# Optimality and the Linear Substitution Calculus*

**Pablo Barenbaum[1] and Eduardo Bonelli[2]**

1 **Universidad de Buenos Aires, Buenos Aires, Argentinia; and**
**Université Paris 7, Paris, France; and**
**Stevens Institute of Technology, Hoboken, NJ, USA**
2 **Universidad Nacional de Quilmes and CONICET, Buenos Aires, Argentinia; and**
**Stevens Institute of Technology, Hoboken, NJ, USA**

**Abstract**

We lift the theory of optimal reduction to a decomposition of the lambda calculus known as the *Linear Substitution Calculus* (LSC). LSC decomposes $\beta$-reduction into finer steps that manipulate substitutions in two distinctive ways: it uses *context rules* that allow substitutions to act "at a distance" and rewrites modulo a set of *equations* that allow substitutions to "float" in a term. We propose a notion of redex family obtained by adapting Lévy labels to support these two distinctive features. This is followed by a proof of the finite family developments theorem (FFD). We then apply FFD to prove an optimal reduction theorem for LSC. We also apply FFD to deduce additional novel properties of LSC, namely an algorithm for standardisation by selection and normalisation of a linear call-by-need reduction strategy. All results are proved in the axiomatic setting of Glauert and Khashidashvili's *Deterministic Residual Structures*.

**1998 ACM Subject Classification** F.4.1 Mathematical Logic

**Keywords and phrases** Rewriting, Lambda Calculus, Explicit Substitutions, Optimal Reduction

**Digital Object Identifier** 10.4230/LIPIcs.FSCD.2017.9

## 1 Introduction

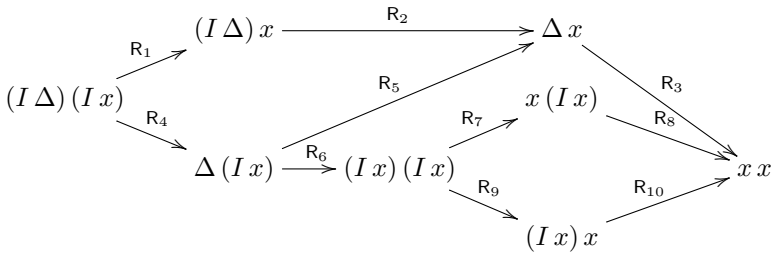
The $\lambda$-calculus distills the essence of functional programming languages. Programs are represented as syntactic terms, and execution corresponds to repeated simplification of these terms using a reduction rule called *$\beta$-reduction*. The study of the $\lambda$-calculus has produced a vast body of work, by no means limited to functional programming. It has also played a key role in laying the foundations of modern *rewriting theory*. Rewriting is an abstract model of computation in which rather than syntactic terms and their step-by-step reduction, one considers sets of *arrows* over arbitrary *objects*. The $\lambda$-calculus is an example of a rewriting system, but there are many other ones, such as graph rewriting systems or first-order term rewriting systems. The impact of the $\lambda$-calculus in rewriting is that its study has suggested generalizations of numerous properties to abstract rewriting frameworks.

There are many variants of the $\lambda$-calculus. In its simplest presentation, it consists of a unique *reduction rule* $\beta$ that models the application of a function to an argument. Despite the conciseness of its definition, the study of the $\lambda$-calculus unveils surprisingly rich mathematical structures. One example is its *denotational semantics*, which attempts to provide *models* for the $\lambda$-calculus, and motivates the theory of *domains*. Another example arises from attempting

* Work partially supported by PICT-2012-2747 (Ministerio de Ciencia, Tecnología e Innovación Productiva, Argentina) and LIA INFINIS.


to compare *derivations*. Given that computation is modeled by reduction and that there are multiple ways to reduce a term, how do these choices compare? This requires analyzing the *derivation space* of a term. The derivation space of a term is the set of all derivations, *i.e.* sequence of (composable) $\beta$-reduction steps, starting from that term. Establishing whether a particular choice produces derivations that are "better" than others in any reasonable sense involves *comparing* the resulting derivations. This, in turn, involves tracking steps around in order to relate the steps of one derivation to those of another one, hence determining that they *correspond* to each other. *Theories of residuals* attempt to provide a framework for analyzing the derivation space.

An example in the $\lambda$-calculus follows in order to provide a better intuition on what is meant by a theory of residuals. Consider the term $(I\,\Delta)\,(I\,x)$, where $\Delta$ stands for $\lambda x.x\,x$ and $I$ for $\lambda x.x$. Below we depict the derivation space of this term. As mentioned, a study of the *structure* of this space involves understanding how derivations are related and, since derivations are built from $\beta$-steps, how $\beta$-steps from one derivation are related to those of another.




An example of a derivation is $\mathsf{R}_4;\mathsf{R}_6;\mathsf{R}_7;\mathsf{R}_8$. It consists of four $\beta$-steps denoted $\mathsf{R}_4$, $\mathsf{R}_6$, $\mathsf{R}_7$, and $\mathsf{R}_8$. Notice that the derivation $\mathsf{R}_4;\mathsf{R}_6;\mathsf{R}_7;\mathsf{R}_8$ essentially performs the same steps as the derivation $\mathsf{R}_4;\mathsf{R}_6;\mathsf{R}_9;\mathsf{R}_{10}$ since the derivations $\mathsf{R}_7;\mathsf{R}_8$ and $\mathsf{R}_9;\mathsf{R}_{10}$ do the same computational work, namely they reduce the two copies of $(I\,x)$ in $(I\,x)\,(I\,x)$, only in a different order (reducible subterms such as $(I\,x)$ are called *redexes*). This suggests *algebraic principles* over derivations, such as $\mathsf{R}_7;\mathsf{R}_8 \simeq \mathsf{R}_9;\mathsf{R}_{10}$. Not all steps can be commuted. For example, $\mathsf{R}_4$ cannot be commuted with $\mathsf{R}_6$ because the former *creates* the latter. Note also that, we write $\mathsf{R}_7;\mathsf{R}_8 \simeq \mathsf{R}_9;\mathsf{R}_{10}$ and not $\mathsf{R}_7;\mathsf{R}_8 \simeq \mathsf{R}_8;\mathsf{R}_7$ because $\mathsf{R}_9$ is the form that $\mathsf{R}_8$ adopts when it is fired from $(I\,x)\,(I\,x)$ rather than from $x\,(I\,x)$; we say that $\mathsf{R}_8$ is a *residual* or what is left of $\mathsf{R}_9$ after $\mathsf{R}_7$. As may be gleamed from this preliminary discussion, it soon becomes clear that any prospective algebraic principles must arise from identifying $\beta$-steps and tracking them along derivations. Such *theories of residuals* mark and track $\beta$-steps. However, this is just the starting point of an analysis of the structure of the derivation space since, when one attempts to prove properties of derivations, one realizes that more general principles are required. The principles include the following, presented in increasing level of complexity:

- *Finite Developments*: marking and tracking *sets* of $\beta$-steps in a term and showing that their reduction terminates;
- *Finite Family Developments*: marking and tracking *sets* of $\beta$-step that may have been created along the way in a derivation and also showing their termination properties;
- *Redex Families*: identifying created $\beta$-steps that are related in the sense that they could be shared;
- *Optimal Reduction*: the apex of residual theory.

Optimal reduction characterizes derivations in the derivation space that are shortest in a precise sense and has close ties with Geometry of Interaction [16]. It arose with a clear motivation in the implementation of the $\lambda$-calculus since it addresses the concern of avoiding

unnecessary $\beta$-steps. This same motivation, bridging the gap between programming languages and their implementation, is shared by *Calculi with Explicit Substitutions*.

**Calculi with Explicit Substitutions (ES).** Substitution in the $\lambda$-calculus is a nontrivial metalanguage operation that simultaneously replaces every occurrence of a variable by a given term. In contrast, in actual implementations of functional programming languages it usually takes various steps to perform a substitution. For example, variables might be bound to values in an environment, and looked up in the environment whenever needed. Calculi with ES were introduced to bridge the gap between the $\lambda$-calculus and its implementations. They are characterized by the presence of an explicit operator in the object language for modeling substitution. A paradigmatic example calculus with ES is $\lambda\sigma$ [1] which includes, among others, rules `beta`[1] $(\lambda x.s)\,t \mapsto s[x/t]$, where $s[x/t]$ denotes an ES, and `app` $(s\,t)[x/u] \mapsto s[x/u]t[x/u]$, for propagating substitutions over applications. Unfortunately, these rules produce a critical pair rendering $\lambda\sigma$ a syntactically *non*-orthogonal system, a situation common to most known calculi with ES, as depicted below where $\to_{\texttt{beta}}$ means application of the `beta`-rule in an arbitrary context:

$$s[x/t][y/u] \;{}_{\texttt{beta}}\!\leftarrow ((\lambda x.s)\,t)[y/u] \to_{\texttt{app}} (\lambda x.s)[y/u]t[y/u]$$

The `beta`-step in the middle term has been "erased" in the right term because `beta` and `app` overlap. It is unclear how to devise a reasonable residual theory in such a situation[2]. The $\lambda$-calculus is thus set apart from traditional calculi with ES since in the latter the lack of orthogonality makes it impossible to address a proper theory of residuals, let alone optimality.

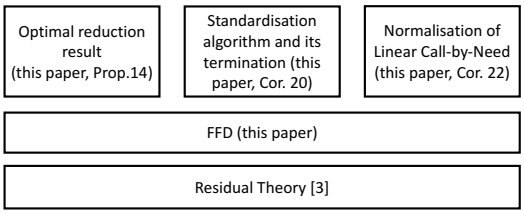
**The Linear Substitution Calculus (LSC).** The LSC is a calculus with ES introduced rather recently [6]. It is based on a *contextual* approach: rewriting rules are expressed using contexts, which allows for non-local interactions between subterms, and obviates the need to propagate explicit substitutions. It is also equipped with a relation of *structural equivalence* between terms, which reflects the exact correspondence between terms and their encoding as proof nets (which are graphs), linear logic being the domain in which the LSC was originally conceived.

