

PRESERVATION OF STRONG NORMALISATION MODULO PERMUTATIONS FOR THE STRUCTURAL λ -CALCULUS

BENIAMINO ACCATTOLI AND DELIA KESNER

LIPN (CNRS and Université Paris-Nord) and INRIA and LIX (École Polytechnique)
e-mail address: beniamino.accattoli@gmail.com

PPS (CNRS and Université Paris-Diderot)
e-mail address: delia.kesner@pps.jussieu.fr

ABSTRACT. Inspired from a recent graphical formalism for λ -calculus based on linear logic technology, we introduce an untyped structural λ -calculus, called λ_j , which combines actions at a distance with exponential rules decomposing the substitution by means of weakening, contraction and dereliction. First, we prove some fundamental properties of λ_j such as confluence and preservation of β -strong normalisation. Secondly, we add a strong bisimulation to λ_j by means of an equational theory which captures in particular Regnier's σ -equivalence. We then complete this bisimulation with two more equations for (de)composition of substitutions and we prove that the resulting calculus still preserves β -strong normalization. Finally, we discuss some consequences of our results.

1. INTRODUCTION

Linear Logic [10] has been very influential in computer science, especially because it gives a mechanism to explicitly control the use of resources by limiting the use of the *structural rules* of weakening and contraction. Erasure (weakening) and duplication (contraction) are restricted to formulas marked with an *exponential* modality $?$, and can only interact with non-linear proofs marked with a bang modality $!$. Intuitionistic and Classical Logic can thus be encoded by a fragment containing such modalities, as for example the Multiplicative Exponential Linear Logic (MELL).

MELL proofs can be represented by sequent trees, but MELL Proof-Nets [10] give a better geometrical representation of proofs that eliminates irrelevant syntactical details. They have been extensively used to develop different encodings of intuitionistic logic/ λ -calculus, giving rise to the geometry of interaction [11].

Normalisation of proofs (*i.e.* *cut elimination*) in MELL Proof-Nets is performed in particular by *exponential* and *commutative* rules. Non-linear proofs are distinguished by surrounding *boxes*; the exponential rules handle all the possible operations on them: erasure, duplication and linear use, corresponding respectively to a cut elimination step involving a box and either a *weakening*, a *contraction* or a *dereliction*. The commutative rule instead allows to *compose* non-linear resources.

Different cut elimination systems [8, 21, 17], defined as *explicit substitution* (ES) calculi, were explained in terms of, or were inspired from, the fine notion of reduction of MELL Proof-Nets. They all 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 some commutative rules. They also share in common an operational semantics defined in terms of a *propagation system* in which a substitution traverses a term until the variables are reached.

The structural λ -calculus. A graphical representation for λ -terms, λ j-dags, has been recently proposed [2]. It denies boxes by representing them with additional edges called *jumps*, and does not need any commutative reduction rule to compose non-linear proofs. This paper studies the term formalism, called λ j-calculus, resulting from reading back λ j-dags (and their correspondent reductions) by means of their sequentialisation theorem [2]. The deep connection between λ j-dags and Danos and Regnier's Pure (*untyped*) Proof-Nets [5] has been already studied in [1].

Beyond this graphical and logical interpretation, the peculiarity of λ j-calculus is that it uses two features which were never combined before: *action at a distance* and *multiplicities*.

Action at a distance means that rewriting rules are specified by means of some constructors which are arbitrary far away from each other. This approach could be understood as inconvenient but this is only apparent since rewriting rules can be locally implemented by means of λ j-dags. The distance rules of λ j do not propagate substitutions through the term except for the linear ones which are evaluated exactly as meta-level substitutions, regardless the distance between the involved constructors (variable and jump).

Multiplicities are intended to count the number of occurrences of a given variable affected by a jump, *i.e.* the rewriting rule to be applied for reducing a term of the form $t[x/u]$ depends on $|t|_x$, the number of free occurrences of the variable x in the term t . Indeed, we distinguish three cases, $|t|_x = 0$, $|t|_x = 1$ and $|t|_x > 1$, which corresponds, respectively, to weakening-box, dereliction-box and contraction-box cut-elimination rules in Proof Nets. It is because of the weakening and contraction rules that we call our language the *structural λ -calculus*.

