of the argument of R and σ is the corresponding reduction, internal to the argument of R*

* The formal statement requires quite more work ($\sigma/R = \tau || R$).

Families by extraction: properties

Theorem (Lévy). $\rhd$ is SN and CR.
Theorem (Lévy). $\rhd$ decides the family relation $\simeq$. More precisely, let ρ and σ be **standard** reductions. Then:

$$\rho R \simeq \sigma S \quad \text{iff} \quad \rho R \ \rhd^* \ \tau T \ \lhd^* \ \sigma S$$

for some $\rhd$-normal redex-with-history τT.

Note: to check $\rho R \simeq \sigma S$ in the general case, standardize ρ and σ first.

Families by extraction: example

Let $I := \lambda x.x$ and $\Delta := \lambda x.x\,x$:

$$
\begin{tikzcd}
 & (\lambda x.x\,I)\,(\lambda y.\Delta\,(y\,z)) \arrow[dl, "\mathbf{A}"'] \arrow[dr, "\mathbf{D}"] & \\
(\lambda y.\Delta\,(y\,z))\,I \arrow[d, "\mathbf{B}"'] & & (\lambda x.x\,I)\,(\lambda y.y\,z\,(y\,z)) \arrow[d, "\mathbf{E}"] \\
\Delta\,(I\,z) \arrow[d, "\mathbf{C}"'] & & (\lambda y.y\,z\,(y\,z))\,I \arrow[d, "\mathbf{F}"] \\
\Delta\,z & & I\,z\,(I\,z) \arrow[d, "\mathbf{G}"] \\
 & & I\,z\,z
\end{tikzcd}
$$

$\mathbf{A\,B\,C} \simeq \mathbf{D\,E\,F\,G}$??

Families by extraction: example contd.

Let $I := \lambda x.x$ and $\Delta := \lambda x.x\,x$:

$$
\begin{tikzcd}
 & (\lambda x.x\,I)\,(\lambda y.\Delta\,(y\,z)) \arrow[dl, "\mathbf{A}"'] \arrow[dr, "\mathbf{D}"] & \\
(\lambda y.\Delta\,(y\,z))\,I \arrow[d, "\mathbf{B}"'] & & (\lambda x.x\,I)\,(\lambda y.y\,z\,(y\,z)) \arrow[d, "\mathbf{E}"] \\
\Delta\,(I\,z) \arrow[d, "\mathbf{C}"'] \arrow[drr, dotted, "\mathbf{H}"] & & (\lambda y.y\,z\,(y\,z))\,I \arrow[d, "\mathbf{F}"] \\
\Delta\,z & & I\,z\,(I\,z) \arrow[d, "\mathbf{G}"] \\
 & & I\,z\,z
\end{tikzcd}
$$

- **A B** is already standard.
- **D E F** is standardized to **A B H**
- **A B H G** $\triangleright$ **A B C** by case 1 (and also 4!) of $\triangleright$
- *Moral:* **A** and **B** contribute to the creation of **G**. – **H** doesn't.

Families by labelling

Informally. Work with a labelled variant of the calculus in such a way that:

- Redexes with history ρR are assigned a label α ("name").
- Residuals of R have the same name as R.
- If contraction of a redex R creates a redex S, the name of R is a proper sublabel of the name of S.

Less informally.

Labels	$\alpha, \beta, \ldots$	$::=$	$\underbrace{\mathbf{a} \mid \mathbf{b} \mid \mathbf{c} \mid \ldots}_{\text{initial labels}} \mid \alpha\,\beta \mid \lceil\alpha\rceil \mid \lfloor\alpha\rfloor$
			modulo associativity: $\alpha(\beta\gamma) = (\alpha\beta)\gamma$
Labelled terms	$t, s, \ldots$	$::=$	$x^{\alpha} \mid \lambda^{\alpha} x.t \mid @^{\alpha}(t, s)$
Adding a label	$\alpha : x^{\beta}$	$\stackrel{\text{def}}{=}$	$x^{\alpha\beta}$
	$\alpha : \lambda^{\beta} x.t$	$\stackrel{\text{def}}{=}$	$\lambda^{\alpha\beta} x.t$
	$\alpha : @^{\beta}(t, s)$	$\stackrel{\text{def}}{=}$	$@^{\alpha\beta}(t, s)$
Substitution	$x^{\alpha}[x/t]$	$\stackrel{\text{def}}{=}$	$\alpha : t$ (plus the expected rules)
Reduction	$@^{\alpha}(\lambda^{\beta} x.t, s)$	$\to_{\ell}$	$\alpha\lceil\beta\rceil : t\{x := \lfloor\beta\rfloor : s\}$