The fact that the LSC encodes a graph-rewriting system based on proof nets, rather than *ad hoc* syntactic machinery for implementing explicit substitutions, is one of the reasons for it being relatively well-behaved. In particular, the LSC does not suffer from the above mentioned problems of other calculi with ES. Recent work has shown that, even though the LSC is not syntactically orthogonal, it enjoys *semantical orthogonality*, which means that it can be given a sensible *theory of residuals* [4]. On the other hand, not all expected properties of residuals that hold for the $\lambda$-calculus turn out to hold for LSC (*e.g.* *enclave* and *stability* fail [4]). Besides, the very same fact that the LSC encodes a graph-rewriting system is the source of some technical challenges, especially because the encoding is based on two distinctive features: the use of context rules and a notion of structural equivalence. One complication is that the usual tree-like representation of terms and nesting of redexes that guide our intuition in the $\lambda$-calculus and first-order term rewriting no longer applies. *E.g.*, in the term $(x\,x)[x/y]$ either of the two occurrences of $x$ might be replaced by $y$, so there are two redexes $(x\,x)[x/y] \to (y\,x)[x/y]$ and $(x\,x)[x/y] \to (x\,y)[x/y]$. These redexes overlap in

[1] $\lambda\sigma$ is actually based on de Bruijn indices, we use variable names for expository purposes.
[2] There are some attempts at addressing residual theories for syntactically non-orthogonal systems [14, 23].

the standard tree reading of terms, yet they should by all means be considered "independent" redexes. Another complication is that redex creation may take place at a distance, such as in the step $(x\,y)[x/I] \to (I\,y)[x/I]$, in which the substitution of $x$ by the identity creates a beta-redex. Anyhow, enough properties are satisfied by LSC's residual theory for it to be a reasonable starting point for following the path set by the $\lambda$-calculus: finite developments, finite family developments, redex families and optimal reduction.

**Goal and value of the paper.** This paper attempts to reclaim, for the LSC, the status of the $\lambda$-calculus by providing a theory of optimality for it. The key technical result on which it builds is a *Finite Family Developments (FFD)* theorem (Thm. 4 on page 9). FFD is used as a tool to develop various novel results, including optimal reduction itself, termination of standardisation procedures, and normalisation strategies. All results in this paper are proved in an axiomatic setting, namely *Deterministic Residual Structures* [15], whose axioms LSC is shown to comply with.




The reader will no doubt realize the technical nature of this paper. Standardisation, normalisation and, most notably, optimal reduction are known to be technical in themselves. The LSC is not much of an aid in this sense, its use of context rules and rewriting modulo a set of equations only seem to make matters yet more technical. We are well aware of this fact and have strived to present the material in such a way that the reader is able to see through the technicalities and perceive the value of this paper, namely how it manages to lift the theory of optimal reduction to *refinements* of the $\lambda$-calculus.

**Structure of this paper.** Sec. 2 defines LSC. We also review the definition of residuals for LSC and present *Deterministic Residual Structures* [15]. The Lévy labeled LSC is presented in Sec. 3. The *Finite Family Developments Theorem* is addressed in Sec. 4, its proof broken down into three principles. Sec. 5 addresses optimal reduction: we recall the notion of *Deterministic Residual Structure* from [15] and then prove that our labeled LSC is an instance of such structure. Sec. 6 introduces standardisation by selection (of multi-redexes) and proves termination. Sec. 7 studies a linear call-by-need strategy and proves that it normalizes. We conclude in Sec. 8. Proofs of all results are included in the extended version.

**Related Work.** The literature on FD is quite extensive; the reader is invited to consult [27, Ch. 4]. Some abstract notions of rewriting establish FD as an axiom [25, 21, 15]. For classical references to FFD there is [18, 12]. FFD generalizes Hyland-Wadsworth labels which records the depth of the labels [27, 8.4.4]. Also, it is referred to as *Generalized Finite Developments* in [19]. FFD was extended to higher-order rewriting [28, 26]. LSC was introduced by Milner [24] and then adopted by Accattoli and Kesner [6, 4] although similar ideas were also developed by de Bruijn and Nederpelt (see [8] for additional references). LSC has somewhat revived the explicit substitutions community given its success in explaining results in the classical $\lambda$-calculus (*e.g.* cost models, call-by-value solvability, call-by-value on open terms, linear head reduction and abstract machines, etc.) [9, 3, 7, 5, 8]. Regarding

standardisation for LSC, [4] proves the existence and uniqueness of standard derivations. However, standardisation algorithms are not studied. Residuals for calculi with ES have also been studied by Melliès [21, 22] where he developed a general theory of rewriting and applied it, among others, to $\lambda\sigma$ [1]. Regarding labels, ES and sharing there is some work [20, 13], however it all addresses weak reduction. We should also mention [29] which uses a calculus of ES and suggests an optimal reduction result for it. However, no proofs are supplied.

## 2 The Linear Substitution Calculus

Given *variables* $x, y, z, \ldots$, the set $\mathcal{T}$ of **terms** is defined by the grammar:

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

A term of the form $t[x/s]$ is called a **substitution**. The notion of free and bound variables is defined as usual, in particular $\lambda x.t$ and $t[x/s]$ bind all free occurrences of $x$ in $t$. We write $\mathsf{fv}(s)$ (resp. $\mathsf{bv}(s)$) for the set of free (resp. bound) variables of $s$. A **context** is a term with a unique occurrence of a singled-out variable $\Box$ called a hole. If $\mathtt{C}$ is a context, then $\mathtt{C}\langle t\rangle$ is the term resulting from replacing the hole in $\mathtt{C}$ with $t$ (possibly resulting in the capture of free variables of $t$ in $\mathtt{C}$). We write $\mathtt{C}\langle\!\langle t\rangle\!\rangle$ when the free variables of $t$ are not captured by $\mathtt{C}$.

Terms are considered up to a set of **structural equations** that allow commuting some substitutions around, in order to quotient out the order imposed by the fact that terms are trees rather than graphs, and to reflect more closely their correspondence with proof-nets. **Structural equivalence**, written $t \sim s$, is the reflexive, symmetric, transitive, and contextual closure of the following axioms:

$$\begin{array}{rcll}
(\lambda x.t)[y/s] & \sim_{\lambda} & \lambda x.t[y/s] & \text{if } x \neq y \text{ and } x \notin \mathsf{fv}(s) \\
(t\,s)[x/u] & \sim_{@} & t[x/u]\,s & \text{if } x \notin \mathsf{fv}(s) \\
t[x/s][y/u] & \sim_{com} & t[y/u][x/s] & \text{if } x \neq y,\ x \notin \mathsf{fv}(u), \text{ and } y \notin \mathsf{fv}(s)
\end{array}$$

▶ **Definition 1.** The LSC is the pair $\langle \mathcal{T}, \to\rangle$ where $\to$ is defined by the rules $\{\mathsf{db}, \mathsf{ls}\}$ modulo the equations $\{\sim_{\lambda}, \sim_{@}, \sim_{com}\}$, *i.e.* $t \to u$ if and only if $t \sim t'(\to_{\mathsf{db}} \cup \to_{\mathsf{ls}})u' \sim u$. Here $\to_{\mathsf{db}}$ is $\mathtt{C}\langle \mapsto_{\mathsf{db}}\rangle$ (*i.e.* the contextual closure of $\mapsto_{\mathsf{db}}$) and $\to_{\mathsf{ls}}$ is $\mathtt{C}\langle \mapsto_{\mathsf{ls}}\rangle$, $\mapsto_{\mathsf{db}}$ and $\mapsto_{\mathsf{ls}}$ being[3]:

$$(\lambda x.t)\mathtt{L}\ s \mapsto_{\mathsf{db}} t[x/s]\mathtt{L} \qquad\qquad \mathtt{C}\langle\!\langle x\rangle\!\rangle[x/t] \mapsto_{\mathsf{ls}} \mathtt{C}\langle t\rangle[x/t]$$

▶ Remark. Originally LSC includes $\to_{\mathsf{gc}}$, defined as the contextual closure of: $t[x/s] \mapsto_{\mathsf{gc}} t$, if $x \notin \mathsf{fv}(t)$. However, in the literature it is often ignored: dropping it simplifies the metatheory (*e.g.* LSC with $\to_{\mathsf{gc}}$ does not enjoy *stability* [4]; *cf.* Rem. 4) at no loss of generality since $\to_{\mathsf{gc}}$ can be postponed past $\to_{\mathsf{db}}$ and $\to_{\mathsf{ls}}$.

A **LSC-step** $(\mathsf{R}, \mathsf{S}, \ldots)$ is either a pair of the form $\langle \mathtt{C}, (\lambda x.t)\mathtt{L}\ s\rangle$ (a $\mathsf{db}$-step) or a triple of the form $\langle \mathtt{D}, \mathtt{C}\langle\!\langle x\rangle\!\rangle[x/t], \mathtt{C}\rangle$ (an $\mathsf{ls}$-step). Steps, as defined here, are often also called *redexes*. We write $\mathsf{src}(\mathsf{R})$ and $\mathsf{tgt}(\mathsf{R})$ for the source and target of $\mathsf{R}$, respectively. Two redexes are said to be **coinitial** (resp. **cofinal**) if their sources (resp. targets) coincide. A **derivation** $(\rho, \sigma, \ldots)$ is a sequence of steps $\mathsf{R}_1 \ldots \mathsf{R}_n$ s.t. $\mathsf{src}(\mathsf{R}_i) = \mathsf{tgt}(\mathsf{R}_{i-1})$ for $i \in 2..n$. We write $\epsilon$ for the empty derivation and $t \twoheadrightarrow s$ if there is a derivation from $t$ to $s$ and say that $t$ is its source and $s$ its target (empty derivations are assumed to be indexed by terms). *E.g.*:

$$\begin{array}{ccccccc}
 & & (\lambda x.\lambda y.xyx)II & \to_{\mathsf{db}} & (\lambda y.xyx)[x/I]I & \to_{\mathsf{ls}} & (\lambda y.Iyx)[x/I]I \\
\to_{\mathsf{db}} & (\lambda y.z[z/y]x)[x/I]I & \sim & & ((\lambda y.z[z/y]x)I)[x/I] & \to_{\mathsf{db}} & (z[z/y]x)[y/I][x/I]
\end{array}$$

[3] For the $\mapsto_{\mathsf{db}}$ rule we have opted to use the more familiar $t\mathtt{L}$ rather than $\mathtt{L}\langle t\rangle$.