Content of the paper. We start by showing that λ j admits a simple and elegant theory *i.e.* it enjoys confluence, full composition (FC), and preservation of β -strong normalisation. The proof of PSN is particularly concise because of the distance approach.

The main result of the paper is that the theory of λj admits a modular extension with respect to propagations of jumps: an equational theory is added on top of λj and the obtained extension is shown to preserve all the good properties we mentioned before. Actually, we focus on PSN, since FC and confluence for the extended λj -calculus turn out to be straightforward.

In the literature there is a huge number of calculi with explicit substitutions, `let` constructs or environments, most of them use some rule to specify commutation (also called propagation or permutation). In order to encompass these formalisms we do not approach propagations as *rewriting rules*, but as equations (which can be used from left to right or vice-versa) defining an *equivalence relation* on terms.

This is only possible because propagations are not needed in λj to compute normal forms, which is a by-product of the *distance* notion. Moreover, any particular orientation of the equations (from left to right *or* from right to left) results in a terminating rewriting relation, so that if the reduction system modulo the equivalence relation enjoys PSN then the system containing *any* orientation of the equations defining this equivalence relation still enjoys PSN.

Equations are introduced in two steps. We first consider commutations between jumps and linear constructors of the calculus (*i.e.* abstractions, left sides of applications and independent jumps). This equivalence, written \equiv_{\circ} , turns out to be a strong bisimulation, *i.e.* a reduction length preserving relation; thus PSN for the reduction system λj modulo the equivalence relation \equiv_{\circ} - noted $\lambda j/\circ$ - immediately follows. We also show that \equiv_{\circ} can be seen as a projection of Regnier's σ -equivalence [33] on a syntax with jumps. Actually, \equiv_{\circ} can be understood as the quotient induced by the translation [1] of λj -terms to Pure Proof-Nets, which is why it is so well-behaved, and why we call it the *graphical equivalence*.

The second step is to extend \equiv_{\circ} with general commutations between jumps and non-linear constructs (right sides of applications and contents of jumps). The resulting *substitution equivalence* \equiv_{obox} , does not only subsume *composition* of jumps, but also *decomposition*. The equations of \equiv_{obox} correspond exactly to the commutative box-box case of Proof-Nets, but they are here considered as an *equivalence* - which is a novelty - and not as rewriting rule. The reduction relation of $\lambda j/\text{obox}$ is a rich rewriting system with subtle behaviour, particularly because \equiv_{obox} affects reduction lengths, and thus is not a strong bisimulation. Nonetheless, we show that $\lambda j/\text{obox}$ enjoys PSN.

This result is non-trivial, and constitutes the main contribution of the paper. The technique used to obtain PSN for $\lambda j/\text{obox}$ consists in

- (1) Projecting $\lambda j/\text{obox}$ reductions into a calculus that we call $\lambda \text{void}/\circ$,
- (2) Proving PSN for $\lambda \text{void}/\circ$,
- (3) Infer PSN for $\lambda j/\text{obox}$ from (1) and (2).

Actually, $\lambda \text{void}/\circ$ can be understood as a *memory* calculus specified by means of *void* jumps - *i.e.* jumps $t[x/u]$ where $x \notin \text{fv}(t)$ - which generalises Klop's Λ_I -calculus [24]. Despite the fact that it appears only as a technical tool we claim that it is a calculus interesting on its own and can be used for proving termination results beyond those of this paper.

The last part of the paper presents some interesting consequences of our main result concerning different variations on $\lambda j/\text{obox}$.

Road Map.

- Section 2 recalls some general notions about abstract rewriting.
- Section 3 presents the λj -calculus and shows that it enjoys basic properties such as full composition, simulation of one-step β -reduction and confluence.

- Section 4 studies preservation of β -strong normalisation (PSN). The PSN property is proved using a modular technique developed in [18], which results in a very short formal argument in our case.
- Section 5 first considers λj enriched with the linear equivalence \equiv_o (related to Regnier's σ -equivalence [33]) and then with the non-linear equivalence \equiv_{obox} , which contains also composition of jumps.
- Section 6 is devoted to the proof of PSN for λj modulo \equiv_{obox} , which is the main contribution of the paper.
- Section 7 discusses some consequences of the PSN result of Section 6.