Families by labelling: properties

- A term is initially labelled iff all of its nodes are decorated with pairwise distinct initial labels.
- The name (or degree) of a redex is the label decorating its abstraction.
- **Theorem (Lévy).** Let ρR and σS be redexes with history starting from the same initially labelled term. Then:

$$\rho R \text{ and } \sigma S \text{ have the same name} \iff \rho R \simeq \sigma S$$

Families by labelling: example

Let $I := \lambda x.x$ and $\Delta := \lambda x.x\,x$:

$$\begin{tikzcd}
 & (\lambda x.x\,I)\,(\lambda y.\Delta\,(y\,z)) \arrow[dl, "\mathbf{A}"'] \arrow[dr, "\mathbf{D}"] & \\
(\lambda y.\Delta\,(y\,z))\,I \arrow[d, "\mathbf{B}"'] & & (\lambda x.x\,I)\,(\lambda y.y\,z\,(y\,z)) \arrow[d, "\mathbf{E}"] \\
\Delta\,(I\,z) \arrow[d, "\mathbf{C}"'] & & (\lambda y.y\,z\,(y\,z))\,I \arrow[d, "\mathbf{F}"] \\
\Delta\,z & & I\,z\,(I\,z) \arrow[d, "\mathbf{G}"] \\
 & & I\,z\,z
\end{tikzcd}$$

$\mathbf{A\,B\,C} \simeq \mathbf{D\,E\,F\,G}$??

Families by labelling: example contd.

Let $I := \lambda^{\mathbf{a}} x.x^{\mathbf{b}}$ and $\Delta := \lambda^{\mathbf{c}} x.@^{\mathbf{d}}(x^{\mathbf{e}}, x^{\mathbf{f}})$:

- **A B C**:

$$\begin{array}{ll}
 & @^{\mathbf{g}}(\lambda^{\mathbf{h}} x.@^{\mathbf{i}}(x^{\mathbf{j}}, I), \lambda^{\mathbf{k}} y.@^{\mathbf{l}}(\Delta, @^{\mathbf{m}}(y^{\mathbf{n}}, z^{\mathbf{o}}))) \\
\xrightarrow{\mathbf{h}}_{\ell} & @^{\mathbf{g}\lceil\mathbf{h}\rceil\mathbf{i}}(\lambda^{\mathbf{j}\lfloor\mathbf{h}\rfloor\mathbf{k}} y.@^{\mathbf{l}}(\Delta, @^{\mathbf{m}}(y^{\mathbf{n}}, z^{\mathbf{o}})), I) \\
\xrightarrow{\mathbf{j}\lceil\mathbf{h}\rceil\mathbf{k}}_{\ell} & @^{\mathbf{g}\lceil\mathbf{h}\rceil\mathbf{i}\lceil\mathbf{j}\lceil\mathbf{h}\rceil\mathbf{k}\rceil\mathbf{l}}(\Delta, @^{\mathbf{m}}(\mathbf{n}\lfloor\mathbf{j}\lceil\mathbf{h}\rceil\mathbf{k}\rfloor : I, z^{\mathbf{o}})) \\
\xrightarrow{\mathbf{n}\lfloor\mathbf{j}\lceil\mathbf{h}\rceil\mathbf{k}\rfloor\mathbf{a}}_{\ell} & @^{\mathbf{g}\lceil\mathbf{h}\rceil\mathbf{i}\lceil\mathbf{j}\lceil\mathbf{h}\rceil\mathbf{k}\rceil\mathbf{l}}(\Delta, z^{\mathbf{m}\lceil\mathbf{n}\lfloor\mathbf{j}\lceil\mathbf{h}\rceil\mathbf{k}\rfloor\mathbf{a}\rceil\mathbf{b}\lfloor\mathbf{n}\lfloor\mathbf{j}\lceil\mathbf{h}\rceil\mathbf{k}\rfloor\mathbf{a}\rfloor\mathbf{o}})
\end{array}$$

