# The Structural $\lambda$-Calculus

Beniamino Accattoli and Delia Kesner

PPS (CNRS and Université Paris Diderot)

**Abstract.** Inspired by a recent graphical formalism for $\lambda$-calculus based on Linear Logic technology, we introduce an untyped structural $\lambda$-calculus, called $\lambda\mathtt{j}$, which combines action at a distance with exponential rules decomposing the substitution by means of weakening, contraction and dereliction. Firstly, we prove fundamental properties such as confluence and preservation of $\beta$-strong normalisation. Secondly, we use $\lambda\mathtt{j}$ to describe known notions of developments and superdevelopments, and introduce a more general one called $\mathtt{XL}$-development. Then we show how to reformulate Regnier's $\sigma$-equivalence in $\lambda\mathtt{j}$ so that it becomes a strong bisimulation. Finally, we prove that explicit composition or de-composition of substitutions can be added to $\lambda\mathtt{j}$ while still preserving $\beta$-strong normalisation.

## 1 Introduction

Computer science has been greatly influenced by Linear Logic [9], especially because it provides a mechanism to explicitly control the use of resources by limiting the liberal use of the *structural rules* of weakening and contraction. Erasure and duplication are restricted to formulas marked with an *exponential* modality?, and can only act on non-linear proofs marked with a bang modality !. Intuitionistic and Classical Logic can thus be encoded by a fragment containing such modalities, notably Multiplicative Exponential Linear Logic (MELL).

MELL proofs can be represented by sequent trees, but MELL Proof-Nets [9] provide a better-suited geometrical representation of proofs that eliminates irrelevant syntactical details. They have been extensively used to develop different encodings of intuitionistic logic/$\lambda$-calculus, giving rise to the geometry of interaction [10].

Normalisation of proofs (*i.e.* *cut elimination*) in MELL Proof-Nets is performed using three groups of rules, *multiplicative*, *exponential* and *commutative*. Non-linear proofs are distinguished by surrounding *boxes* which are handled by exponential rules: erasure, duplication and linear use correspond respectively to a cut elimination step involving a box and either a *weakening*, a *contraction* or a *dereliction*. The commutative rule allows to *compose* non-linear resources.

Different cut elimination systems [7,15,13], called *explicit substitution* (ES) calculi, were explained in terms of, or inspired by, the notion of reduction of MELL Proof-Nets. All of them use the idea that the content of a substitution/cut is a non-linear resource, *i.e.* a box that can be composed with another one by means of commutative rules. They also have in common an operational semantics defined in terms of a *propagation system* in which a substitution traverses a term to reach the variable occurrences.

© Springer-Verlag Berlin Heidelberg 2010

A graph formalism for $\lambda$-terms inspired by Intuitionistic MELL has recently been proposed [1]. It avoids boxes by representing them through additional edges called *jumps*, and has no commutative reduction rule. This paper studies the term formalism, called $\lambda\mathtt{j}$-calculus, resulting from the reading back of the graphs $\lambda\mathtt{j}$-dags (and their reductions) by means of their sequentialisation theorem [1].

No rule of $\lambda\mathtt{j}$ propagates cuts, as the constructors in a term interact *at a distance*, *i.e.* they work modulo positions of cuts. Action at a distance is not a complete novelty [21,5,22], but none of the previous approaches faithfully reflect resource control as suggested by Linear Logic. We propose to recognise such behaviour as a new paradigm, more primitive than ES, particularly because propagations can be added on top of action at a distance (as we shall show). Despite the absence of commutative rules in $\lambda\mathtt{j}$, cuts can be composed, but in a different (more natural) way.

Similarly to formalisms [16] inspired by Proof-Nets, cut elimination is defined in terms of the number of free occurrences of variables in a term, here called *multiplicities.* More precisely, the weakening-box rule (resp. dereliction-box and contraction-box) applies to terms that are of the form $t[x/u]$ when $|t|_x = 0$ (resp. $|t|_x = 1$ and $|t|_x > 1$). The computation is, however, performed without propagating $[x/u]$, which we call a *jump* to stress that such action at a distance is really different from propagation in ES calculi. The rules of $\lambda\mathtt{j}$ therefore combine action at a distance, due to the tight correspondance with a graphical formalism, with exponential rules, due to the strong affinity with Linear Logic. Because of the weakening and contraction rules we call our language the *structural* $\lambda$-calculus.

Some calculi using either distance or multiplicities already exist, but without combining the two: only together those concepts unleash their full expressive power. Indeed, [5,22] use distance rules to refine $\beta$-reduction, but add ES to the syntax without distinguishing between dereliction and contraction. This causes the formalism to be less expressive than $\lambda\mathtt{j}$ as discussed in Sections 4 and 6. Milner defines a $\lambda$-calculus with ES inspired by another graphical formalism, Bigraphs [21], where cuts also act at a distance. Again, he neither distinguishes between dereliction and contraction, nor does his $\beta$-rule exploit distance. The same goes for [28,23].

This paper studies the $\lambda\mathtt{j}$-calculus focusing on four different aspects:

– **Basic properties:** Section 2 presents the calculus while Section 3 shows full composition, simulation of one-step $\beta$-reduction, confluence, and preservation of $\beta$-strong normalisation (PSN). Particularly, we prove PSN using a modular technique [14], which results in an extremely short formal argument thanks to the absence of propagations.
– **Developments:** The $\lambda\mathtt{j}$-calculus is a powerful, elegant and concise tool for studying $\beta$-reduction. As an example, in Section 4 we analyse the redex creation mechanism of $\lambda$-calculus, using normal forms of certain subsystems of $\lambda\mathtt{j}$ to characterise the result of full developments [12] and full superdevelopments [17]. By adding *more distance* to the previous subsystems, we characterise the result of a new, more powerful notion of development, which we call $\mathtt{XL}$-development.
– **Operational equivalence:** Section 5 studies an operational equivalence $\equiv_o$ which equates $\lambda\mathtt{j}$-terms behaving the same way but differing only in

the positioning of their jumps. The relation $\equiv_o$ includes a reformulation of Regnier's $\sigma$-equivalence [26], but also contains commutation for independent jumps. We show that $\equiv_o$ is a strong bisimulation on $\lambda\mathtt{j}$-terms. Interestingly, this result holds only because of distance.

– **(De)composition of jumps:** In Section 6 we consider two further extensions of the system devised in Section 5, including, respectively, explicit composition and decomposition of jumps. We prove both new reduction relations to be confluent modulo $\equiv_o$ and to enjoy PSN. The two systems, reintroducing some propagation rules, bridge the gap with traditional ES calculi and implementations. The PSN proofs in this section are the more technically demanding proofs of this paper, and a non-trivial contribution to the theory of termination proofs of ES calculi.

A technical report including detailed proofs can be found in [2].

## 2 The Calculus

The set $\mathcal{T}$ of **terms** is defined by the following grammar:

$$t ::= x \text{ (variable)} \mid \lambda x.t \text{ (abstraction)} \mid t\ t \text{ (application)} \mid \mid t[x/t] \text{ (closure)}$$

The object $[x/t]$, which is not a term itself, is called a **jump**. A term without jumps is a **$\lambda$-term**. We use the notation $\overline{v}_n^1$ for a list of terms $v_1 \ldots v_n$, $t\ \overline{v}_n^1$ for $(\ldots(t\ v_1)\ldots v_n)$ and $t[x_i/u_i]_n^1$ for $t[x_1/u_1]\ldots[x_n/u_n]$ $(n \geq 0)$.

**Free** and **bound** variables of $t$, respectively written $\mathtt{fv}(t)$ and $\mathtt{bv}(t)$, are defined as usual. The constructors $\lambda x.u$ and $u[x/v]$ bind the free occurrences of $x$ in $u$. The congruence generated by renaming of bound variables is called **$\alpha$-conversion**. Thus for example $(\lambda y.x)[x/y] =_\alpha (\lambda y'.x')[x'/y]$.

The **multiplicity of the variable $x$ in the term $t$** is defined as the number of free occurrences of $x$ in $t$, written $|t|_x$. We use $|t|_\Gamma$ for $\Sigma_{x\in\Gamma}|t|_x$. When $|t|_x = n \geq 2$, we write $t_{[y]_x}$ for the **non-deterministic replacement** of $i$ $(1 \leq i \leq n-1)$ occurrences of $x$ in $t$ by a *fresh* variable $y$. Thus, $(x\ z)[z/x]_{[y]_x}$ may denote either $(y\ z)[z/x]$ or $(x\ z)[z/y]$ but not $(y\ z)[z/y]$.

A **(meta-level) substitution** is a finite function from variables to terms. **Substitution** is defined, as usual, modulo $\alpha$-conversion so that the capture of variables is avoided. We use $t\sigma$ to denote the **application** of the **meta-level substitution** $\sigma$ to the term $t$. Thus for example $(\lambda y.x)\{x/y\} = \lambda z.y$. Moreover, $t\{x/u\} = t$ if $x \notin \mathtt{fv}(t)$. We use juxtaposition of substitutions to denote composition so that $\tau\sigma$ is the substitution given by $x(\tau\sigma) := (x\tau)\sigma$. Besides $\alpha$-conversion, we consider the following rewriting rules:

$$\begin{array}{llll}
(\mathtt{dB}) & (\lambda x.t)\mathtt{L}\ u \rightarrow t[x/u]\mathtt{L} & & \\
(\mathtt{w}) & t[x/u] & \rightarrow t & \text{if } |t|_x = 0 \\
(\mathtt{d}) & t[x/u] & \rightarrow t\{x/u\} & \text{if } |t|_x = 1 \\
(\mathtt{c}) & t[x/u] & \rightarrow t_{[y]_x}[x/u][y/u] & \text{if } |t|_x \geq 2
\end{array}$$

where we use the (meta)notation $\mathtt{L}$ for a list $[x_i/u_i]_n^1$ with $n \geq 0$.