**Residuals for LSC [4].** Given markers $\mathbf{a}, \mathbf{b}, \mathbf{c}, \ldots$, **marked terms**[4] are defined as follows, where $\alpha$ ranges over markers: $t, s ::= x \mid x^{\alpha} \mid t\, s \mid \lambda x.t \mid \lambda x^{\alpha}.t \mid t[x/s]$. Since markers are intended to mark redexes, we consider only *well-marked terms*: terms where marks are only placed on redexes (*cf.* [4]). For example, $\lambda x^{\mathbf{a}}.x$ and $\lambda x.x^{\mathbf{a}}$ are not well-marked. **Marked reduction** $\overset{\alpha}{\to}$ on well-marked terms is defined as the contextual closure of the following rewriting rules, where the contexts below are also well-marked:

$$(\lambda x^{\mathbf{a}}.t)\mathtt{L}s \overset{\mathbf{a}}{\mapsto}_{\mathtt{dB}} t[x/s]\mathtt{L} \qquad\qquad \mathtt{C}\langle\!\langle x^{\mathbf{a}} \rangle\!\rangle[x/t] \overset{\mathbf{a}}{\mapsto}_{\mathsf{ls}} \mathtt{C}\langle\!\langle t \rangle\!\rangle[x/t]$$

We write $\mathtt{Red}(s)$ (resp. $\mathtt{Red}_{\mathbf{a}}(s)$) for the set of redexes (resp. marked $\mathbf{a}$) in $s$. The **set of residuals of** $\mathsf{R}$ **after** $\mathsf{S}$ is given by $\mathsf{R}/\mathsf{S} := \{\mathtt{Red}_{\mathbf{a}}(u') \mid \mathtt{mark}(t, \mathsf{R}, \mathbf{a}) \overset{\mathsf{S}}{\to} u'\}$, where $\mathtt{mark}(t, \mathsf{R}, \mathbf{a})$ denotes the result of marking redex $\mathsf{R}$ in $t$ with $\mathbf{a}$. Given steps $\mathsf{S}$ and $\mathsf{T}$ such that $\mathsf{tgt}(\mathsf{S}) = \mathsf{src}(\mathsf{T})$, we say that $\mathsf{S}$ **creates** $\mathsf{T}$ if there is no $\mathsf{R}$ such that $\mathsf{R}/\mathsf{S} = \mathsf{T}$. A **multistep** is a non-empty finite set $\mathcal{M}$ of coinitial steps. The residual relation may be extended to multisteps as expected: $\mathcal{M}/\mathsf{S} \overset{\text{def}}{=} \bigcup_{\mathsf{R}\in\mathcal{M}} \mathsf{R}/\mathsf{S}$. Also, we may define the residual of a set of steps after a derivation: $\mathcal{M}/\epsilon \overset{\text{def}}{=} \mathcal{M}$, and $\mathcal{M}/\mathsf{S}\sigma \overset{\text{def}}{=} (\mathcal{M}/\mathsf{S})/\sigma$. Examples of the residual relation follow [4]: let $v = (x^{\mathbf{b}}x^{\mathbf{b}}x^{\mathbf{c}}y^{\mathbf{c}})[x/y][y/w]$, $\mathsf{S} = \langle \Box[y/w], (x^{\mathbf{b}}x^{\mathbf{b}}x^{\mathbf{c}}y^{\mathbf{c}})[x/y], x^{\mathbf{b}}\Box\, x^{\mathbf{c}}y^{\mathbf{c}}\rangle$ (so that $v \overset{\mathsf{S}}{\to} (x^{\mathbf{b}}y\, x^{\mathbf{c}}y^{\mathbf{c}})[x/y][y/w]$), and $\mathsf{R} = \langle \Box[y/w], (x^{\mathbf{b}}x^{\mathbf{b}}x^{\mathbf{c}}y^{\mathbf{c}})[x/y], \Box\, x^{\mathbf{b}}x^{\mathbf{c}}y^{\mathbf{c}}\rangle$. Observe that $\mathtt{mark}(v, \mathsf{R}, \mathbf{a}) = (x^{\mathbf{a}}x^{\mathbf{b}}x^{\mathbf{c}}y^{\mathbf{c}})[x/y][y/w] \overset{\mathsf{S}}{\to} (x^{\mathbf{a}}yx^{\mathbf{c}}x^{\mathbf{c}})[x/y][y/w]$. Therefore, if $\mathcal{M} = \{\mathsf{R}\}$, then $\mathcal{M}/\mathsf{S} = \{\mathsf{R}'\}$ where $\mathsf{R}' = \langle \Box\,[y/w], (x^{\mathbf{b}}y\, x^{\mathbf{c}}y^{\mathbf{c}})[x/y], \Box\, y\, x^{\mathbf{c}}y^{\mathbf{c}}\rangle$. Suppose now that $\mathcal{M} = \mathtt{Red}_{\mathbf{c}}(v)$. Then a similar analysis for each $\mathsf{R} \in \mathcal{M}$ yields $\mathcal{M}/\mathsf{S} = \{\mathsf{R}_1, \mathsf{R}_2\}$ where $\mathsf{R}_1 = \langle \Box\,[y/w], (x^{\mathbf{b}}y\, x^{\mathbf{c}}y^{\mathbf{c}})[x/y], x^{\mathbf{b}}y\,\Box\, y^{\mathbf{c}}\rangle$ and $\mathsf{R}_2 = \langle \Box\,, v, (x^{\mathbf{b}}y\, x^{\mathbf{c}}\Box)[x/y]\rangle$.

▶ Remark. Structural equivalence $\sim$ can be lifted to well-marked terms and the residual relation on steps shown to pass the equations in the sense that they induce a bijection between the steps they relate. Moreover, $\sim$ is a strong bisimulation with respect to $\to$ [4].

Marks are useful to study *developments*. For any $\mathcal{M} \subseteq \mathtt{Red}(t)$, a (possibly infinite) derivation from $t$, $\rho = \mathsf{R}_1\mathsf{R}_2\ldots$, is a **development** of $\mathcal{M}$ iff $\mathsf{R}_i \in \mathcal{M}/\mathsf{R}_1 \ldots \mathsf{R}_{i-1}$ for all $i$. A development $\rho$ of $\mathcal{M}$ is said to be **complete** if it is *maximal*, *i.e.* if there is no non-empty derivation $\sigma$ s.t. $\rho\sigma$ is also a development of $\mathcal{M}$. Note that if $\rho$ is finite, then $\mathcal{M}/\rho = \emptyset$. *E.g.* $t_0 = (xx)[x/t] \to (xt)[x/t] \to (tt)[x/t]$ is a complete development of the set containing the two $\mathsf{ls}$-steps of $t_0$.

## An Abstract Framework: Deterministic Residual Structures

Abstract rewriting frameworks, such as *Orthogonal Axiomatic Rewrite Systems* [21] and *Deterministic Residual Structures (DRS)* [15], single out properties that well-behaved residuals should enjoy. LSC with the above defined notion of residual satisfies the properties of both[5] of these frameworks. Here we briefly describe DRS since they shall be used when we address the applications of FFD to LSC.

An **Abstract Rewrite System** (ARS) is a tuple $\langle Obj, Stp, \mathsf{src}, \mathsf{tgt}\rangle$ of *objects*, *steps* and functions $\mathsf{src}$ and $\mathsf{tgt}$ that return the source and target, resp., of a step. Moreover, we assume that ARSs are finitely branching, *i.e.* that there is only a finite number of steps having the same object as source.

[4] [4] speaks of *labeled terms*, we use "marked" to stress that they are not to be confused with Lévy labels introduced in Sec. 3.

[5] Although, see comment on enclave and stability in the introduction.

▶ **Definition 2** (Deterministic Residual Structure)**.** A DRS is an ARS endowed with a residual relation $\_/\_$ satisfying the following axioms:

1. Unique ancestor. If $\mathsf{R} \in \mathsf{R}_1/\mathsf{S}$ and $\mathsf{R} \in \mathsf{R}_2/\mathsf{S}$ then $\mathsf{R}_1 = \mathsf{R}_2$.
2. Acyclicity. If $\mathsf{R} \neq \mathsf{S}$ and $\mathsf{R}/\mathsf{S} = \emptyset$ then $\mathsf{S}/\mathsf{R} \neq \emptyset$.
3. Finite Developments (FD). Let $\rho, \sigma$ be derivations and $\mathcal{M}$ be a set of coinitial steps.
   a. Finite. If $\rho$ a *development* of $\mathcal{M}$, then $\rho$ is finite.
   b. Cofinal. If $\rho, \sigma$ are *complete developments* of $\mathcal{M}$, then $\rho$ and $\sigma$ end in the same term.
   c. Equivalent. If $\rho, \sigma$ are complete developments of $\mathcal{M}$ then they induce the same residual relation, *i.e.* $\mathsf{R}/\rho = \mathsf{R}/\sigma$ for every step $\mathsf{R}$ coinitial with $\rho$.

In the case of LSC: Acyclicity is immediate; Unique ancestor and Finite Developments are proved in [4]. We next introduce some definitions proper to any DRS.

Each multistep determines a "super-step" by taking (any) complete development of that set. Its target is well-defined by axiom Cofinal. A **multistep reduction** $\mathsf{D}$ is a sequence of multisteps $\mathcal{M}_1 \dots \mathcal{M}_n$ s.t. $\mathsf{src}(\mathcal{M}_i) = \mathsf{tgt}(\mathcal{M}_{i-1})$ for $i \in 2..n$.