- **D E F G**:

$$\begin{array}{ll}
 & @^{\mathbf{g}}(\lambda^{\mathbf{h}} x.@^{\mathbf{i}}(x^{\mathbf{j}}, I), \lambda^{\mathbf{k}} y.@^{\mathbf{l}}(\Delta, @^{\mathbf{m}}(y^{\mathbf{n}}, z^{\mathbf{o}}))) \\
\xrightarrow{\mathbf{c}}_{\ell} & @^{\mathbf{g}}(\lambda^{\mathbf{h}} x.@^{\mathbf{i}}(x^{\mathbf{j}}, I), \lambda^{\mathbf{k}} y.@^{\mathbf{l}\lceil\mathbf{c}\rceil\mathbf{d}}(@^{\mathbf{e}\lfloor\mathbf{c}\rfloor\mathbf{m}}(y^{\mathbf{n}}, z^{\mathbf{o}}), @^{\mathbf{f}\lfloor\mathbf{c}\rfloor\mathbf{m}}(y^{\mathbf{n}}, z^{\mathbf{o}}))) \\
\xrightarrow{\mathbf{h}}_{\ell} & @^{\mathbf{g}\lceil\mathbf{h}\rceil\mathbf{i}}(\lambda^{\mathbf{j}\lfloor\mathbf{h}\rfloor\mathbf{k}} y.@^{\mathbf{l}\lceil\mathbf{c}\rceil\mathbf{d}}(@^{\mathbf{e}\lfloor\mathbf{c}\rfloor\mathbf{m}}(y^{\mathbf{n}}, z^{\mathbf{o}}), @^{\mathbf{f}\lfloor\mathbf{c}\rfloor\mathbf{m}}(y^{\mathbf{n}}, z^{\mathbf{o}})), I) \\
\xrightarrow{\mathbf{j}\lceil\mathbf{h}\rceil\mathbf{k}}_{\ell} & @^{\mathbf{g}\lceil\mathbf{h}\rceil\mathbf{i}\lceil\mathbf{j}\lfloor\mathbf{h}\rfloor\mathbf{k}\rceil\mathbf{l}\lceil\mathbf{c}\rceil\mathbf{d}}(@^{\mathbf{e}\lfloor\mathbf{c}\rfloor\mathbf{m}}(\mathbf{n}\lfloor\mathbf{j}\lfloor\mathbf{h}\rfloor\mathbf{k}\rfloor : I, z^{\mathbf{o}}), @^{\mathbf{f}\lfloor\mathbf{c}\rfloor\mathbf{m}}(\mathbf{n}\lfloor\mathbf{j}\lfloor\mathbf{h}\rfloor\mathbf{k}\rfloor : I, z^{\mathbf{o}})) \\
\xrightarrow{\mathbf{n}\lfloor\mathbf{j}\lceil\mathbf{h}\rceil\mathbf{k}\rfloor\mathbf{a}}_{\ell} & @^{\mathbf{g}\lceil\mathbf{h}\rceil\mathbf{i}\lceil\mathbf{j}\lfloor\mathbf{h}\rfloor\mathbf{k}\rceil\mathbf{l}\lceil\mathbf{c}\rceil\mathbf{d}}(@^{\mathbf{e}\lfloor\mathbf{c}\rfloor\mathbf{m}}(\mathbf{n}\lfloor\mathbf{j}\lfloor\mathbf{h}\rfloor\mathbf{k}\rfloor : I, z^{\mathbf{o}}), z^{\mathbf{f}\lfloor\mathbf{c}\rfloor\mathbf{m}\lceil\mathbf{n}\lfloor\mathbf{j}\lfloor\mathbf{h}\rfloor\mathbf{k}\rfloor\mathbf{a}\rceil\mathbf{b}\lfloor\mathbf{n}\lfloor\mathbf{j}\lfloor\mathbf{h}\rfloor\mathbf{k}\rfloor\mathbf{a}\rfloor\mathbf{o}})
\end{array}$$

- Then **A B C** $\simeq$ **D E F G**.
- Note that $\mathbf{h} \subset \mathbf{j}\lceil\mathbf{h}\rceil\mathbf{k} \subset \mathbf{n}\lfloor\mathbf{j}\lceil\mathbf{h}\rceil\mathbf{k}\rfloor\mathbf{a} \not\supset \mathbf{c}$.