Note that $\mathtt{dB}$ reformulates the classical $\mathtt{B}$-rule of ES calculi as a *distance* rule which skips the jumps affecting the abstraction of the redex. This same rule notably appears in weak ES calculi [19] to avoid the the $\beta$-redexes that are hidden by blocked substitutions. Here, the $\mathtt{dB}$-rule is the natural term counterpart of a graphical and *local* rule in proof-nets and $\lambda j$-dags. Section 4 puts the expressiveness of this concept in evidence. The rules $\mathtt{w}$, $\mathtt{d}$ and $\mathtt{c}$ are to be understood as the $\mathtt{w}$eakening, $\mathtt{d}$ereliction and $\mathtt{c}$ontraction rules in $\lambda j$-dags.

It is worth noting that $\lambda\mathtt{j}$ allows to *compose* jumps, as for example reduction from $t = y[x/zy][y/v]$ *computes* the (simultaneous) jumps in $y[x/zv][y/v]$. Usually, the so-called *composition* of the two jumps of $t$ rather yields $y[y/v][x/zy[y/v]]$. We will study this more structural notion in Section 6.

The **rewriting relation** $\rightarrow_{\lambda\mathtt{j}}$ (resp. $\rightarrow_{\mathtt{j}}$) is generated by all (resp. all expect $\mathtt{dB}$) the previous rewriting rules modulo $\alpha$-conversion. The $\mathtt{j}$-rewriting rules are based on global side conditions, which may seem difficult to implement. However, if implementation is done via graphical formalisms (such as proof-nets, bigraphs, $\lambda j$-dags), these conditions become local and completely harmless.

Now consider any reduction relation $\mathcal{R}$. A term $t$ is said to be in **$\mathcal{R}$-normal form**, written $\mathcal{R}$-nf, if there is no $u$ so that $t \rightarrow_{\mathcal{R}} u$. We use $\mathcal{R}(t)$ to denote the *unique* $\mathcal{R}$-nf of $t$, when it exists. A term $t$ is **$\mathcal{R}$-strongly normalising** or **$\mathcal{R}$-terminating**, written $t \in \mathcal{SN}_{\mathcal{R}}$, if there is no infinite $\mathcal{R}$-reduction sequence starting at $t$, in which case $\eta_{\mathcal{R}}(t)$ denotes the **maximal length of a $\mathcal{R}$-reduction sequence starting at** $t$. The relation $\mathcal{R}$ is called **complete** if it is strongly normalising and confluent.

## 3 Main Properties

In this section we prove some sanity properties of the calculus: full composition, simulation of one-step $\beta$-reduction, confluence and PSN. Since the first three can easily be shown using standard rewriting technology, we concentrate on proving PSN, which usually is tricky, but turns out to be surprisingly simple in our case. By induction on $|t|_x$ (and not on $t$!) we get the following lemma:

**Lemma 1 (Full Composition (FC)).** *Let $t, u \in \mathcal{T}$. Then $t[x/u] \rightarrow^+_{\mathtt{j}} t\{x/u\}$.*

**Corollary 1 (Simulation).** *Let $t \in \lambda$-term. If $t \rightarrow_\beta t'$, then $t \rightarrow^+_{\lambda\mathtt{j}} t'$.*

The following notion, which counts the maximal number of free occurrences of a variable $x$ that may appear during a $\mathtt{j}$-reduction sequence from a term $t$, will be useful for various proofs. The **potential multiplicity** of the variable $x$ in the term $t$, written $\mathtt{M}_x(t)$, is defined for $\alpha$-equivalence classes as follows: if $x \notin \mathtt{fv}(t)$, then $\mathtt{M}_x(t) := 0$; otherwise:

$$\begin{array}{llll} \mathtt{M}_x(x) & := 1 & \mathtt{M}_x(u\ v) & := \mathtt{M}_x(u) + \mathtt{M}_x(v) \\ \mathtt{M}_x(\lambda y.u) & := \mathtt{M}_x(u) & \mathtt{M}_x(u[y/v]) & := \mathtt{M}_x(u) + \mathtt{max}(1, \mathtt{M}_y(u)) \cdot \mathtt{M}_x(v) \end{array}$$

**Lemma 2.** *The $\mathtt{j}$-reduction relation is complete.*

*Proof.* For each $t \in \mathcal{T}$ define a finite multiset of natural numbers $\mathtt{jm}(t)$ given by:

$$\begin{array}{llll} \mathtt{jm}(x) & := [\,] & \mathtt{jm}(tu) & := \mathtt{jm}(t) \sqcup \mathtt{jm}(u) \\ \mathtt{jm}(\lambda x.t) & := \mathtt{jm}(t) & \mathtt{jm}(t[x/u]) & := [\mathtt{M}_x(t)] \sqcup \mathtt{jm}(t) \sqcup \mathtt{max}(1, \mathtt{M}_x(t)) \cdot \mathtt{jm}(u) \end{array}$$

where $\sqcup$ is multiset union and $n \cdot [a_1, \ldots, a_n] := [n \cdot a_1, \ldots, n \cdot a_n]$. Then show that $\rightarrow_{\mathtt{j}}$ strictly decreases $\mathtt{jm}(\_)$ and thus conclude termination. Local confluence (straightforward) and termination imply confluence by Newman's Lemma.

This is used to prove confluence of $\rightarrow^*_{\lambda\mathtt{j}}$, by means of the Tait and Martin-Löf technique (details in [2]):

**Theorem 1 (Confluence).** *For all $t, u_1, u_2 \in \mathcal{T}$, if $t \rightarrow^*_{\lambda\mathtt{j}} u_i$ $(i = 1, 2)$, then $\exists v$ s.t. $u_i \rightarrow^*_{\lambda\mathtt{j}} v$ $(i = 1, 2)$.*

We now discuss PSN. A reduction system $\mathcal{R}$ is said to enjoy the **PSN property** w.r.t. another system $\mathcal{S}$ iff every term which is $\mathcal{S}$-strongly normalising is also $\mathcal{R}$-strongly normalising. Here PSN will mean PSN w.r.t. $\beta$-reduction.

The proof of PSN can be stated in terms of the **IE** property which relates termination of **I**mplicit substitution to termination of **E**xplicit substitution. A reduction system $\mathcal{R}$ enjoys the **IE property** iff for $n \geq 0$ and for all $t, u, \overline{v}^1_n \in \lambda$-terms: $u \in \mathcal{SN}_{\mathcal{R}}$ and $t\{x/u\}\overline{v}^1_n \in \mathcal{SN}_{\mathcal{R}}$ imply $t[x/u]\overline{v}^1_n \in \mathcal{SN}_{\mathcal{R}}$.

**Theorem 2 (IE implies PSN).** *A reduction relation $\mathcal{R}$ enjoys PSN if $\mathcal{R}$ verifies the* **IE***-property and the following:*

**(F0)** *If $\overline{t}^1_n \in \lambda$-terms in $\mathcal{SN}_{\mathcal{R}}$, then $x\overline{t}^1_n \in \mathcal{SN}_{\mathcal{R}}$.*
**(F1)** *If $u \in \lambda$-term in $\mathcal{SN}_{\mathcal{R}}$, then $\lambda x.u \in \mathcal{SN}_{\mathcal{R}}$.*
**(F2)** *The only $\mathcal{R}$-reducts of a $\lambda$-term $(\lambda x.u)v\overline{t}^1_n$ are $u[x/v]\overline{t}^1_n$ and those coming from internal reduction on $u, v, \overline{t}^1_n$.*

Intuitively, the first two requirements (F0) and (F1) mean that head-normal forms are stable under $\mathcal{R}$. The last requirement (F2) means that the head-redex can only be refined by $\mathcal{R}$, but nothing else.

*Proof.* We show $t \in \mathcal{SN}_{\mathcal{R}}$ by induction on the definition of $t \in \mathcal{SN}_\beta$ (as in [29]):

– If $t = x\overline{t}^1_n$ with $t_i \in \mathcal{SN}_\beta$, then (i.h.) $t_i \in \mathcal{SN}_{\mathcal{R}}$ and thus (**F0**) $x\overline{t}^1_n \in \mathcal{SN}_{\mathcal{R}}$.
– If $t = \lambda x.u$ with $u \in \mathcal{SN}_\beta$, then (i.h.) $u \in \mathcal{SN}_{\mathcal{R}}$ and thus (**F1**) $\lambda x.u \in \mathcal{SN}_{\mathcal{R}}$.
– If $t = (\lambda x.u)v\overline{t}^1_n$, with $u\{x/v\}\overline{t}^1_n \in \mathcal{SN}_\beta$ and $v \in \mathcal{SN}_\beta$, then (i.h.) both terms are in $\mathcal{SN}_{\mathcal{R}}$, **IE** gives $U = u[x/v]\overline{t}^1_n \in \mathcal{SN}_{\mathcal{R}}$, so in particular $u, v, \overline{t}^1_n \in \mathcal{SN}_{\mathcal{R}}$. We show $t \in \mathcal{SN}_{\mathcal{R}}$ by induction on $\eta_{\mathcal{R}}(u) + \eta_{\mathcal{R}}(v) + \Sigma_i\ \eta_{\mathcal{R}}(t_i)$. For that, we show that every $\mathcal{R}$-reduct of $t$ is in $\mathcal{SN}_{\mathcal{R}}$.
  Now, if $t \rightarrow_{\mathcal{R}} t'$ is an internal reduction, apply the i.h. Otherwise, **F2** gives $t \rightarrow_{\mathcal{R}} u[x/v]t_1 \ldots t_n = U$ which is in $\mathcal{SN}_{\mathcal{R}}$.