Define $\tau_1 \mathsf{R} \sigma \tau_2 \equiv^1 \tau_1 \mathsf{S} \rho \tau_2$, where $\sigma$ is a complete development of $\mathsf{S}/\mathsf{R}$ and $\rho$ is a complete development of $\mathsf{R}/\mathsf{S}$. We define **permutation equivalence**, $\equiv$, as the reflexive and transitive closure of $\equiv^1$. Note that $\rho \equiv \sigma$ implies $/\rho = /\sigma$ (*i.e.* they induce the same residual relation).

Let $\mathcal{X}$ be a set of objects in a DRS. An object $s$ is **$\mathcal{X}$-normalizing** if there is a derivation from $s$ to an object in $\mathcal{X}$. We call a step $\mathsf{R} \in \mathtt{Red}(s)$ **$\mathcal{X}$-needed** if at least one residual of it is contracted in any reduction from $s$ to an object in $\mathcal{X}$. *E.g.* for $\mathcal{X}$ the set of normal forms, the underlined step is needed in $\lambda x.\underline{II}$, but not if $\mathcal{X}$ is the set of abstractions.

A **redex with history** in a DRS is a non-empty derivation, usually written $\rho\mathsf{R}$ to single out the last step. We write $\mathsf{Hist}(t) \stackrel{\text{def}}{=} \{\rho\mathsf{R} \mid \mathsf{src}(\rho) = t\}$ for the set of redexes with history whose source is the object $t$. The **copy relation** between coinitial redexes with history, written $\rho\mathsf{R} \leq \sigma\mathsf{S}$ is defined to hold if and only if there is a derivation $\tau$ such that $\rho\tau \equiv \sigma$ and $\mathsf{S} \in \mathsf{R}/\tau$. The reflexive, symmetric and transitive closure of $\leq$, written $\leftrightsquigarrow$, is called the **family relation**. Its equivalence classes are called **redex families**. *E.g.*. if $\rho : \Delta(\underline{II}I) \to \Delta(II)$ and $\sigma : \underline{\Delta(III)} \to (\underline{II}I)(III) \to (II)(III)$, then $\sigma(\overline{II})(III)$ is in the family of $\rho\Delta(\overline{II})$ since $\sigma(\overline{II})(III) \leftrightsquigarrow \rho\Delta(\overline{II})$.

Let $\mathcal{F}$ be a set of redex families $\{\mathsf{Fam}_{\leftrightsquigarrow}(\rho_i)\}_{i\in I}$ such that $\rho_i \in \mathsf{Hist}(t)$, for some fixed $t$. A **family development** of $\mathcal{F}$ is a pair ${}_\tau|\rho$ where the first component is a "history" $\tau : t \twoheadrightarrow t'$ and the second component is a (possibly infinite) derivation $\rho = \mathsf{R}_1\mathsf{R}_2 \dots$ from $t'$ such that for every index $i \geq 1$, we have $\mathsf{Fam}_{\leftrightsquigarrow}(\tau\mathsf{R}_1 \dots \mathsf{R}_i) \in \mathcal{F}$. Usually the history $\tau$ is empty, $\rho$ starts from $t$, and we identify ${}_\epsilon|\rho$ with $\rho$. A family development ${}_\tau|\rho$ of $\mathcal{M}$ is said to be **complete** if it is *maximal*, *i.e.* if there is no non-empty derivation $\sigma$ s.t. ${}_\tau|\rho\sigma$ is also a family development of $\mathcal{F}$.

## 3 The Labeled LSC

Given *initial labels* $\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots$ including a distinguished one "$\bullet$", we define **labels** $\mathcal{L}$ as:

$$\alpha, \beta ::= \mathbf{a} \mid \lceil\alpha\rceil \mid \lfloor\alpha\rfloor \mid \mathtt{db}(\alpha) \mid \alpha\,\beta\,.$$

We assume juxtaposition $\alpha\,\beta$ to be associative. Labels that are *not* of the form $\alpha\,\beta$ are called **atomic labels**. Labels of the form $\mathtt{db}(\alpha)$ will be used to leave a trace indicating that a $\mathsf{db}$-step was contracted (*cf.* Rem. 3). Similarly, "$\bullet$" will be used to leave a trace indicating the place in which an $\mathsf{ls}$-step was contracted. The remaining labels play a similar rôle to that of Lévy labels for $\lambda$-calculus.

The set of **labeled terms** ($\mathcal{T}^\ell$), **labeled contexts** and **labeled substitution contexts** are defined by the following grammar:

$$\begin{array}{rcl} t,s,u,r,q & ::= & x^\alpha \mid \lambda^\alpha x.t \mid @^\alpha(t,s) \mid t[x/s] \\ \mathtt{C} & ::= & \Box \mid \lambda^\alpha x.\mathtt{C} \mid @^\alpha(\mathtt{C},t) \mid @^\alpha(t,\mathtt{C}) \mid \mathtt{C}[x/t] \mid t[x/\mathtt{C}] \\ \mathtt{L} & ::= & \Box \mid \mathtt{L}[x/t] \end{array}$$

Note that substitutions are not labeled. The **external label** of a term $t$, written $\ell(t)$, is the label decorating the outermost node of $t$, jumping over substitutions:

$$\begin{array}{rclcrcl} \ell(x^\alpha) & \stackrel{\text{def}}{=} & \alpha & \qquad & \ell(\lambda^\alpha x.t) & \stackrel{\text{def}}{=} & \alpha \\ \ell(@^\alpha(t,s)) & \stackrel{\text{def}}{=} & \alpha & & \ell(t[x/s]) & \stackrel{\text{def}}{=} & \ell(t) \end{array}$$

We also define the following operation $\alpha : t$ for adding a label to an (already labeled) term, jumping over substitutions:

$$\begin{array}{rclcrcl} \alpha : x^\beta & \stackrel{\text{def}}{=} & x^{\alpha\beta} & \qquad & \alpha : (\lambda^\beta x.t) & \stackrel{\text{def}}{=} & \lambda^{\alpha\beta} x.t \\ \alpha : @^\beta(t,s) & \stackrel{\text{def}}{=} & @^{\alpha\beta}(t,s) & & \alpha : (t[x/s]) & \stackrel{\text{def}}{=} & (\alpha : t)[x/s] \end{array}$$

Note that we have $\ell(t\mathtt{L}) = \ell(t)$ and $\alpha : (t\mathtt{L}) = (\alpha : t)\mathtt{L}$.

We shall require one more operation on labels. The **outermost (resp. innermost) atomic label** of a label $\alpha$ is written $\uparrow(\alpha)$ (resp. $\downarrow(\alpha)$) and defined as:

$$\uparrow(\alpha) \stackrel{\text{def}}{=} \begin{cases} \uparrow(\alpha_1) & \text{if } \alpha = \alpha_1\alpha_2 \\ \alpha & \text{otherwise} \end{cases} \qquad \downarrow(\alpha) \stackrel{\text{def}}{=} \begin{cases} \downarrow(\alpha_2) & \text{if } \alpha = \alpha_1\alpha_2 \\ \alpha & \text{otherwise} \end{cases}$$

These functions are well-defined modulo associativity of juxtaposition. We also write $\uparrow(t)$ for $\uparrow(\ell(t))$.

A labeled term is **initially labeled** if all its labels are initial and pairwise distinct. A labeled term $t \in \mathcal{T}^\ell$ is a **variant** of an (unlabeled) term $t_0 \in \mathcal{T}$ if erasing all the labels from $t$ yields $t_0$. We say that two labeled terms $t, s \in \mathcal{T}^\ell$ are variants of each other if they are variants of the same unlabeled term. Similarly, we may say that two labeled steps (resp. derivations) are variants of an unlabeled step (resp. derivation), or of each other. Sometimes we write $t^\ell$ to stand for a labeled variant of an unlabeled term $t$, and similarly for labeled steps and labeled derivations.

**Redex names** $\mathcal{RN}$ are defined as follows, where $\alpha'$ stands for the sort of atomic labels $\mu,\nu,\xi ::= \mathtt{db}(\alpha) \mid \alpha' \bullet \alpha'$. Note that, although we often identify redex names with the labels that represent them, they should be regarded as being of different sorts.

▶ **Definition 3** (Labeled LSC)**.** LLSC is the pair $\langle \mathcal{T}^\ell, \to_\ell \rangle$, where $\to_\ell \stackrel{\text{def}}{=} \to_{\ell\,\mathsf{db}} \cup \to_{\ell\,\mathsf{ls}}$, and $\to_{\ell\,\mathsf{db}} \stackrel{\text{def}}{=} \mathtt{C}\langle \mapsto_{\mathsf{db}} \rangle$ and $\to_{\ell\,\mathsf{ls}} \stackrel{\text{def}}{=} \mathtt{C}\langle \mapsto_{\mathsf{ls}} \rangle$. Relations $\mapsto_{\mathsf{db}}$ and $\mapsto_{\mathsf{ls}}$ are defined as:

$$\begin{array}{rcl} @^\alpha((\lambda^\beta x.t)\mathtt{L}, s) & \mapsto_{\mathsf{db}} & \alpha\lceil \mathtt{db}(\beta)\rceil : t[x/\lfloor \mathtt{db}(\beta)\rfloor : s]\mathtt{L} \\ \mathtt{C}\langle\!\langle x^\alpha \rangle\!\rangle[x/t] & \mapsto_{\mathsf{ls}} & \mathtt{C}\langle \alpha \bullet : t\rangle[x/t] \end{array}$$