The Lévy labelled λ-calculus: more properties

- A predicate $\mathcal{P}$ on labels is said to be bounded iff there exists a bound $M \in \mathbb{N}$ such that for every label α:

$$\mathcal{P}(\alpha) \implies \mathtt{h}(\alpha) \leq M$$

  where $\mathtt{h}(\alpha)$ is the height of α (seen as a tree).

Then:

- **Theorem (Lévy).** The labelled λ-calculus restricted to any predicate is Church-Rosser. (In particular, it is CR).
- **Theorem (Lévy).** The labelled λ-calculus restricted to any bounded predicate is SN.
- There are three ways of creating redexes in the λ-calculus. They are the generalizations of the following examples:
  - $(\lambda x.\lambda y.t)\, s\, u \to (\lambda y.t\{x := s\})\, u$
  - $(\lambda x.x)\, (\lambda x.t)\, s \to (\lambda x.t)\, s$
  - $(\lambda x.x\, y)\lambda z.s \to (\lambda z.s)\, y$

  Key property.
  The name of a redex contains the names of all of its "causes" as sublabels.

The *linear substitution calculus* (LSC)

- Calculus of explicit substitutions "at a distance".
- Similar to calculi studied by Milner, De Bruijn, Nederpelt.
- Promoted by Accattoli and Kesner.

The *linear substitution calculus*: definition

- Terms and contexts:

$$t, s, u, \ldots ::= x \mid \lambda x.t \mid t\,s \mid t[x/s]$$

$$\mathtt{L} ::= \Box \mid \mathtt{L}[x/t]$$

$$C ::= \Box \mid \lambda x.C \mid C\,s \mid t\,C \mid C[x/s] \mid t[x/C]$$

- Notation for plugging terms into contexts:

$$C\langle t\rangle \text{ (capturing) vs. } C\langle\!\langle t\rangle\!\rangle \text{ (non-capturing)}$$

$$t\mathtt{L} \text{ rather than } \mathtt{L}\langle t\rangle$$

- Reduction:

$$\begin{array}{rcll} (\lambda x.t)\mathtt{L}\,s & \rightarrow_{\mathsf{db}} & t[x/s]\mathtt{L} & \\ C\langle\!\langle x\rangle\!\rangle[x/t] & \rightarrow_{\mathsf{ls}} & C\langle t\rangle[x/t] & \\ t[x/s] & \rightarrow_{\mathsf{gc}} & t & \text{if } x \notin \mathsf{fv}(t) \end{array}$$

- Example:

$$(\lambda x.\lambda y.x\,x)\,t\,s \rightarrow \ldots$$

Redex creation in the LSC

There are seven redex creation cases. They are the generalizations of the following examples:

$$
\begin{array}{l|lcl}
\textbf{db creates db.} & (\lambda x.\lambda y.t)\, s\, u & \to_{\mathsf{db}} & (\underline{\lambda} y.t)[x/s]\, u \\
\textbf{db creates ls.} & (\lambda x.x)\, t & \to_{\mathsf{db}} & \underline{x}[x/t] \\
\textbf{db creates gc.} & (\lambda x.y)\, t & \to_{\mathsf{db}} & y[\underline{x}/t] \\
\textbf{ls creates db}^{\uparrow}\textbf{.} & x[x/\lambda y.t]\, s & \to_{\mathsf{ls}} & (\underline{\lambda} y.t)[x/\lambda y.t]\, s \\
\textbf{ls creates db}^{\downarrow}\textbf{.} & (x\, s)[x/\lambda y.t] & \to_{\mathsf{ls}} & ((\underline{\lambda} y.t)\, s)[x/\lambda y.t] \\
\textbf{ls creates gc.} & x[x/t] & \to_{\mathsf{ls}} & t[\underline{x}/t] \\
\textbf{gc creates gc.} & a[x/y][y/z] & \to_{\mathsf{gc}} & a[\underline{y}/z]
\end{array}
$$