**Theorem 3 (IE for $\lambda\mathtt{j}$).** *$\lambda\mathtt{j}$ enjoys the* **IE** *property.*

*Proof.* We show the following more general statement. For all terms $t, \overline{u}^1_m$ $(m \geq 1), \overline{v}^1_n$ $(n \geq 0)$, if $\overline{u}^1_m \in \mathcal{SN}_{\lambda\mathtt{j}}$ & $t\{x_i/u_i\}^1_m\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j}}$, then $t[x_i/u_i]^1_m\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j}}$, where $x_i \neq x_j$ for $i, j = 1 \ldots m$ and $x_i \notin \mathtt{fv}(u_j)$ for $i, j = 1 \ldots m$. The **IE** property holds by taking $m = 1$. Now, we prove $t[x_i/u_i]^1_m\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j}}$, by induction on the 3-tuple $\langle \eta_{\lambda\mathtt{j}}(t\{x_i/u_i\}^1_m\overline{v}^1_n), \mathtt{o}_{\overline{x}^1_m}(t), \eta_{\lambda\mathtt{j}}(\overline{u}^1_m)\rangle$ where $\mathtt{o}_{x_i}(t) = 3^{|t|_{x_i}}$ and $\mathtt{o}_{\overline{x}^1_m}(t) = \Sigma_i\ \mathtt{o}_{x_i}(t)$. Details in [2].

In contrast to known PSN proofs for calculi with ES and composition of substitutions [4,13,15], we get a very concise and simple proof of the **IE** property, and thus of PSN, due to the fact that $\lambda\mathtt{j}$ has no propagation rule. Indeed, since $\lambda\mathtt{j}$-reduction enjoys the **IE**-property and **F0**, **F1** and **F2** in Theorem 2 are straightforward for the $\lambda\mathtt{j}$-calculus, we get:

**Corollary 2 (PSN for $\lambda\mathtt{j}$).** *Let $t \in \lambda$-term. If $t \in \mathcal{SN}_\beta$, then $t \in \mathcal{SN}_{\lambda\mathtt{j}}$.*

## 4 Developments and All That

In $\lambda$-calculus creation of redexes can be classified in three types [18]:

**(Type 1)** $((\lambda x.\lambda y.t)\ u)\ v \rightarrow_\beta (\lambda y.t\{x/u\})\ v$.
**(Type 2)** $(\lambda x.x)\ (\lambda y.t)\ u \rightarrow_\beta (\lambda y.t)\ u$.
**(Type 3)** $(\lambda x.C[x\ v])\ (\lambda y.u) \rightarrow_\beta C\{x/\lambda y.u\}[(\lambda y.u)\ v]$

When $\lambda$-terms are considered as trees, the first and second type create a redex *upward*, while the third creates it *downward*, which is the dangerous kind of creation since it may lead to divergence.

According to the previous classification, different ways to compute a term can be defined. A reduction sequence starting at $t$ is a **development** [12] (resp. a **full development**) if only (resp. all the) residuals of redexes (resp. all the redexes) of $t$ are contracted. A more liberal notion, called **L-development** here, and known as superdevelopment [17], allows to also reduce created redexes of type 1 and 2. A major result states that all developments (resp. L-developments) of a $\lambda$-term are finite, and that the results of all full developments (resp. full L-developments) coincide.

Note that reductions of type 1 and 2 are acceptable because the created redex is *hidden* in the initial term, so that non-termination only happens when creating redexes of type 3. However, *linear creations* of type 3 - *i.e.* creations which do not involve duplications - are also safe, and infinite reductions only happen if redexes created after duplication are reduced - we call such cases *non-linear creations* of type 3. As an example, consider $\Omega = (\lambda x.x\ x)\ (\lambda x.x\ x)$ whose infinite reduction involves only non-linear creations of the third type. These observations suggest that banning the third type of creation is excessive: it is sufficient to avoid non-linear ones. This extended form of L-development needs a language capable of distinguishing the linear/erasing/duplicating nature of redexes. This section extends the notion of L-development to that of **XL-development**, which also reduces linearly created redexes of type 3, and provides a finiteness result.

The following table summarises the behaviour of each computational notion studied in this section on the $\lambda$-term $u_0 = (I\ I)\ ((\lambda z.z\ y)\ I)$, where $I = \lambda x.x$.

$$\begin{array}{ll} \text{full development of } u_0 & = I\ (I\ y) \\ \text{full L-development of } u_0 & = I\ y \\ \text{full XL-development of } u_0 & = y \end{array} \qquad (1)$$

The specification of all the reduction subsystems used in this section exploits the idea of *multiplicity*. Thus, the $\lambda$j-calculus provides a uniform and expressive framework to reason about creation of redexes in $\lambda$-calculus.

A **development** (resp. full development) of a term $t$ is a reduction sequence in which only (resp. all the) residuals of redex occurrences (resp. all the redex occurrences) that already exist in $t$ are contracted. There are many proofs of finiteness of developments, like [12,29]. The **result of a full development** of a $\lambda$-term is unique and can simply be defined by induction on the structure of terms as follows:

$$\begin{array}{llll} x^\circ & := x & ((\lambda x.t)\ u)^\circ & := t^\circ\{x/u^\circ\} \\ (\lambda x.t)^\circ & := \lambda x.t^\circ & (t\ u)^\circ & := t^\circ\ u^\circ \quad \text{if } t \neq \lambda \end{array}$$

But in $\lambda\mathtt{j}$ it can also be characterised in a more operational way. Let $\mathtt{B}$ be the rewriting rule $(\lambda x.t)u \rightarrow_{\mathtt{B}} t[x/u]$, which is the restriction of our $\mathtt{dB}$-rule to a *proximity* action. This relation is trivially *complete* so that we use $\mathtt{B}(t)$ for the (unique) $\mathtt{B}$-nf of the term $t$. By induction on $t$ we get the following corollary:

**Corollary 3.** *Let $t \in \lambda$-term. Then $t^{\circ} = \mathtt{j}(\mathtt{B}(t))$.*

Developments can be extended to **$\mathtt{L}$-developments** which also reduce created redexes of type 1 and 2 and are always finite. The **result of a full $\mathtt{L}$-development** of a $\lambda$-term is unique and admits the following inductive definition [17]:

$$\begin{array}{llll} x^{\circ\circ} := x & \qquad (t\ u)^{\circ\circ} := t^{\circ\circ}\ u^{\circ\circ} & \text{if } t^{\circ\circ} \neq \lambda \\ (\lambda x.t)^{\circ\circ} := \lambda x.t^{\circ\circ} & \qquad (t\ u)^{\circ\circ} := t_1\{x/u^{\circ\circ}\} & \text{if } t^{\circ\circ} = \lambda x.t_1 \end{array}$$

Remark that $t^{\circ\circ} \neq \lambda$ implies $t \neq \lambda$.

Let us recover $t^{\circ\circ}$ by means of our language $\lambda\mathtt{j}$. The key to operationally describe the first type of creation is the *distance* $\mathtt{dB}$-rule, whose (unique) nf will be noted $\mathtt{dB}(t)$. Replacing our definition of development $\mathtt{j}(\mathtt{B}(t))$ with $\mathtt{j}(\mathtt{dB}(t))$ gives:

$$\mathtt{dB}(((\lambda x.\lambda y.t)\ u)\ v) = \mathtt{dB}(t)[y/\mathtt{dB}(v)][x/\mathtt{dB}(u)] = M$$

Then, computing jumps, we get:

$$\mathtt{j}(M) = \mathtt{j}(\mathtt{dB}(t))\{x/\mathtt{j}(\mathtt{dB}(u))\}\{y/\mathtt{j}(\mathtt{dB}(v))\}$$

And we are done. Now, to specify $\mathtt{L}$-developments within our language $\lambda\mathtt{j}$ we also need to capture the second type of creation. We would therefore need to use $\mathtt{dB} \cup \mathtt{d} \cup \mathtt{w}$ instead of $\mathtt{dB}$, but our (distance) $\mathtt{d}$-rule turns out to be too powerful since created redexes of type 3 would also be captured as shown by the term $(\lambda x.x\ t)(\lambda y.u)$, where $x \notin \mathtt{fv}(t)$. Thus, the reduction $\mathtt{d}$ is restricted to act only on variables, written $\mathtt{md}$ (for *minimal dereliction*), so that $\rightarrow_{\mathtt{md}}$ is the context closure of the rule $x[x/u] \rightarrow u$. We then let $\mathtt{A}$ be the relation $\mathtt{dB} \cup \mathtt{md} \cup \mathtt{w}$.