The **name** of the $\mathsf{db}$-step above is $\mathtt{db}(\beta)$ and that of the $\mathsf{ls}$-step is $\downarrow(\alpha) \bullet \uparrow(t)$. We write $t \xrightarrow{\mu}_\ell s$ whenever there is a step $t \to_\ell s$ such that name of the contracted step is $\mu$. An example of a reduction in LLSC follows, it shows how a $\mathsf{db}$ redex can create a $\mathsf{db}$-step:

$$\begin{array}{rl} & @^{\mathbf{a}}(@^{\mathbf{b}}(\lambda^{\mathbf{c}} x.\lambda^{\mathbf{d}} y.x^{\mathbf{e}}, z^{\mathbf{f}}), z^{\mathbf{g}}) \\ \xrightarrow{\mathtt{db}(\mathbf{c})}_\ell & @^{\mathbf{a}}((\lambda^{\mathbf{b}\lceil \mathtt{db}(\mathbf{c})\rceil \mathbf{d}} y.x^{\mathbf{e}})[x/z^{\lfloor \mathtt{db}(\mathbf{c})\rfloor \mathbf{f}}], z^{\mathbf{g}}) \\ \xrightarrow{\mathtt{db}(\mathbf{b}\lceil \mathtt{db}(\mathbf{c})\rceil \mathbf{d})}_\ell & x^{\mathbf{a}\lceil \mathtt{db}(\mathbf{b}\lceil \mathtt{db}(\mathbf{c})\rceil \mathbf{d})\rceil}[y/z^{\lfloor \mathtt{db}(\mathbf{b}\lceil \mathtt{db}(\mathbf{c})\rceil \mathbf{d})\rfloor \mathbf{g}}][x/z^{\lfloor \mathtt{db}(\mathbf{c})\rfloor \mathbf{f}}] \end{array}$$

The other two forms of redex creation in LSC are when a db-step creates an ls-step (*e.g.* $(\lambda x.x)y \to_{\mathsf{db}} x[x/y]$) and when an ls-step creates a db-step (*e.g.* $(xy)[x/I] \to_{\mathsf{ls}} (Iy)[x/I]$).

Structural equivalence (Sec. 2) can be lifted to labeled terms as expected. The resulting **labeled structural equivalence**, also called $\sim$, is a strong bisimulation with respect to labeled reduction. LLSC is thus well-defined over $\sim$-equivalence classes. Furthermore, given two equivalent terms $t_1 \sim t_2$ there is a bijection $f$ between the set of steps of $t_1$ and the set of steps of $t_2$ such that $\mathsf{tgt}(\mathsf{R}) \sim \mathsf{tgt}(f(\mathsf{R}))$. The resulting system LLSC/ $\sim$ will also enjoy Church-Rosser and Finite Family Developments that LLSC will be shown to enjoy in Sec. 4.

▶ Remark. Labels of the form $\mathtt{db}(\alpha)$ (not present in Lévy labeling for $\lambda$-calculus) are included for technical reasons. Consider the following example in which an ls-step creates a db-step:

$$@^{\mathbf{a}}(x^{\mathbf{b}}, t)[x/\lambda^{\mathbf{c}} y.s] \xrightarrow{\mathbf{b}\bullet\mathbf{c}}_{\ell} @^{\mathbf{a}}(\lambda^{\mathbf{b}\bullet\mathbf{c}} y.s, t)[x/\lambda^{\mathbf{c}} y.s]$$

If the name of the db-step at the right hand side was declared to be the label decorating the $\lambda$-node, namely $\mathbf{b}\bullet\mathbf{c}$, it would coincide with the name of the ls-step we have just fired.

## 4 Finite Family Developments

FFD relies on the following properties of LLSC:

**Property 1:** Labeled reduction ($\to_\ell$) is weak Church–Rosser.

**Property 2:** Residuals of a step have the same name: $\mathsf{S}' \in \mathsf{S}/\rho$ implies $\mathsf{S}$ and $\mathsf{S}'$ have the same name in any labeling of any LLSC derivation $\rho$.

**Property 3:** Creation implies name contribution: if $\mathsf{R}$ creates $\mathsf{S}$ then $\mu \overset{\mathsf{Name}}{\hookrightarrow} \nu$, where $\mu$ denotes the name of $\mathsf{R}$ and $\nu$ denotes the name of $\mathsf{S}$. The latter relation is called *name contribution* and is defined as the transitive closure of the following rules:

1. $\mathtt{db}(\beta) \overset{\mathsf{Name}}{\hookrightarrow} \mathtt{db}(\alpha \lceil \mathtt{db}(\beta) \rceil \gamma)$
2. $\mathtt{db}(\beta) \overset{\mathsf{Name}}{\hookrightarrow} \alpha \bullet \lfloor \mathtt{db}(\beta) \rfloor$ where $\alpha$ is any atomic label.
3. $\downarrow(\alpha) \bullet \uparrow(\beta) \overset{\mathsf{Name}}{\hookrightarrow} \mathtt{db}(\alpha \bullet \beta)$

We next set out to prove FFD. Its precise statement is:

▶ **Theorem 4** (FFD). *Let $\mathcal{F}$ be a finite set of redex families in $\mathsf{Hist}(t)$ for some term $t$.*

1. *(Finite) there is no infinite family development of $\mathcal{F}$;*
2. *(Cofinal) the complete family developments of $\mathcal{F}$ all end in the same term; and*
3. *(Equivalent) any two complete family developments $\rho$ and $\sigma$ of $\mathcal{F}$ satisfy $\rho \equiv \sigma$,* i.e. *they are permutation equivalent.*

(Finite). Labeled reduction is clearly not SN since it can simulate $\beta$-reduction. However, if we restrict redex names to those that verify a *bounded predicate* [18], then we do obtain SN. A predicate on redex names $P : \mathcal{RN} \to \mathsf{Bool}$ is said to be **bounded** if the set $\{h(\mu) \mid P(\mu) \text{ holds}\}$ is bounded, where the *height* $h(\mu)$ *of a redex name* $\mu$ is the height of $\mu$ interpreted as a label, and the height of a label is defined as follows[6]:

$$h(\mathbf{a}) \overset{\text{def}}{=} 1 \qquad h(\alpha\beta) \overset{\text{def}}{=} \max\{h(\alpha), h(\beta)\} \qquad h(\mathbf{f}(\alpha)) \overset{\text{def}}{=} 1 + h(\alpha) \text{ if } \mathbf{f} \in \{\lceil\cdot\rceil, \lfloor\cdot\rfloor, \mathtt{db}(\cdot)\}$$

We write $\to_\ell^P$ for labeled reduction restricted to contracting steps whose names verify the predicate $P$. SN for $\to_\ell^P$ relies on the abstract termination result[7]: WCR $\wedge$ WN $\wedge$ Inc $\implies$

[6] This operation is well-defined modulo associativity of juxtaposition.

[7] Due to Klop and Nederpelt (see for instance [27, Theorem 1.2.3 (iii)]).

SN. WCR follows from **Property 1**, and the local confluence diagram for a pair of coinitial steps $\mathsf{R}$, $\mathsf{S}$ is closed with their relative residuals, which have the *same name* as their ancestors (**Property 2**). WN is attained by picking, at each step, a *non-duplicating* redex $\mathsf{R}$. This implies that $\mathsf{R}$ itself has no residual, every other $\to_\ell^P$-step $\mathsf{S} \neq \mathsf{R}$ has exactly one residual with the same name (**Property 2**) and steps created by $\mathsf{R}$ will have height strictly greater than that of $\mathsf{R}$ since creation implies name contribution (**Property 3**). Finally, Inc is rather easy given that we have a bound on $P$.

▶ **Proposition 5** (Bounded reduction is SN). *Let $P$ be a bounded predicate. Then $\to_\ell^P$ is SN.*

We may now conclude with a proof of (Finite): it follows from Prop. 5, the fact that all names in a redex family are identical, and that we only have a finite set of families. The axiom (Cofinal) follows from confluence of LLSC, a consequence of **Property 1**, Finite and Newman's Lemma (WCR+SN ⇒ CR). The axiom (Equivalent) follows from the fact that LLSC enjoys *algebraic confluence*: the confluence diagram for two coinitial derivations $\rho$ and $\sigma$ can be closed by tiling it with elementary permutation diagrams. This concludes the proof of FFD.

▶ Remark. We end the section with a remark on *stability*, stated as follows. Let $\mathsf{R} \neq \mathsf{S}$ be coinitial steps and let $\mathsf{T}_1, \mathsf{T}_2, \mathsf{T}_3$ be steps such that $\mathsf{T}_3 \in \mathsf{T}_1/(\mathsf{R}/\mathsf{S})$ and $\mathsf{T}_3 \in \mathsf{T}_2/(\mathsf{S}/\mathsf{R})$. Then there exists a step $\mathsf{T}_0$ such that $\mathsf{T}_1 \in \mathsf{T}_0/\mathsf{R}$ and $\mathsf{T}_2 \in \mathsf{T}_0/\mathsf{S}$. Stability is known to fail if $\to_{\mathsf{gc}}$ is added to LSC (indeed, it suffices to consider the two ways in which a $\mathsf{gc}$-step can be created in a term such as $x[y/z][z/t]$). Stability for LSC is an easy consequence[8] of the fact that residuals of steps have the same name as their ancestors (**Property 2**).

## 5 Optimal Reduction for LSC

An optimal reduction [18, 10] computes a value, assuming it exists, in the least number of steps. More precisely, if $\mathcal{A}$ and $\mathcal{B}$ are ARSs, we say that $\mathcal{B}$ is a **sub-ARS** of $\mathcal{A}$ if (1) they have the same objects, *i.e.* $Obj(\mathcal{A}) = Obj(\mathcal{B})$, (2) all the steps of $\mathcal{B}$ are also in $\mathcal{A}$, *i.e.* $Stp(\mathcal{B}) \subseteq Stp(\mathcal{A})$, and (3) the source (resp. target) of a step in $\mathcal{B}$ coincides with its source (resp. target) in $\mathcal{A}$. A **strategy** in an ARS $\mathcal{A}$ is a sub-ARS of $\mathcal{A}$ having the same set of objects and normal forms (*cf.* [27, Def. 9.1.1]). If $\mathcal{X}$ is a set of objects, a strategy is **$\mathcal{X}$-optimal** if for any object $t$ the length of any reduction from $t$ to an object $s \in \mathcal{X}$ is minimal among all the possible reductions from $t$ to $s$. Strategies such as call-by-name and call-by-value are not optimal: the former duplicates arguments and the latter evaluates unnecessary arguments. Call-by-need evaluates only arguments that are needed and stores their value for subsequent lookup and is indeed optimal [11]. However, all these are strategies in the ARS of closed $\lambda$-terms with *weak* reduction, in the sense that $\beta$-steps are not performed under lambdas: the set of normal forms are the abstractions. It is relatively easy to implement call-by-need in this case since it suffices to share *subterms* by labeling them [11]. Optimal reduction for the ARS of $\lambda$-terms with *strong* (*i.e.* unrestricted) reduction is more complicated since it involves reducing *under* lambdas: the set of normal forms is the usual set of $\beta$-normal forms. As a consequence, it requires sharing *contexts*, which notably complicates its implementation. Here we concentrate on a characterization of which of these steps should be shared, leaving implementation concerns, such as how to share contexts, for future work.