The labelled LSC

Labels	$\alpha, \beta, \ldots$	$::=$	$\mathbf{a} \mid \alpha\,\beta \mid \lceil\alpha\rceil \mid \lfloor\alpha\rfloor \mid \mathtt{db}(\alpha)$ mod associativity: $\alpha(\beta\gamma) = (\alpha\beta)\gamma$ distinguished initial labels: $\bullet$, $\otimes$
Label sets	Ω	$::=$	$\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ only initial labels, treated as sets
Terms	$t, s, \ldots$	$::=$	$x^{\alpha} \mid \lambda^{\alpha}_{\Omega} x.t \mid @^{\alpha}(t, s) \mid t[x/s]_{\Omega}$
Adding labels	$\alpha : x^{\beta}$	$\stackrel{\text{def}}{=}$	$x^{\alpha\beta}$
	$\alpha : \lambda^{\beta}_{\Omega} x.t$	$\stackrel{\text{def}}{=}$	$\lambda^{\alpha\beta}_{\Omega} x.t$
	$\alpha : @^{\beta}(t, s)$	$\stackrel{\text{def}}{=}$	$@^{\alpha\beta}(t, s)$
	$\alpha : (t[x/s]_{\Omega})$	$\stackrel{\text{def}}{=}$	$(\alpha : t)[x/s]_{\Omega}$
Reduction	$@^{\alpha}(\lambda^{\beta}_{\Omega} x.t, s)$	$\xrightarrow{\mathtt{db}(\beta)}_{\ell\,\mathsf{db}}$	$\alpha\lceil\beta\rceil : t[x/\lfloor\beta\rfloor : s]_{\Omega}$
	$C\langle\!\langle x^{\alpha}\rangle\!\rangle[x/t]_{\Omega}$	$\xrightarrow{\downarrow(\alpha)\,\bullet\,\uparrow(t)}_{\ell\,\mathsf{ls}}$	$C\langle \alpha\bullet : t\rangle[x/t]_{\Omega}$
	$t[x/s]_{\Omega}$	$\xrightarrow{\{\mathbf{a}\,\bullet\,\uparrow(s)\ \mid\ \mathbf{a}\in\Omega\}}_{\ell\,\mathsf{gc}}$	t

The labelled LSC

Some results:

- The labelled LSC is confluent for arbitrary predicates.
- The labelled LSC is SN for bounded predicates.
- *Issue:* we do **not** capture the "gc $\Rightarrow$ gc" creation case. This is reasonable since the property of redex stability does not hold[3], *i.e.* there are multiple ways of creating gc redexes:

$$
\begin{tikzcd}
 & & a[x/b[y/z]][z/c] \arrow[dll, bend right, "gc([y/z])"'] \arrow[drr, bend left, "ls(z\leftarrow c)"] \arrow[d, "/" marking, "gc([z/c])"] & & \\
a[x/b][z/c] \arrow[d, "gc([z/c])"'] \arrow[drr, bend right, "gc([x/b])"] & & {} & & a[x/b[y/c]][z/c] \arrow[d, "gc([z/c])"] \arrow[dll, bend left, "gc([x/b[y/c]])"'] \\
{} & & a[z/c] \arrow[d, "gc([z/c])"] & & {} \\
 & & {} & &
\end{tikzcd}
$$

[3] At least according to Lévy's formulation.

Current and future work

Short-term:

- Develop a method of contraction by extraction $\triangleright$.
  Piggyback on the work on standardization for the LSC by Accattoli, Bonelli, Kesner and Lombardi.

  This presents some difficulties by the fact that redexes are not stable.
- Characterize redex families proving the equivalences:

$$\text{zig-zag} \iff \text{extraction} \iff \text{labelling}$$

Long-term:

- Give a sharing graph implementation for optimal reduction.
- *Fuzzy question:* study how the "built-in" sharing of the LSC (explicit substitutions) relates with sharing for optimal reduction.

Basic references

Optimality

- Andrea Asperti, Stefano Guerrini.
  The Optimal Implementation of Functional Programming Languages.
- Jean-Jacques Lévy.
  Réductions correctes et optimales dans le lambda-calcul.
- Andrea Asperti, Harry Mairson.
  Parallel beta reduction is not elementary recursive.

Linear substitution calculus

- Beniamino Accattoli, Delia Kesner.
  The structural λ-calculus.
- Beniamino Accattoli, Eduardo Bonelli, Delia Kesner, Carlos Lombardi.
  A Nonstandard Standardization Theorem.