**Lemma 3.** *The reduction relation $\rightarrow_{\mathtt{A}}$ is complete.*

*Proof.* Termination of $\mathtt{A}$ is straightforward. Confluence follows from local confluence (straightforward by case-analysis) and Newman's Lemma.

Interestingly, $\rightarrow_{\mathtt{A}}$ cannot be weakened to $\rightarrow_{\mathtt{dB}\cup\mathtt{md}}$ as illustrated by the term $s = ((\lambda x.((\lambda y.x)\ t))\ \lambda z.z)\ u$. Due to lack of space the proof of the following proposition has been omitted (details in [2]):

**Proposition 1.** *Let $t$ be a $\lambda$-term. Then $t^{\circ\circ} = \mathtt{j}(\mathtt{A}(t))$.*

It is now natural to relax the previous relation $\mathtt{A}$ from $\mathtt{dB} \cup \mathtt{md} \cup \mathtt{w}$ to $\mathtt{dB} \cup \mathtt{d} \cup \mathtt{w}$, in other words, to also allow unrestricted $\mathtt{d}$-steps. Thus $\mathtt{L}$-developments are extended to **$\mathtt{XL}$-developments**, which also allow linear creations of type 3. Completeness of this extended notion is stated by the following lemma, proved similarly to Lemma 3.

**Lemma 4.** *The reduction relation $\rightarrow_{\mathtt{dB}\cup\mathtt{d}\cup\mathtt{w}}$ is complete.*

The result of a **full $\mathtt{XL}$-development** of a $\lambda$-term $t$, noted $t^{\circ\circ\circ}$, is defined by $\mathtt{j}((\mathtt{dB} \cup \mathtt{d} \cup \mathtt{w})(t))$ where $(\mathtt{dB} \cup \mathtt{d} \cup \mathtt{w})(t)$ denotes the (unique) $(\mathtt{dB} \cup \mathtt{d} \cup \mathtt{w})$-nf of $t$. This notion extends $\mathtt{L}$-developments in a deterministic way, *i.e.* provides a complete reduction relation for $\lambda$-terms, more liberal than $\mathtt{L}$-developments.

It is well known that every **affine** $\lambda$-term $t$ (*i.e.* a term where no variable has more than one occurrence in $t$) is $\beta$-strongly normalising (the number of constructors strictly diminishes with each step). Moreover, $\beta$-reduction of affine terms can be performed in $\lambda\mathtt{j}$ using only $\mathtt{dB} \cup \mathtt{d} \cup \mathtt{w}$, *i.e.* $\beta$-nf$(t) = (\mathtt{dB} \cup \mathtt{d} \cup \mathtt{w})(t)$. Thus:

**Corollary 4.** *Let $t$ be an affine $\lambda$-term. Then $t^{\circ\circ\circ} = \beta$-nf$(t)$.*

We hope that our extended notion of $\mathtt{XL}$-development can be applied to obtain more expressive solutions for higher-order matching problems, which arise for example in higher-order logic programming, logical frameworks, program transformations, etc. Indeed, the approach of higher-order matching in *untyped* frameworks [8,6], which currently uses $\mathtt{L}$-developments, may be improved using $\mathtt{XL}$-developments, as suggested by example (1) at the beginning of this section.

## 5 Bisimilar Terms

The simplicity of the $\lambda\mathtt{j}$-calculus naturally suggests the study of some operational equivalence which should equate terms that differ only concerning the positioning of their jumps but *behave identically.* For instance, if $y \notin \mathtt{fv}(u)$, then $\lambda y.t[x/u]$ and $(\lambda y.t)[x/u]$ behave equivalently: there is a bijection between their redexes and their reducts, *i.e.* they are *bisimilar.* This idea is reminiscent of Regnier's equivalence on $\lambda$-terms [25], here written $\sigma^R$:

$$\begin{array}{lll} (\lambda x.\lambda y.t)\ u \equiv_{\sigma_1^R} \lambda y.((\lambda x.t)\ u) & & \text{if } y \notin \mathtt{fv}(u) \\ (\lambda x.t\ v)\ u \ \ \equiv_{\sigma_2^R} (\lambda x.t)\ u\ v & & \text{if } x \notin \mathtt{fv}(v) \end{array}$$

Reduction of the $\mathtt{dB}$-redexes in the previous equations yields the following $\sigma$-equivalence notion, now on $\lambda\mathtt{j}$-terms:

$$\begin{array}{lll} (\lambda y.t)[x/u] \equiv_{\sigma_1} \lambda y.t[x/u] & \text{if } y \notin \mathtt{fv}(u) \\ (t\ v)[x/u] \ \ \equiv_{\sigma_2} t[x/u]\ v & \text{if } x \notin \mathtt{fv}(v) \end{array}$$

This is not very surprising since $\sigma^R$-equivalence was introduced by noting that the two terms of each equation represent the *same* MELL proof-net modulo multiplicative redexes, which correspond exactly to the $\mathtt{dB}$-redexes of the $\lambda\mathtt{j}$-calculus. Regnier proved that $\sigma^R$-equivalent terms have the same maximal $\beta$-reduction length. However, this does not imply that $\sigma^R$-equivalence is a strong bisimilarity on $\lambda$-terms. Indeed, take $\lambda$-terms $t_0 = ((\lambda x.\lambda y.y)\ z)\ w \equiv_{\sigma_1^R} (\lambda y.((\lambda x.y)\ z))\ w = t_1$. Both share the same $\beta$-normal form $w$ and $\eta_\beta(t_0) = \eta_\beta(t_1)$. Nevertheless, $t_0$ has one redex, while $t_1$ has two redexes, and the redex of $t_1$ involving $w$ has no corresponding redex in $t_0$. They also differ in terms of creation of redexes: the result of the full development of $t_0$ has a *created* redex, while the result of the full development of $t_1$ is the normal form of the term. Our reformulation of $\sigma^R$, however, equates two $\lambda\mathtt{j}$-terms $t'_0$ and $t'_1$ which are strongly bisimilar:

$$t_0 \rightarrow_{\mathtt{dB}} t'_0 = (\lambda y.y)[x/z]\ w \quad \equiv_{\sigma_1} \quad (\lambda y.y[z/x])\ w = t'_1\ {}_{\mathtt{dB}}\!\leftarrow t_1 \qquad (2)$$

Actually, bisimulation holds also for permutation of *independent* jumps [13]:

$$t[x/u][y/v] \equiv_{\mathtt{CS}} t[y/v][x/u] \qquad \text{if } y \notin \mathtt{fv}(u) \ \& \ x \notin \mathtt{fv}(v)$$

While CS should naturally remain an equivalence, $\sigma$ has often been restricted to being considered a *reduction* relation [26], for no good reason. Here, we add CS and $\sigma$ to $\lambda\mathtt{j}$ without any trouble, in particular without loosing the PSN property (Corollary 8). The **operational equivalence** relation generated by $\mathtt{o} = \{\alpha, \mathtt{CS}, \sigma_1, \sigma_2\}$ realises a strong bisimulation, proved by induction on $\equiv_{\mathtt{o}}$:

**Proposition 2 (Strong Bisimulation).** *For all $t, u, u' \in \mathcal{T}$ s.t. $t \equiv_{\mathtt{o}} u \to_{\lambda\mathtt{j}} u'$ $\exists t'$ s.t. $t \to_{\lambda\mathtt{j}} t' \equiv_{\mathtt{o}} u'$.*

Such bisimulation implies that two o-equivalent terms share the same maximal reduction length. Moreover, the strong bisimulation would not hold without distance rules. Indeed, the two $\sigma_1$-equivalent terms $t'_0$ and $t'_1$ in (2) do not have the same B-redexes but the same dB-redexes.

## 6 (De)composing Substitutions

Explicit substitution (ES) calculi may or may not include rewriting rules to *explicitly* compose substitutions. One often adds them to recover confluence on terms with metavariables. However, naïve rules may break the PSN property, so that *safe* composition rules are needed to recover both PSN and confluence on terms with metavariables [13]. The $\lambda\mathtt{j}$-calculus is peculiar as it allows to compose substitutions, but only *implicitly.* Indeed, a term $t[x/u][y/v]$ s.t. $y \in \mathtt{fv}(u)$ & $y \in \mathtt{fv}(t)$ reduces in various steps to $t[x/u\{y/v\}][y/v]$, but not to the explicit composition $t[x/u[y/v]][y/v]$. One of the aims of this section is adding *explicit composition* to $\lambda\mathtt{j}$ keeping PSN and confluence.

The second aim of this section concerns *explicit decomposition.* Indeed, some calculi [24,20,27,11] explicitly *decompose* substitutions, *i.e.* reduce $t[x/u[y/v]]$ to $t[x/u][y/v]$. We show that even in such a case PSN and confluence still hold.