In the case of LSC, $\mathcal{X}$-optimality is not very interesting when $\mathcal{X}$ is the set of normal forms: since LSC has no erasing rules, all steps are trivially $\mathcal{X}$-needed. *E.g.* the $\mathsf{db}$-step in

[8] Also a consequence of LSC being a Deterministic Family Structure (*cf.* Sec. 5 and Lem. 4.1 of [15]).

$x[y/II]$ is needed to get to the normal form $x[y/I[z/I]]$. However, $II$ may be considered junk in that it is the body of a substitution whose target variable $y$ has no occurrence in $x$. Therefore, we introduce a more refined notion of *result* as a candidate for our set $\mathcal{X}$. We are only interested in steps in a term $t$ that are not junk in the sense that they have residuals in the $\mathsf{gc}$-normal form. Let $\mathsf{nf}_{\mathsf{gc}}(t)$ stand for the $\mathsf{gc}$-normal form of $t$. Our candidate $\mathcal{X}$ is the set of **reachable normal forms**, defined as $\mathsf{RNF} \stackrel{\text{def}}{=} \{t \mid \mathsf{nf}_{\mathsf{gc}}(t) \text{ is in } \to_{\mathsf{db} \cup \mathsf{ls}}\text{-normal form}\}$. Later we shall see that it has the properties required of a set of results (*cf.* notion of stable set of objects below).

## 5.1 An Abstract Framework for Optimal Reduction

An abstract framework for obtaining optimal reduction results was developed by Khasidashvili and Glauert [15]. They introduce axioms on DRS that verse over steps, residuals and redex families and show that if they are satisfied, then an optimal reduction result holds. These axioms are collected in a structure called *Deterministic Family Structures (DFS)*:

$$\text{ARS (Sec. 2)} \subseteq \text{DRS (Def. 2)} \subseteq \text{DFS (Def. 6)}.$$

▶ **Definition 6.** A **Deterministic Family Structure** is a triple $\langle R, \simeq, \hookrightarrow \rangle$, where $R$ is a DRS, $\simeq$ is an equivalence relation between coinitial redexes with history whose equivalence classes are called *families*, and $\hookrightarrow$ is a binary relation of *contribution* between coinitial families. The family of a redex with history $\rho\mathsf{R}$ is written $\mathsf{Fam}_{\simeq}(\rho\mathsf{R})$. Two families are coinitial if their representatives are coinitial. Moreover, the following axioms hold:

1. Initial. If $\mathsf{R}, \mathsf{S}$ are distinct coinitial steps, then $\mathsf{Fam}_{\simeq}(\mathsf{R}) \neq \mathsf{Fam}_{\simeq}(\mathsf{S})$.
2. Copy. $\leq \subseteq \simeq$. Recall that $\leq$ is the copy relation of DRS.
3. Finite Family Developments. Any derivation that contracts redexes of a finite number of families is finite.
4. Creation. If $\rho\mathsf{R}$ is a redex with history and $\mathsf{R}$ creates $\mathsf{S}$, then $\mathsf{Fam}_{\simeq}(\rho\mathsf{R}) \hookrightarrow \mathsf{Fam}_{\simeq}(\rho\mathsf{RS})$.
5. Contribution. Given any two coinitial families $\phi_1, \phi_2 \in \mathsf{Hist}(t)/\simeq$, the relation $\phi_1 \hookrightarrow \phi_2$ holds, if and only if, for every redex with history $\sigma\mathsf{S} \in \phi_2$, there is a redex with history $\rho\mathsf{R} \in \phi_1$ such that $\rho\mathsf{R}$ is a prefix of $\sigma$ (*i.e.* $\sigma = \rho\mathsf{R}\sigma'$).

A **family reduction** in a DFS is a multistep reduction $\mathcal{M}_1 \dots \mathcal{M}_\mathsf{n}$ such that for each $i \in 1..n$ all the steps in $\mathcal{M}_\mathsf{i}$ belong to the same redex family. More precisely, for all $i \in 1..n$ and any two $\mathsf{R}, \mathsf{S} \in \mathcal{M}_\mathsf{i}$, it holds that $\mathcal{M}_1 \dots \mathcal{M}_{\mathsf{i}-1}\mathsf{R} \simeq \mathcal{M}_1 \dots \mathcal{M}_{\mathsf{i}-1}\mathsf{S}$. A family reduction is **complete** if each $\mathcal{M}_\mathsf{i}$ is a maximal set of steps that have $\mathsf{src}(\mathcal{M}_\mathsf{i})$ as source and belong to the same family. Let $\mathcal{X}$ be a set of objects. A family reduction is **$\mathcal{X}$-needed** if each $\mathcal{M}_\mathsf{i}$ contains at least one $\mathcal{X}$-needed step (*cf.* Sec. 2).

For a set $\mathcal{X}$ of objects to be admitted as a set of results it has to satisfy the following property. A set $\mathcal{X}$ of objects is **stable** if: 1) $\mathcal{X}$ is closed under parallel moves, *i.e.* for any $t \notin \mathcal{X}$, any $\rho : t \twoheadrightarrow s \in \mathcal{X}$, and any $\sigma : t \twoheadrightarrow u$ which does not contain objects in $\mathcal{X}$, the final object of $\rho/\sigma$ is in $\mathcal{X}$; and 2) $\mathcal{X}$ is closed under unneeded expansion, *i.e.* for any $t \xrightarrow{\mathsf{R}} s$ such that $t \notin \mathcal{X}$ and $s \in \mathcal{X}$, the step $\mathsf{R}$ is $\mathcal{X}$-needed. The set of LSC-normal forms and the set of abstractions are stable. Less obvious is the fact that $\mathsf{RNF}$ is a stable set. This is non-trivial. For item 1) we show that the set $\mathsf{RNF}$ is closed under reduction, which entails that it is closed under parallel moves. For item 2) we strengthen the notion of reachable steps to that of *strongly reachable steps* (reachable steps that are minimal w.r.t. the box order introduced in [4] for the purposes of studying standardisation).

▶ **Lemma 7.** *The set of reachable-normal forms* $\mathsf{RNF}$ *is stable.*

The main result of this section is the following theorem. It is a corollary of Prop. 9 and of Theorem 5.2 in [15]:

▶ **Theorem 8.** *Let $\mathcal{X}$ be a stable set of terms of* LSC*. Let $t$ be a $\mathcal{X}$-normalising term. Then any $\mathcal{X}$-needed $\mathcal{X}$-normalizing complete family-reduction $\rho : t \twoheadrightarrow t' \in \mathcal{X}$ is $\mathcal{X}$-optimal,* i.e. *it has a minimal number of family-reduction steps.*

## 5.2 LSC is a Deterministic Family Structure

Given two coinitial redexes with history $\rho\mathsf{R}$, $\sigma\mathsf{S}$, the binary relations of **family equivalence** $\rho\mathsf{R} \overset{\mathsf{Fam}}{\simeq} \sigma\mathsf{S}$ and **family contribution** $\rho\mathsf{R} \overset{\mathsf{Fam}}{\hookrightarrow} \sigma\mathsf{S}$ are defined as follows. Consider labeled variants $\rho^\ell\mathsf{R}^\ell$ and $\sigma^\ell\mathsf{S}^\ell$ of $\rho\mathsf{R}$ and $\sigma\mathsf{S}$ respectively, starting from the same initially labeled term. Let $\mu$ be the name of $\mathsf{R}^\ell$ and let $\nu$ be the name of $\mathsf{S}^\ell$. We declare $\rho\mathsf{R} \overset{\mathsf{Fam}}{\simeq} \sigma\mathsf{S}$ to hold if and only if $\mu = \nu$ and $\rho\mathsf{R} \overset{\mathsf{Fam}}{\hookrightarrow} \sigma\mathsf{S}$ to hold if and only if $\mu \overset{\mathsf{Name}}{\hookrightarrow} \nu$. Relation $\overset{\mathsf{Fam}}{\hookrightarrow}$ is also extended to coinitial families, declaring $\phi_1 \overset{\mathsf{Fam}}{\hookrightarrow} \phi_2$ to hold whenever for any $\rho\mathsf{R} \in \phi_1$ and $\sigma\mathsf{S} \in \phi_2$ we have $\rho\mathsf{R} \overset{\mathsf{Fam}}{\hookrightarrow} \sigma\mathsf{S}$. It is straightforward to check that this is well-defined, regardless of the choice of representatives.

▶ **Proposition 9.** $(\text{LSC}, \overset{\mathsf{Fam}}{\simeq}, \overset{\mathsf{Fam}}{\hookrightarrow})$ *is a Deterministic Family Structure.*

The axioms Initial, Copy, and Creation can be checked by exhaustive case analysis. The Finite family developments axiom has already been established in Thm. 4. The Contribution axiom is more demanding and relies on a non-trivial application of FFD.

# 6 Standardisation by Selection for LSC

We introduce an abstract notion of *uniform multi-selection strategy*, show that repeated application of this strategy terminates using FFD in any DFS, and finally that two permutation equivalent derivations produce the same multiderivation. Then we instantiate our abstract result to LSC, obtaining an algorithm for standardizing LSC derivations by picking multisteps according to a given parametric partial order on its steps.