Composition (boxing) and decomposition (unboxing) are dual systems:

| The **Boxing** system b | The **Unboxing** system u |
|---|---|
| if $x \notin \mathtt{fv}(t)$ & $x \in \mathtt{fv}(v)$ : | if $x \notin \mathtt{fv}(t)$ & $x \in \mathtt{fv}(v)$ : |
| $(t\ v)[x/u] \to_{\mathtt{ab}} t\ v[x/u]$ | $t\ v[x/u] \to_{\mathtt{au}} (t\ v)[x/u]$ |
| $t[y/v][x/u] \to_{\mathtt{sb}} t[y/v[x/u]]$ | $t[y/v[x/u]] \to_{\mathtt{su}} t[y/v][x/u]$ |

The boxing system reflects the commutative box-box rule of Linear Logic, the unboxing system is obtained by reversing its rules. Moreover, we consider the system modulo the o-equivalence. Choosing a particular orientation for $\sigma_1$ and $\sigma_2$ leads to a full set of propagating rules, that is, something closer to traditional ES calculi. We prefer, however, to work modulo an equivalence to obtain a more general result. Remark that the constraint $x \notin \mathtt{fv}(t)$ for the unboxing rules does not limit their applicability, as it can always be satisfied through $\alpha$-equivalence.

*Digression.* It is natural to wonder if one could also work modulo (de)composition, *i.e.* adding two more general axioms:

$$\begin{array}{lll} (t\ v)[x/u] & \equiv_{\sigma_3} t\ v[x/u] & \text{if } x \notin \mathtt{fv}(t) \\ t[y/v][x/u] & \equiv_{\sigma_4} t[y/v[x/u]] & \text{if } x \notin \mathtt{fv}(t) \end{array}$$

The answer is no, as these last two congruences break the PSN property, if naïvely added. For example: let $u = (z\ z)[z/y]$, then

$$\begin{array}{lll} t = u[x/u] = (z\ z)[z/y][x/u] & \equiv_{\sigma_4} (z\ z)[z/y[x/u]] & \rightarrow_{\mathtt{c}} \\ (z_1\ z_2)[z_1/y[x/u]][z_2/y[x/u]] & \rightarrow^{+}_{\mathtt{d}} y[x/u]\ (y[x/u]) & \equiv_{\sigma_2,\sigma_3,\alpha} \\ (y\ y)[x_1/u][x/u] & \equiv_{\sigma_4} (y\ y)[x_1/u[x/u]] & \end{array}$$

*i.e.* $t$ reduces to a term containing $t$. Now, take $(\lambda x.((\lambda z.zz)y))\ ((\lambda z.zz)y) \in \mathcal{SN}_\beta$ which reduces to $t$, so that it is no longer strongly normalising in the $\lambda\mathtt{j}$-calculus extended by the five previous equations $\{\mathtt{CS}, \sigma_1, \sigma_2, \sigma_3, \sigma_4\}$.

Such a counter-example can be avoided imposing the constraint "$x \in \mathtt{fv}(v)$" to $\sigma_3$ and $\sigma_4$ (note that such constraint is also found in the definition of the boxing system). Nevertheless, $\lambda\mathtt{j}$-reduction modulo the *constrained* equivalences $\{\mathtt{CS}, \sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ is an incredibly subtle and complex relation. For instance, $\mathtt{w}$-steps cannot be postponed, nor can the use of equivalences. Two natural canonical representations of the equivalence classes are obtained by pushing jumps towards the variables, or as far away from them as possible. None of them is stable by reduction, so working with equivalence classes is impossible. The PSN property for this calculus, if it holds, is very challenging.

One of the difficulties is that the equivalence $\{\mathtt{CS}, \sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ is not a bisimulation: the reducts $(xx_1)[x/y[y/z]][x_1/y[y/z]]$ of $t_2 = (xx)[x/y[y/z]]$ and $(xx_1)[x/y][x_1/y][y/z]$ of $t_3 = (xx)[x/y][y/z]$ are no longer equivalent. Nevertheless, $t_2$ and $t_3$ share the same normal form, and thus are still operationally equivalent, but in a weaker sense.

From here on we use the letter $\mathtt{p}$ to denote a **parameter** which represents any of the propagation systems $\{\mathtt{b}, \mathtt{u}\}$. For every $\mathtt{p} \in \{\mathtt{b}, \mathtt{u}\}$ we consider its associated **structural reduction system** $\lambda\mathtt{j}_\mathtt{p}/\mathtt{o}$, written $\lambda\mathtt{j}_\mathtt{b}/\mathtt{o}$ and $\lambda\mathtt{j}_\mathtt{u}/\mathtt{o}$ respectively, defined by the reduction relation $\mathtt{dB} \cup \mathtt{j} \cup \mathtt{p}$ modulo the equivalence relation $\mathtt{o}$, a relation which is denoted by $(\mathtt{dB} \cup \mathtt{j} \cup \mathtt{p})/\mathtt{o}$. Both structural systems have good properties. By interpreting $t$ into $\mathtt{j}(t)$ and using Theorem 1 we get:

**Theorem 4 (Confluence Modulo).** *For all $t_1, t_2 \in \mathcal{T}$, if $t_1 \equiv_\mathtt{o} t_2$ and $t_i \rightarrow^*_{\lambda\mathtt{j}_\mathtt{p}/\mathtt{o}} u_i$ $(i = 1, 2)$, then $\exists v_i$ $(i = 1, 2)$ s.t. $u_i \rightarrow^*_{\lambda\mathtt{j}} v_i$ $(i = 1, 2)$ and $v_1 \equiv_\mathtt{o} v_2$.*

To prove PSN for $\lambda\mathtt{j}_\mathtt{b}/\mathtt{o}$ and $\lambda\mathtt{j}_\mathtt{u}/\mathtt{o}$ it is sufficient, according to Theorem 2, to show the **IE** property. However, a simple inductive argument like the one used for $\lambda\mathtt{j}$-reduction relation does no longer work. Therefore we shall show the **IE** property by adapting the technique in [13]. This has proven a challenging venture, so that this section presents the perhaps most important technical achievement in this paper. We split the proof into the following steps:

1. Define a labelling to mark some $\lambda\mathtt{j}_\mathtt{p}/\mathtt{o}$-strongly normalising terms used within jumps. Thus for example $t[\![x/u]\!]$ means that $u \in \mathcal{T}$ and $u \in \mathcal{SN}_{\lambda\mathtt{j}_\mathtt{p}/\mathtt{o}}$.

2. Enrich the original $\lambda\mathtt{j_p/o}$-reduction system with a relation used only to propagate terminating labelled jumps. Let $\mathcal{J}_\mathtt{p}/\mathtt{O}$ be the resulting calculus.
3. Show that $u \in \mathcal{SN}_{\lambda\mathtt{j_p/o}}$ and $t\{x/u\}\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j_p/o}}$ imply $t\llbracket x/u\rrbracket\overline{v}^1_n \in \mathcal{SN}_{\mathcal{J}_\mathtt{p}/\mathtt{O}}$.
4. Show that $t\llbracket x/u\rrbracket\overline{v}^1_n \in \mathcal{SN}_{\mathcal{J}_\mathtt{p}/\mathtt{O}}$ implies $t[x/u]\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j_p/o}}$.

In Sections 6.1 and 6.2 points 1 and 2 are developed, while Section 6.3 deals with points 3 and 4.

## 6.1 The Labelled Systems

Each labelled system is defined by a set of labelled terms together with a set of reduction rules and axioms.

**Definition 1 (Labelled Terms).** *Let* $\mathtt{p} \in \{\mathtt{b},\mathtt{u}\}$. *The set* $\mathbb{T}_\mathtt{p}$ *of* **labelled p-terms** *is generated using the following grammar:*

$$t ::= x \mid tt \mid \lambda x.t \mid t[x/t] \mid t\llbracket x/v\rrbracket \;\; (v \in \mathcal{T} \cap \mathcal{SN}_{\lambda\mathtt{j_p/o}})$$

Now consider the following reduction subsystems:

$\boxed{\text{The } \textbf{Labelled Equations } \underline{\mathtt{CS}}}$

if $y \notin \mathtt{fv}(u)$ & $x \notin \mathtt{fv}(v)$ :

$$\begin{array}{lll} t\llbracket x/u\rrbracket[y/v] & \equiv_{\underline{\mathtt{CS}}_1} & t[y/v]\llbracket x/u\rrbracket \\ t\llbracket x/u\rrbracket\llbracket y/v\rrbracket & \equiv_{\underline{\mathtt{CS}}_2} & t\llbracket y/v\rrbracket\llbracket x/u\rrbracket \end{array}$$

$\boxed{\text{The } \textbf{Labelled Unboxing} \text{ system } \underline{\mathtt{u}}}$

if $x \notin \mathtt{fv}(t)$ & $x \in \mathtt{fv}(v)$ :

$$\begin{array}{lll} tv\llbracket x/u\rrbracket & \rightarrow_{\underline{\mathtt{au}}} & (tv)\llbracket x/u\rrbracket \\ t[y/v\llbracket x/u\rrbracket] & \rightarrow_{\underline{\mathtt{su}}_1} & t[y/v]\llbracket x/u\rrbracket \\ t\llbracket y/v[x/u]\rrbracket & \rightarrow_{\underline{\mathtt{su}}_2} & t\llbracket y/v\rrbracket\llbracket x/u\rrbracket \end{array}$$

$\boxed{\text{The } \textbf{Generalised } \mathtt{dB} \text{ rule}}$

$$(\lambda x.t)\mathbb{L}u \rightarrow_{\mathtt{gdB}} t[x/u]\mathbb{L}$$

$\boxed{\text{The } \textbf{Labelled Equations } \underline{\sigma}}$

$$\begin{array}{llll} (\lambda y.t)\llbracket x/u\rrbracket & \equiv_{\underline{\sigma}_1} & \lambda y.t\llbracket x/u\rrbracket & \text{if } y \notin \mathtt{fv}(u) \\ (tv)\llbracket x/u\rrbracket & \equiv_{\underline{\sigma}_2} & t\llbracket x/u\rrbracket v & \text{if } x \notin \mathtt{fv}(v) \end{array}$$

$\boxed{\text{The } \textbf{Labelled Boxing} \text{ system } \underline{\mathtt{b}}}$

if $x \in \mathtt{fv}(v)$ :

$$\begin{array}{lll} (tv)\llbracket x/u\rrbracket & \rightarrow_{\underline{\mathtt{ab}}} & tv\llbracket x/u\rrbracket \\ t[y/v]\llbracket x/u\rrbracket & \rightarrow_{\underline{\mathtt{sb}}} & t[y/v\llbracket x/u\rrbracket] \end{array}$$

$\boxed{\text{The } \textbf{Labelled Jumping} \text{ system } \underline{\mathtt{j}}}$

$$\begin{array}{llll} t\llbracket x/u\rrbracket & \rightarrow_{\underline{\mathtt{w}}} & t & \text{if } |t|_x = 0 \\ t\llbracket x/u\rrbracket & \rightarrow_{\underline{\mathtt{d}}} & t\{x/u\} & \text{if } |t|_x = 1 \\ t\llbracket x/u\rrbracket & \rightarrow_{\underline{\mathtt{c}}} & t_{[y]_x}\llbracket x/u\rrbracket\llbracket y/u\rrbracket & \text{if } |t|_x \geq 2 \end{array}$$

where $\mathbb{L}$ is a list of jumps, some of which, potentially all, may be labelled. Note that $\mathtt{dB}$-reduction on the set $\mathcal{T}$ just is a particular case of $\mathtt{gdB}$-reduction on $\mathbb{T}_\mathtt{p}$. The **equivalence relation** $\underline{\alpha}$ (resp. $\underline{\mathtt{o}}$) is generated by axiom $\alpha$ (resp. $\{\underline{\alpha}, \underline{\mathtt{CS}}, \underline{\sigma}\}$) on labelled terms. The **equivalence relation** $\mathtt{O}$ is generated by $\mathtt{o} \cup \underline{\mathtt{o}}$. The **reduction relation** $\mathcal{J}_\mathtt{p}$ (resp. $\mathcal{J}_\mathtt{p}/\mathtt{O}$) is generated by $(\mathtt{gdB} \cup \mathtt{j} \cup \underline{\mathtt{j}} \cup \mathtt{p} \cup \underline{\mathtt{p}})$ (resp. $\mathtt{gdB} \cup \mathtt{j} \cup \underline{\mathtt{j}} \cup \mathtt{p} \cup \underline{\mathtt{p}}$ modulo $\mathtt{O}$). The relation $\mathcal{J}_\mathtt{p}$ can be understood as the union of two disjoint reduction relations, respectively called forgettable and persistent. Forgettable reductions do not create persistent redexes, and they are strongly normalising (Lemma 6). These two facts imply that termination of $\mathcal{J}_\mathtt{p}$ does not depend on its forgettable subsystem.

The **forgettable** reduction relation $\rightarrow_{\mathtt{Fp}}$:

– If $t \rightarrow_{\underline{\mathtt{j}},\underline{\mathtt{p}}} t'$, then $t \rightarrow_{\mathtt{Fp}} t'$.
– If $v \rightarrow_{\lambda\mathtt{j_p/o}} v'$, then $u\llbracket x/v\rrbracket \rightarrow_{\mathtt{Fp}} u\llbracket x/v'\rrbracket$.
– If $t \rightarrow_{\mathtt{Fp}} t'$, then $tu \rightarrow_{\mathtt{Fp}} t'u$, $ut \rightarrow_{\mathtt{Fp}} ut'$, $\lambda x.t \rightarrow_{\mathtt{Fp}} \lambda x.t'$, $t[x/u] \rightarrow_{\mathtt{Fp}} t'[x/u]$, $u[x/t] \rightarrow_{\mathtt{Fp}} u[x/t']$ and $t\llbracket x/u\rrbracket \rightarrow_{\mathtt{Fp}} t'\llbracket x/u\rrbracket$.

The **persistent** reduction relation $\rightarrow_{\mathtt{Pp}}$:

- If $t \mapsto_{\mathtt{gdB,j,p}} t'$ (where $\mapsto$ denotes root reduction), then $t \rightarrow_{\mathtt{Pp}} t'$.
- If $t \rightarrow_{\mathtt{Pp}} t'$, then $tu \rightarrow_{\mathtt{Pp}} t'u$, $ut \rightarrow_{\mathtt{Pp}} ut'$, $\lambda x.t \rightarrow_{\mathtt{Pp}} \lambda x.t'$, $t[x/u] \rightarrow_{\mathtt{Pp}} t'[x/u]$, $u[x/t] \rightarrow_{\mathtt{Pp}} u[x/t']$ and $t[\![x/u]\!] \rightarrow_{\mathtt{Pp}} t'[\![x/u]\!]$.

## 6.2 Well-Formed Labelled Terms

In order to prove that the $\lambda\mathtt{j}_{\mathtt{p}}/\mathtt{o}$-calculus enjoys PSN, according to Theorem 2 it is sufficient to show the **IE**-property. The reasoning for that is splitted in two steps: we first show that $u \in \mathcal{SN}_{\lambda\mathtt{j}_{\mathtt{p}}/\mathtt{o}}$ and $t\{x/u\}\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j}_{\mathtt{p}}/\mathtt{o}}$ imply $t[\![x/u]\!]\overline{v}^1_n \in \mathcal{SN}_{\mathcal{J}_{\mathtt{p}}/\mathtt{O}}$ (Corollary 6), thereafter we prove that $t[\![x/u]\!]\overline{v}^1_n \in \mathcal{SN}_{\mathcal{J}_{\mathtt{p}}/\mathtt{O}}$ implies $t[x/u]\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j}_{\mathtt{p}}/\mathtt{o}}$ (Corollary 7).

The first implication is much more difficult to prove, particularly because termination of the forgettable subsystem $\mathtt{F}_{\mathtt{p}}$, proved using a strictly decreasing measure on labelled terms, is required. This measure is based on the assumption that all terms inside labelled jumps are $\lambda\mathtt{j}_{\mathtt{p}}/\mathtt{o}$-strongly normalising w.r.t. the environment in which they are evaluated. Moreover, this property of labelled jumps needs to be preserved by reduction and equivalence. We concretely formalise this using the following key notions.

The set of **labelled free variables** of $t \in \mathbb{T}_{\mathtt{p}}$ is given by:

$$\begin{array}{llll}
\mathbb{L}\mathtt{fv}(x) & := [\,] & \mathbb{L}\mathtt{fv}(u[x/v]) & := (\mathbb{L}\mathtt{fv}(u) \setminus \{x\}) \cup \mathbb{L}\mathtt{fv}(v) \\
\mathbb{L}\mathtt{fv}(uv) & := \mathbb{L}\mathtt{fv}(u) \cup \mathbb{L}\mathtt{fv}(v) & \mathbb{L}\mathtt{fv}(u[\![x/v]\!]) & := (\mathbb{L}\mathtt{fv}(u) \setminus \{x\}) \cup \mathtt{fv}(v) \\
\mathbb{L}\mathtt{fv}(\lambda x.u) & := \mathbb{L}\mathtt{fv}(u) \setminus \{x\} & &
\end{array}$$

Note that $u \in \mathcal{T}$ implies $\mathbb{L}\mathtt{fv}(u) = [\,]$ and also $\mathbb{L}\mathtt{fv}(t) \subseteq \mathtt{fv}(t)$. Moreover, if $t \equiv_{\mathtt{O}} u$ then $\mathbb{L}\mathtt{fv}(t) = \mathbb{L}\mathtt{fv}(u)$, and if $t \rightarrow_{\mathcal{J}_{\mathtt{p}}} u$ then $\mathbb{L}\mathtt{fv}(t) \supseteq \mathbb{L}\mathtt{fv}(u)$.