**Uniform Multi-Selection Strategies.** A step $\mathsf{R}$ **belongs** to a derivation $\rho$, written $\mathsf{R} \triangleleft \rho$, if and only if $\rho = \rho_1\mathsf{R}'\rho_2$ and $\mathsf{R}' \in \mathsf{R}/\rho_1$. Given a DRS $\mathcal{A}$, we write $Stp^+$ for the set of multisteps, *i.e.* non-empty finite sets of coinitial steps, and we let $\mathsf{D}$, $\mathsf{E}$, etc. range over multiderivations, *i.e.* derivations in the DRS whose steps are multisteps. A multistep $\mathcal{M}$ belongs to a derivation $\rho$, written $\mathcal{M} \triangleleft \rho$, if and only if $\mathsf{R} \triangleleft \rho$ for all $\mathsf{R} \in \mathcal{M}$. If $\mathsf{D} = \mathcal{M}_1 \dots \mathcal{M}_\mathsf{n}$ is a multiderivation, we say that a derivation $\rho$ is a complete development of $\mathsf{D}$ if $\rho = \rho_1 \dots \rho_\mathsf{n}$, where each $\rho_\mathsf{i}$ is a complete development of the multistep $\mathcal{M}_\mathsf{i}$. By FD a complete development always exists and any two complete developments are permutation equivalent. We write $\partial\mathsf{D}$ to stand for some complete development of $\mathsf{D}$, and $\rho/\mathsf{D}$ for $\rho/\partial\mathsf{D}$. A **multi-selection strategy** is a function $\mathbb{M}$ that maps every non-empty derivation $\rho$ to a coinitial multistep $\mathcal{M} \in Stp^+$ such that $\mathcal{M} \triangleleft \rho$ and $\mathcal{M}/\rho = \emptyset$, *i.e.* residuals of every step appear somewhere in the sequence, and there are no residuals of any step left after the sequence. It is, moreover, **uniform** if $\rho \equiv \sigma$ implies $\mathbb{M}(\rho) = \mathbb{M}(\sigma)$ for any non-empty $\rho, \sigma$. *E.g.* $\mathbb{M}_{\text{Triv}}(\mathsf{R}\rho) \overset{\text{def}}{=} \{\mathsf{R}\}$ is a (trivial) multi-selection strategy, which is not uniform.

The **multiderivation induced by a multi-selection strategy $\mathbb{M}$ on a derivation** $\rho$, written $\mathbb{M}^\star(\rho)$, is a sequence of multisteps defined as follows:

$$\mathbb{M}^\star(\epsilon) = \epsilon \qquad \mathbb{M}^\star(\rho) = \mathbb{M}(\rho)\,\mathbb{M}^\star(\rho/\mathbb{M}(\rho)) \quad \text{if } \rho \neq \epsilon$$

Successively applying a multi-selection strategy $\mathbb{M}$ to build a reduction sequence $\mathbb{M}^\star(\rho)$ terminates, as long as the input $\rho$ is finite, *i.e.* recursion is well-founded. This relies on FFD.

▶ **Lemma 10** (Induced multiderivations preserve finiteness)**.** *Suppose that $\mathbb{M}$ is a multi-selection strategy in a DFS. If $\rho$ is finite, then $\mathbb{M}^\star(\rho)$ is also finite.*

By definition, a uniform multi-selection strategy $\mathbb{M}$, when given two permutation equivalent derivations, always selects the same multistep. It, in fact, yields the same multiderivation.

▶ **Lemma 11.** *Let $\mathbb{M}$ be a uniform multi-selection strategy in a DFS, and let $\rho$, $\sigma$ be finite derivations. If $\rho \equiv \sigma$ then $\mathbb{M}^\star(\rho) = \mathbb{M}^\star(\sigma)$.*

▶ **Lemma 12.** *Let $\mathbb{M}$ be a multi-selection strategy in a DFS, and $\rho$ a finite derivation. Then $\rho \equiv \partial\mathbb{M}^\star(\rho)$.*

A multiderivation $\mathsf{D}$ is said to be **$\mathbb{M}$-compliant** if and only if $\mathbb{M}^\star(\partial\mathsf{D}) = \mathsf{D}$.

▶ **Proposition 13.** *Let $\mathbb{M}$ be a uniform multi-selection strategy in a DFS. For any finite derivation $\rho$ there exists a unique multiderivation $\mathsf{D}$ such that $\rho \equiv \partial\mathsf{D}$ and $\mathsf{D}$ is $\mathbb{M}$-compliant. Namely, $\mathsf{D} = \mathbb{M}^\star(\rho)$.*

**Standardisation Algorithm for LSC.** For each term $t$ let $\mathsf{Out}(t)$ be the set of steps whose source is $t$ in LSC, and let $<_t$ be an arbitrary strict *partial order* on $\mathsf{Out}(t)$. The **arbitrary selector** $\mathbb{M}_<$ is defined as follows: $\mathbb{M}_<(\rho) \stackrel{\text{def}}{=} \{\mathsf{R} \mid \mathsf{R}/\rho = \emptyset \text{ and } \mathsf{R} \text{ is minimal}\}$. By minimal we mean that there is no step $\mathsf{R}'$ such that $\mathsf{R}'/\rho = \emptyset$ and $\mathsf{R}' <_{\mathsf{src}(\rho)} \mathsf{R}$. Note that $\mathbb{M}_<$ is a non-empty finite set, given that the set $\{\mathsf{R} \mid \mathsf{R}/\rho = \emptyset\}$ is non-empty and finite, so it has at least one minimal element.

▶ **Lemma 14.** *$\mathbb{M}_<$ is a uniform multi-selection strategy.*

▶ **Corollary 15** (Standardisation by arbitrary selection for LSC)**.** *For each finite sequence $\rho$ in* LSC*, there is a unique multiderivation $\mathsf{D}$ such that $\rho \equiv \partial\mathsf{D}$ and $\mathsf{D}$ is $\mathbb{M}_<$-compliant. Moreover, if the order $<_t$ is computable, then $\mathsf{D}$ is computable from $\rho$, namely $\mathsf{D} = \mathbb{M}^\star_<(\rho)$.*

For example, let $\rho : x[x/t] \to x[x/t'] \to t'[x/t'] \to t''[x/t']$, where $t \to t' \to t''$.

1. If $<^1$ is the trivial partial order in which every step is incomparable, *i.e.* $\mathsf{R} <^1_t \mathsf{S}$ never holds, then $\mathbb{M}^\star_{<^1}(\rho) : x[x/t] \multimap t'[x/t'] \to t''[x/t']$. The first step is a proper multistep.
2. Let $<^2$ be the total left-to-right order, defined so that $\mathsf{R} <^2_t \mathsf{S}$ holds whenever $\mathsf{R}$ is to the left of $\mathsf{S}$. Then $\mathbb{M}^\star_{<^2}(\rho) : x[x/t] \to t[x/t] \to t'[x/t] \to t'[x/t'] \to t''[x/t']$.
3. If $<^3$ is the total right-to-left order, defined so that $\mathsf{R} <^3_t \mathsf{S}$ holds if $\mathsf{R}$ is to the right of $\mathsf{S}$. Then $\mathbb{M}^\star_{<^3}(\rho) = \rho : x[x/t] \to x[x/t'] \to t'[x/t'] \to t''[x/t']$.

## 7 Normalisation of the Linear Needed Strategy in LSC

Recall that a **strategy** in an ARS is a sub-ARS having the same objects and normal forms. We write $\mathsf{NF}(\mathcal{A})$ for the set of normal forms of an ARS $\mathcal{A}$, and $t \to_{\mathcal{A}} s$ to emphasize that a given step is in $\mathcal{A}$. If $\mathcal{X}$ is a superset of the normal forms of $\mathcal{A}$, a strategy $\mathbb{S}$ is said to be **$\mathcal{X}$-normalizing** if for every object $t$ such that there exists a reduction $t \twoheadrightarrow_{\mathcal{A}} s \in \mathcal{X}$, every maximal reduction from $t$ in the strategy $\mathbb{S}$ contains an object in $\mathcal{X}$. A sub-ARS $\mathcal{B}$ is **residual-invariant** if for any steps $\mathsf{R}$ and $\mathsf{S}$ such that $\mathsf{R} \in \mathcal{B}$ and $\mathsf{S} \neq \mathsf{R}$, there exists a step $\mathsf{R}' \in \mathbb{S}$ such that $\mathsf{R}' \in \mathsf{R}/\mathsf{S}$. A sub-ARS $\mathcal{B}$ is **closed** if the set $\mathsf{NF}(\mathcal{B})$ is closed by reduction, *i.e.* $t \to_{\mathcal{A}} s$ and $t \in \mathsf{NF}(\mathcal{B})$ imply $s \in \mathsf{NF}(\mathcal{B})$. Observe that any sub-ARS $\mathcal{B}$ can

be extended to a strategy $\mathbb{S}_{\mathcal{B}}$ by adjoining the steps going out from normal forms, *i.e.* by setting $Stp(\mathbb{S}_{\mathcal{B}}) := Stp(\mathcal{B}) \cup \{\mathsf{R} \in Stp(\mathcal{A}) \mid \mathsf{src}(\mathsf{R}) \in \mathsf{NF}(\mathcal{B})\}$. We will instantiate the following normalisation result to the linear call-by-need strategy of LSC which we define below.