A labelled term $t \in \mathbb{T}_{\mathtt{p}}$ is $\mathtt{SN}$**-labelled** for a substitution $\gamma$ iff $\mathtt{SN}\mathbb{L}_{\mathtt{p}}(t,\gamma)$ holds:

$$\begin{array}{llll}
\mathtt{SN}\mathbb{L}_{\mathtt{p}}(x,\gamma) & := \mathtt{true} & \mathtt{SN}\mathbb{L}_{\mathtt{p}}(tu,\gamma) & := \mathtt{SN}\mathbb{L}_{\mathtt{p}}(t,\gamma)\ \&\ \mathtt{SN}\mathbb{L}_{\mathtt{p}}(u,\gamma) \\
\mathtt{SN}\mathbb{L}_{\mathtt{p}}(\lambda x.t,\gamma) & := \mathtt{SN}\mathbb{L}_{\mathtt{p}}(t,\gamma) & \mathtt{SN}\mathbb{L}_{\mathtt{p}}(t[x/u],\gamma) & := \mathtt{SN}\mathbb{L}_{\mathtt{p}}(t,\gamma)\ \&\ \mathtt{SN}\mathbb{L}_{\mathtt{p}}(u,\gamma) \\
 & & \mathtt{SN}\mathbb{L}_{\mathtt{p}}(t[\![x/u]\!],\gamma) & := \mathtt{SN}\mathbb{L}_{\mathtt{p}}(t,\{x/u\}\gamma)\ \&\ u\gamma \in \mathcal{SN}_{\lambda\mathtt{j}_{\mathtt{p}}/\mathtt{o}}
\end{array}$$

A $\mathtt{p}$-labelled term $t$ is $\mathtt{p}$**-well-formed**, written $t \in \mathbb{WF}_{\mathtt{p}}$, iff **(1)** $\mathtt{SN}\mathbb{L}_{\mathtt{p}}(t,[\,])$ **(2)** every subterm $u[y/v]$ or $\lambda y.u$ in $t$ verifies $y \notin \mathbb{L}\mathtt{fv}(u)$ **(3)** $\mathtt{p} = \mathtt{b}$ implies subterms $u[\![y/v]\!] \in t$ verify $y \notin \mathbb{L}\mathtt{fv}(u)$. Thus for example $t_0 = (xx)[\![x/y]\!][\![y/z]\!]$ is not $\mathtt{b}$-well-formed since $y$ is not a labelled free variable of $t_0$, whereas $t_0$ is $\mathtt{u}$-well-formed since $z \in \mathcal{SN}_{\lambda\mathtt{j}_{\mathtt{u}}/\mathtt{o}}$. Also, $t_1 = y[y/x][x/\lambda z.zz]$ is $\mathtt{b}$ and $\mathtt{u}$ well-formed but $t_2 = y[\![y/xx]\!][\![x/\lambda z.zz]\!]$ is not. More precisely, $x$ is a labelled free variable of $y[\![y/xx]\!]$ so that $t_2$ is not $\mathtt{b}$-well-formed, and $\mathtt{SN}\mathbb{L}_{\mathtt{u}}(t_2,\emptyset)$ does not hold (since $(\lambda z.zz)(\lambda z.zz) \notin \mathcal{SN}_{\lambda\mathtt{j}_{\mathtt{u}}/\mathtt{o}}$) hence $t_2$ is not $\mathtt{u}$-well-formed.

$\mathtt{SN}$-labelled (and well-formed) terms are stable by equivalence and reduction.

**Lemma 5.** *Let* $\mathtt{SN}\mathbb{L}_{\mathtt{p}}(t_0,\gamma)$. *If* $t_0 \equiv_{\mathtt{O}} t_1$ *or* $t_0 \rightarrow_{\mathcal{J}_{\mathtt{p}}} t_1$, *then* $\mathtt{SN}\mathbb{L}_{\mathtt{p}}(t_1,\gamma)$.

**Corollary 5.** *Let* $t \in \mathbb{WF}_{\mathtt{p}}$. *If* $t \equiv_{\mathtt{O}} t'$ *or* $t \rightarrow_{\mathcal{J}_{\mathtt{p}}} t'$, *then* $t' \in \mathbb{WF}_{\mathtt{p}}$.

The given corollary is essential in developing the termination proofs for the forgettable relations $\mathtt{F}_{\mathtt{b}}/\mathtt{O}$ and $\mathtt{F}_{\mathtt{u}}/\mathtt{O}$. More precisely, for each forgettable reduction $\mathtt{F}_{\mathtt{p}}/\mathtt{O}$, with $\mathtt{p} \in \{\mathtt{b},\mathtt{u}\}$, we define a measure on $\mathtt{p}$-well-formed labelled terms which strictly decreases by $\mathtt{F}_{\mathtt{p}}/\mathtt{O}$-reduction. For lack of space we relegate the proof to [2].

**Lemma 6.** *The relation* $\rightarrow_{\mathtt{F_b/0}}$ *(resp.* $\rightarrow_{\mathtt{F_u/0}}$*) is terminating on* $\mathbb{WF}_{\mathtt{b}}$ *(resp.* $\mathbb{WF}_{\mathtt{u}}$*).*

## 6.3 From Implicit to Explicit through Labelled

To show our first point, namely, that $u \in \mathcal{SN}_{\lambda\mathtt{j_p/o}}$ and $t\{x/u\}\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j_p/o}}$ imply $t[\![x/u]\!]\overline{v}^1_n \in \mathcal{SN}_{\mathcal{J}_\mathtt{p}/\mathtt{0}}$, we now consider the following projection function $\mathbb{P}(\_)$ from labelled terms to terms, which also projects $\mathcal{J}_\mathtt{p}/\mathtt{0}$ into the reduction $\lambda\mathtt{j_p/o}$:

$$\begin{array}{lll} \mathbb{P}(x) := x & \mathbb{P}(\lambda x.t) := \lambda x.\mathbb{P}(t) & \mathbb{P}(t[x/u]) := \mathbb{P}(t)[x/\mathbb{P}(u)] \\ & \mathbb{P}(tu) \quad := \mathbb{P}(t)\mathbb{P}(u) & \mathbb{P}(t[\![x/u]\!]) := \mathbb{P}(t)\{x/u\} \end{array}$$

Note that $u \in \mathcal{T}$ implies $\mathbb{P}(u) = u$.

**Lemma 7.** *Let* $t_0 \in \mathbb{T}_\mathtt{p}$. *Then,*

1. $t_0 \equiv_\mathtt{0} t_1$ *implies* $\mathbb{P}(t_0) \equiv_\mathtt{o} \mathbb{P}(t_1)$.
2. $t_0 \rightarrow_{\mathtt{Fp}} t_1$ *implies* $\mathbb{P}(t_0) \rightarrow^*_{\lambda\mathtt{j_p/o}} \mathbb{P}(t_1)$.
3. $t_0 \rightarrow_{\mathtt{Pp}} t_1$ *implies* $\mathbb{P}(t_0) \rightarrow^+_{\lambda\mathtt{j_p/o}} \mathbb{P}(t_1)$.

*Proof.* By induction on labelled terms. The case $t_0 \rightarrow_{\underline{\mathtt{su}}_2} t_1$ uses Lemma 1.

**Lemma 8.** *Let* $t \in \mathbb{WF}_\mathtt{p}$. *If* $\mathbb{P}(t) \in \mathcal{SN}_{\lambda\mathtt{j_p/o}}$, *then* $t \in \mathcal{SN}_{\mathcal{J}_\mathtt{p}/\mathtt{0}}$.

*Proof.* Since $\rightarrow_{\mathcal{J}_\mathtt{p}} = \rightarrow_{\mathtt{F_p}} \cup \rightarrow_{\mathtt{P_p}}$ we show that $t \in \mathcal{SN}_{\mathtt{F_p \cup P_p/0}}$ by using Lemma 7 and termination of the forgettable relations (Lemma 6).

Now let $\mathtt{p} \in \{\mathtt{b}, \mathtt{u}\}$ and consider $t, u, \overline{v}^1_n \in \mathcal{T}$ s.t. $u \in \mathcal{SN}_{\lambda\mathtt{j_p/o}}$. We immediately get $t[\![x/u]\!]\overline{v}^1_n \in \mathbb{WF}_\mathtt{p}$. Using $\mathbb{P}(t[\![x/u]\!]\overline{v}^1_n) = t\{x/u\}\overline{v}^1_n$ we thus conclude:

**Corollary 6.** *Let* $t, u, \overline{v}^1_n \in \mathcal{T}$. *If* $u \in \mathcal{SN}_{\lambda\mathtt{j_p/o}}$ *&* $t\{x/u\}\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j_p/o}}$, *then* $t[\![x/u]\!]\overline{v}^1_n \in \mathcal{SN}_{\mathcal{J}_\mathtt{p}/\mathtt{0}}$.

The last point of our proof is to show that $t[\![x/u]\!]\overline{v}^1_n \in \mathcal{SN}_{\mathcal{J}_\mathtt{p}/\mathtt{0}}$ implies $t[x/u]\overline{v}^1_n \in \mathcal{SN}_{\lambda\mathtt{j_p/o}}$ by relating labelled terms and reductions to unlabelled terms and reductions. To do that, let us introduce an **unlabelling function on labelled terms**:

$$\begin{array}{lll} \mathtt{U}(x) := x & \mathtt{U}(\lambda x.t) := \lambda x.\mathtt{U}(t) & \mathtt{U}(t[x/u]) := \mathtt{U}(t)[x/\mathtt{U}(u)] \\ & \mathtt{U}(tu) \quad := \mathtt{U}(t)\mathtt{U}(u) & \mathtt{U}(t[\![x/u]\!]) := \mathtt{U}(t)[x/u] \end{array}$$

There is a one-to-one correspondence between labelled and unlabelled reduction. Moreover, well-formed labelled terms are essential here to get the following:

**Lemma 9.** *Let* $t \in \mathbb{WF}_\mathtt{p}$. *If* $t \in \mathcal{SN}_{\mathcal{J}_\mathtt{p}/\mathtt{0}}$, *then* $\mathtt{U}(t) \in \mathcal{SN}_{\lambda\mathtt{j_p/o}}$.

*Proof.* We prove $\mathtt{U}(t) \in \mathcal{SN}_{\lambda\mathtt{j_p/o}}$ by induction on $\eta_{\mathcal{J}_\mathtt{p}/\mathtt{0}}(t)$. This is done by considering all the $\lambda\mathtt{j_p/o}$-reducts of $\mathtt{U}(t)$ and using the following: if $t \in \mathbb{WF}_\mathtt{p}$ and $\mathtt{U}(t) \rightarrow_{\lambda\mathtt{j_p/o}} u$, then $\exists\ v \in \mathbb{WF}_\mathtt{p}$ s.t. $t \rightarrow_{\mathcal{J}_\mathtt{p}/\mathtt{0}} v$ and $\mathtt{U}(v) = u$.

Now let $\mathtt{p} \in \{\mathtt{b}, \mathtt{u}\}$ and consider $t, u, \overline{v}^1_n \in \mathcal{T}$ s.t. $u \in \mathcal{SN}_{\lambda \mathtt{j}_\mathtt{p}/\mathtt{o}}$. We immediately get $t[\![x/u]\!]\overline{v}^1_n \in \mathbb{WF}_\mathtt{p}$. Using $\mathtt{U}(t[\![x/u]\!]\overline{v}^1_n) = t[x/u]\overline{v}^1_n$ we thus conclude:

**Corollary 7.** *Let $t, u, \overline{v}^1_n \in \mathcal{T}$. If $t[\![x/u]\!]\overline{v}^1_n \in \mathbb{WF}_\mathtt{p} \cap \mathcal{SN}_{\mathcal{J}_\mathtt{p}/\mathtt{O}}$, then $t[x/u]\overline{v}^1_n \in \mathcal{SN}_{\lambda \mathtt{j}_\mathtt{p}/\mathtt{o}}$.*

From Corollaries 6 and 7 we get:

**Lemma 10 (IE for $\lambda \mathtt{j}_\mathtt{p}/\mathtt{o}$).** *For $\mathtt{p} \in \{\mathtt{b}, \mathtt{u}\}$, $\lambda \mathtt{j}_\mathtt{p}/\mathtt{o}$ enjoys the* **IE** *property.*

Theorem 2 thus allows us to conclude with the main result of this section:

**Corollary 8 (PSN for $\lambda \mathtt{j}_\mathtt{p}/\mathtt{o}$).** *For $\mathtt{p} \in \{\mathtt{b}, \mathtt{u}\}$, $\lambda \mathtt{j}_\mathtt{p}/\mathtt{o}$ enjoys PSN.*

## 7 Conclusions

We have introduced the structural $\lambda\mathtt{j}$-calculus, a concise but expressive $\lambda$-calculus with jumps. No prior knowledge of Linear Logic is necessary to understand $\lambda\mathtt{j}$, despite their strong connection. We have established many different sanity properties for $\lambda\mathtt{j}$ such as confluence and PSN. We have used $\lambda\mathtt{j}$ as an operational framework to elaborate new characterisations of the well-known notions of full developments and L-developments, and to obtain the new, more powerful notion of XL-development. Finally, we have modularly added commutation of independent jumps, $\sigma$-equivalence and two kinds of propagations of jumps, while showing that PSN still holds.

As noted in Section 6, PSN for the $\lambda\mathtt{j}$-calculus plus the constrained equivalences $\{\mathtt{CS}, \sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ is - at present - a challenging conjecture. Indeed, the merging of the two similar, yet different uses of $\{\sigma_3, \sigma_4\}$ that we study in this paper presents several non-trivial difficulties. A further in-depth re-elaboration of the labelling technique would be necessary, perhaps even the use of a completely different technique dealing with reduction modulo a set of equations.

An interesting research direction is the study of linear head reduction [3] for $\lambda$-calculus - which is closely connected to game semantics and abstract machines - whose formulation is not a strategy in the usual sense. Indeed, jumps and distance permit to reformulate linear head reduction as a strategy of $\lambda\mathtt{j}$.

It would also be interesting to exploit distance and multiplicities in other frameworks for example when dealing with pattern matching, continuations or differential features.

## References

1. Accattoli, B., Guerrini, S.: Jumping boxes. representing lambda-calculus boxes by jumps. In: Grädel, E., Kahle, R. (eds.) CSL 2009. LNCS, vol. 5771, pp. 55–70. Springer, Heidelberg (2009)
2. Accattoli, B., Kesner, D.: The structural calculus $\lambda$j. Technical report, PPS, CNRS and University Paris-Diderot (2010)
3. Danos, V., Regnier, L.: Reversible, irreversible and optimal lambda-machines. TCS 227(1), 79–97 (1999)
4. David, R., Guillaume, B.: A $\lambda$-calculus with explicit weakening and explicit substitution. MSCS 11, 169–206 (2001)

5. de Bruijn, N.G.: Generalizing Automath by Means of a Lambda-Typed Lambda Calculus. In: Mathematical Logic and Theoretical Computer Science. Lecture Notes in Pure and Applied Mathematics, vol. 106, pp. 71–92. Marcel Dekker, New York (1987)
6. de Moor, O., Sittampalam, G.: Higher-order matching for program transformation. TCS 269(1-2), 135–162 (2001)
7. Di Cosmo, R., Kesner, D., Polonovski, E.: Proof nets and explicit substitutions. MSCS 13(3), 409–450 (2003)
8. Faure, G.: Matching modulo superdevelopments application to second-order matching. In: Hermann, M., Voronkov, A. (eds.) LPAR 2006. LNCS (LNAI), vol. 4246, pp. 60–74. Springer, Heidelberg (2006)
9. Girard, J.-Y.: Linear logic. TCS 50 (1987)
10. Girard, J.-Y.: Geometry of interaction i: an interpretation of system f. In: Proc. of the Logic Colloquim, vol. 88, pp. 221–260 (1989)
11. Herbelin, H., Zimmermann, S.: An operational account of call-by-value minimal and classical lambda-calculus in ”natural deduction” form. In: Curien, P.-L. (ed.) TLCA 2009. LNCS, vol. 5608, pp. 142–156. Springer, Heidelberg (2009)
12. Hindley, J.R.: Reductions of residuals are finite. Transactions of the American Mathematical Society 240, 345–361 (1978)
13. Kesner, D.: The theory of calculi with explicit substitutions revisited. In: Duparc, J., Henzinger, T.A. (eds.) CSL 2007. LNCS, vol. 4646, pp. 238–252. Springer, Heidelberg (2007)
14. Kesner, D.: A theory of explicit substitutions with safe and full composition. LMCS 5(3:1), 1–29 (2009)
15. Kesner, D., Lengrand, S.: Resource operators for lambda-calculus. I & C 205(4), 419–473 (2007)
16. Kesner, D., Renaud, F.: The prismoid of resources. In: Královič, R., Niwiński, D. (eds.) MFCS 2009. LNCS, vol. 5734, pp. 464–476. Springer, Heidelberg (2009)
17. Klop, J.-W., van Oostrom, V., van Raamsdonk, F.: Combinatory reduction systems: introduction and survey. TCS 121(1/2), 279–308 (1993)
18. Lévy, J.-J.: Réductions correctes et optimales dans le lambda-calcul. PhD thesis, Univ. Paris VII, France (1978)
19. Lévy, J.-J., Maranget, L.: Explicit substitutions and programming languages. In: Pandu Rangan, C., Raman, V., Sarukkai, S. (eds.) FSTTCS 1999. LNCS, vol. 1738, pp. 181–200. Springer, Heidelberg (1999)
20. Maraist, J., Odersky, M., Turner, D.N., Wadler, P.: Call-by-name, call-by-value, call-by-need and the linear lambda calculus. TCS 228(1-2), 175–210 (1999)
21. Milner, R.: Local bigraphs and confluence: two conjectures. In: Proc. of 13th EXPRESS. ENTCS, vol. 175. Elsevier, Amsterdam (2006)
22. Nederpelt, R.P.: The fine-structure of lambda calculus. Technical Report CSN 92/07, Eindhoven Univ. of Technology (1992)
23. Ó Conchúir, S.: Proving PSN by simulating non-local substitutions with local substitution. In: Proceedings of the 3rd HOR, August 2006, pp. 37–42 (2006)
24. Ohta, Y., Hasegawa, M.: A terminating and confluent linear lambda calculus. In: Pfenning, F. (ed.) RTA 2006. LNCS, vol. 4098, pp. 166–180. Springer, Heidelberg (2006)
25. Regnier, L.: Lambda-calcul et réseaux. Thèse de doctorat, Univ. Paris VII (1992)
26. Regnier, L.: Une équivalence sur les lambda-termes. TCS 2(126), 281–292 (1994)
27. Schwichtenberg, H.: Termination of permutative conversions in intuitionistic Gentzen calculi. TCS 212(1-2), 247–260 (1999)
28. Severi, P., Poll, E.: Pure type systems with definitions. In: Matiyasevich, Y.V., Nerode, A. (eds.) LFCS 1994. LNCS, vol. 813, pp. 316–328. Springer, Heidelberg (1994)
29. van Raamsdonk, F.: Confluence and Normalization for Higher-Order Rewriting. PhD thesis, Amsterdam Univ., Netherlands (1996)