▶ **Proposition 16.** *Let $\mathcal{B}$ be a closed residual-invariant sub-ARS in a DFS. Then the corresponding strategy $\mathbb{S}_{\mathcal{B}}$ is $\mathsf{NF}(\mathcal{B})$-normalizing.*

**Needed linear reduction** for LSC is the sub-ARS $\mathbb{NL}$ of LSC defined as follows. Need contexts are defined by the grammar $\mathtt{N} ::= \Box \mid \mathtt{N}\,t \mid \mathtt{N}[x/t] \mid \mathtt{N}\langle\langle x \rangle\rangle[x/\mathtt{N}]$. The reduction rule $\to_{\mathbb{NL}}$ is the union of the usual $\mathsf{db}$ rule, and the $\mathsf{lsnl}$ rule $\mathtt{N}\langle\langle x \rangle\rangle[x/\mathtt{vL}] \mapsto_{\mathsf{lsnl}} \mathtt{N}\langle \mathtt{vL} \rangle[x/\mathtt{vL}]$, both closed by need contexts, where $\mathtt{v}$ stands for a *value*, *i.e.* a term of the form $\lambda y.t$. Note that it is in fact a sub-ARS for LSC, *i.e.* the $\mathsf{lsnl}$ rule is a particular case of the $\mathsf{ls}$ rule, and closure by need contexts is a particular case of closure by general contexts. A similar, albeit slightly different call-by-need calculus based on LSC has been studied in [2] to relate the execution model of abstract machines with reduction in calculi with ES. In [17] it is shown, via intersection types, that it is a sound and complete implementation of call-by-name.

The set of **needed linear normal forms** $\mathsf{NLNF}$ is defined by the grammar $A ::= (\lambda x.t)\mathtt{L} \mid \mathtt{N}\langle\langle x \rangle\rangle$. Terms of the form $(\lambda x.t)\mathtt{L}$ are called **answers**, and $\mathtt{N}\langle\langle x \rangle\rangle$ are called **structures**. In structures, $\mathtt{N}$ does not bind $x$, the latter called its **needed variable**.

▶ **Corollary 17.** *The strategy $\mathbb{S}_{\mathbb{NL}}$ associated to the sub-ARS $\mathbb{NL}$ is $\mathsf{NLNF}$-normalizing.*

The proof consists in first showing that $\mathsf{NF}(\mathbb{NL})$ coincides with the set $\mathsf{NLNF}$, and then that the sub-ARS $\mathbb{NL}$ is closed residual-invariant. These items rely on a number of lemmata such as the fact that the needed variable in a structure is unique and that answers cannot be written as of the form $\mathtt{N}\langle\Delta\rangle$ where $\Delta$ is a redex or a variable not bound by $\mathtt{N}$.

## 8 Conclusions

The *Linear Substitution Calculus* sits between calculi with ES and the $\lambda$-calculus: it has ES but admits a theory of residuals. We devise a theory of optimal reduction for LSC. We start from the theory of residuals developed in [4] and use it to prove a Finite Family Developments result. This is achieved by introducing a Lévy labeling and associated notion of redex family which supports the two distinctive features of LSC, namely its use of *context rules* that allow substitutions to act "at a distance" and also the set of equations modulo which it rewrites which allow substitutions to "float" in a term. We then apply FFD to prove a number of novel results for LSC including: an optimal reduction result, an algorithm for standardisation by selection, and normalization of a linear call-by-need reduction strategy.

Perhaps the most relevant future work is devising an appropriate notion of extraction and showing that all three characterizations (labeling, zig-zag and extraction) of redex family coincide. This is non-trivial and has elided us for some time. Also, there is the topic of graph based implementations, labels and virtual redexes (*cf.* notion of paths in Ch.6 of [10]).

**Acknowledgements.** To Thibaut Balabonski and Beniamino Accattoli for helpful discussions.

## References

1 Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. *J. Funct. Program.*, 1(4):375–416, 1991. doi:10.1017/S0956796800000186.

2 Beniamino Accattoli, Pablo Barenbaum, and Damiano Mazza. Distilling abstract machines. In J. Jeuring and M. Chakravarty, editors, *Proceedings of ICFP*, pages 363–376. ACM, 2014.

3 Beniamino Accattoli, Pablo Barenbaum, and Damiano Mazza. A strong distillery. In Xinyu Feng and Sungwoo Park, editors, *APLAS 2015, Pohang, South Korea, November 30 – December 2, 2015, Proceedings*, volume 9458 of *LNCS*, pages 231–250. Springer, 2015. doi:10.1007/978-3-319-26529-2_13.

4 Beniamino Accattoli, Eduardo Bonelli, Delia Kesner, and Carlos Lombardi. A nonstandard standardization theorem. In Suresh Jagannathan and Peter Sewell, editors, *POPL'14, San Diego, CA, USA, January 20-21, 2014*, pages 659–670. ACM, 2014. doi:10.1145/2535838.2535886.

5 Beniamino Accattoli and Claudio Sacerdoti Coen. On the relative usefulness of fireballs. In *LICS 2015, Kyoto, Japan, July 6-10, 2015*, pages 141–155. IEEE Computer Society, 2015. doi:10.1109/LICS.2015.23.

6 Beniamino Accattoli and Delia Kesner. The structural *lambda*-calculus. In Anuj Dawar and Helmut Veith, editors, *CSL 2010*, volume 6247 of *LNCS*, pages 381–395. Springer, 2010. doi:10.1007/978-3-642-15205-4_30.

7 Beniamino Accattoli and Ugo Dal Lago. Beta reduction is invariant, indeed. In Thomas A. Henzinger and Dale Miller, editors, *CSL-LICS'14, Vienna, Austria, July 14-18, 2014*, pages 8:1–8:10. ACM, 2014. doi:10.1145/2603088.2603105.

8 Beniamino Accattoli and Ugo Dal Lago. (leftmost-outermost) beta reduction is invariant, indeed. *Logical Methods in Computer Science*, 12(1), 2016. doi:10.2168/LMCS-12(1:4)2016.

9 Beniamino Accattoli and Luca Paolini. Call-by-value solvability, revisited. In Tom Schrijvers and Peter Thiemann, editors, *FLOPS 2012, Kobe, Japan, May 23-25, 2012. Proceedings*, volume 7294 of *LNCS*, pages 4–16. Springer, 2012. doi:10.1007/978-3-642-29822-6_4.

10 Andrea Asperti and Stefano Guerrini. *The Optimal Implementation of Functional Programming Languages*. Cambridge Tracts in Theoretical Computer Science. CUP, 1999.

11 Thibaut Balabonski. Weak optimality, and the meaning of sharing. In Greg Morrisett and Tarmo Uustalu, editors, *ICFP'13, Boston, MA, USA – September 25-27, 2013*, pages 263–274. ACM, 2013. doi:10.1145/2500365.2500606.

12 Henk Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103. Elsevier, 1984.

13 Zine-El-Abidine Benaissa, Pierre Lescanne, and Kristoffer Høgsbro Rose. Modeling sharing and recursion for weak reduction strategies using explicit substitution. In Herbert Kuchen and S. Doaitse Swierstra, editors, *PLILP'96, Aachen, Germany, September 24-27, 1996, Proceedings*, volume 1140 of *LNCS*, pages 393–407. Springer, 1996. doi:10.1007/3-540-61756-6_99.

14 Gérard Boudol. Computational semantics of term rewriting systems. In M. Nivat and J.C. Reynolds, editors, *Algebraic Methods in Semantics*, page 169–236. CUP, 1985.

15 John R. W. Glauert and Zurab Khasidashvili. Relative normalization in deterministic residual structures. In Hélène Kirchner, editor, *CAAP'96, Linköping, Sweden, April, 22-24, 1996, Proceedings*, volume 1059 of *LNCS*, pages 180–195. Springer, 1996. doi:10.1007/3-540-61064-2_37.

16 Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. The geometry of optimal lambda reduction. In Ravi Sethi, editor, *POPL'92, Albuquerque, New Mexico, USA, January 19-22, 1992*, pages 15–26. ACM Press, 1992. doi:10.1145/143165.143172.

17 Delia Kesner. Reasoning about call-by-need by means of types. In *FoSSaCS 2016*, pages 424–441. Springer-Verlag, 2016.

**18** Jean-Jacques Lévy. *Réductions correctes et optimales dans le lambda-calcul*. PhD thesis, Université de Paris 7, 1978.

**19** Jean-Jacques Lévy. Generalized finite developments. In *Essays in Honour of Gilles Kahn*. CUP, 2007.

**20** Luc Maranget. Optimal derivations in weak lambda-calculi and in orthogonal terms rewriting systems. In David S. Wise, editor, *POPL'91, Orlando, Florida, USA, January 21-23, 1991*, pages 255–269. ACM Press, 1991. doi:10.1145/99583.99618.

**21** Paul-André Melliès. *Description Abstraite des Systèmes de Réécriture*. PhD thesis, Université Paris 7, december 1996.

**22** Paul-André Melliès. Axiomatic rewriting theory II: the $\lambda\sigma$-calculus enjoys finite normalisation cones. *J. Log. Comput.*, 10(3):461–487, 2000. doi:10.1093/logcom/10.3.461.

**23** Paul-André Melliès. Axiomatic rewriting theory VI residual theory revisited. In Sophie Tison, editor, *RTA 2002, Copenhagen, Denmark, July 22-24, 2002, Proceedings*, volume 2378 of *LNCS*, pages 24–50. Springer, 2002. doi:10.1007/3-540-45610-4_4.

**24** Robin Milner. Local bigraphs and confluence: Two conjectures: (extended abstract). *Electr. Notes Theor. Comput. Sci.*, 175(3):65–73, 2007. doi:10.1016/j.entcs.2006.07.035.

**25** Michael J. O'Donnell. *Computing in Systems Described by Equations*, volume 58 of *LNCS*. Springer, 1977. doi:10.1007/3-540-08531-9.

**26** H. J. Sander Bruggink. A proof of finite family developments for higher-order rewriting using a prefix property. In Frank Pfenning, editor, *RTA 2006*, volume 4098 of *LNCS*, pages 372–386. Springer, 2006. doi:10.1007/11805618_28.

**27** Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. CUP, 2003.

**28** Vincent van Oostrom. Finite family developments. In Hubert Comon, editor, *RTA-97, Sitges, Spain, June 2-5, 1997, Proceedings*, volume 1232 of *LNCS*, pages 308–322. Springer, 1997. doi:10.1007/3-540-62950-5_80.

**29** Vincent van Oostrom, Kees-Jan van de Looij, and Marijn Zwitserlood. ] (lambdascope). Workshop on Algebra and Logic on Programming Systems (ALPS), April 2004.