This paper covers some basic results in [3] by extending them considerably. Indeed, the propagation systems considered in [3] are just particular cases of the general equational theory \equiv_{obox} studied in this paper. The proof technique used here to show PSN for λj modulo \equiv_{obox} puts in evidence another calculus $\lambda\text{void}/o$ that has interest in its own. Moreover, interesting consequences of the main result are included in Section 7.

Related Work. Action at a distance has already been used in [30, 7, 31], but none of the previous approaches takes advantage of distance plus control of resources by means of multiplicities. Other works use multiplicities [22] but not distance so that the resulting formalism contains a lot of rules, which is really less handy. We think that the combined approach is more primitive than ES, and the resulting theory is much simpler. Using distance *and* multiplicities also provides modularity: the substitution rules become independent from the set of constructors of the calculus, and thus any change in the language does not cause any changes in the associated rewriting rules. Our combined approach does not only capture the well-known notions of developments [14] and superdevelopments [25], but also allows to introduce XL-developments, a more powerful notion of development defined in [3].

In the literature there are many calculi which dealt with permutations of constructors in intuitionistic calculi, but all use *reduction* rules rather than *equations*, which is less powerful. Some that can be captured by our graphical equivalence appear in [16, 33, 23] and those captured by our substitution equivalence are [9, 13, 37]. Intuitionistic calculi inspired from Linear Logic Proof Nets appear for example in [20, 18, 22].

2. PRELIMINARY NOTIONS

As several reduction notions are used along the paper, we first introduce general definitions of rewriting.

A **reduction system** is a pair $(R, \rightarrow_{\mathcal{R}})$ consisting of a set R and a binary relation $\rightarrow_{\mathcal{R}}$ on R . When $(a, b) \in \rightarrow_{\mathcal{R}}$ we write $a \rightarrow_{\mathcal{R}} b$ and we say that a **\mathcal{R} -reduces** to b . The inverse of $\rightarrow_{\mathcal{R}}$ is written $\mathcal{R}\leftarrow$, *i.e.* $b \mathcal{R}\leftarrow a$ iff $a \rightarrow_{\mathcal{R}} b$. The reflexive and transitive (resp. transitive) closure of $\rightarrow_{\mathcal{R}}$ is written $\rightarrow_{\mathcal{R}}^*$ (resp. $\rightarrow_{\mathcal{R}}^+$). Composition of relations is denoted by juxtaposition. Given $k \geq 0$, we write $a \xrightarrow{k}_{\mathcal{R}} b$ iff a is \mathcal{R} -related to b in k steps, *i.e.* $a \xrightarrow{0}_{\mathcal{R}} b$ if $a = b$ and $a \xrightarrow{n+1}_{\mathcal{R}} b$ if $\exists c$ s.t. $a \rightarrow_{\mathcal{R}} c$ and $c \xrightarrow{n}_{\mathcal{R}} b$.

Given a **reduction system** $(R, \rightarrow_{\mathcal{R}})$, we use the following reduction notions and conventions:

- \mathcal{R} is **locally confluent** if $\mathcal{R}\leftarrow \rightarrow_{\mathcal{R}} \subseteq \rightarrow_{\mathcal{R}}^* \mathcal{R}\leftarrow$, *i.e.* if $a \rightarrow_{\mathcal{R}} b$ and $a \rightarrow_{\mathcal{R}} c$, then $\exists d$ s.t. $b \rightarrow_{\mathcal{R}}^* d$ and $c \rightarrow_{\mathcal{R}}^* d$.
- \mathcal{R} is **confluent** if $\mathcal{R}\leftarrow \rightarrow_{\mathcal{R}}^* \subseteq \rightarrow_{\mathcal{R}}^* \mathcal{R}\leftarrow$, *i.e.* if $a \rightarrow_{\mathcal{R}}^* b$ and $a \rightarrow_{\mathcal{R}}^* c$, then $\exists d$ s.t. $b \rightarrow_{\mathcal{R}}^* d$ and $c \rightarrow_{\mathcal{R}}^* d$.
- $s \in R$ is in **\mathcal{R} -normal form**, written $s \in \mathcal{R}\text{-nf}$, if there is no s' such that $s \rightarrow_{\mathcal{R}} s'$.

- $s \in R$ has an **\mathcal{R} -normal form** iff there exists $u \in \mathcal{R}\text{-nf}$ such that $s \rightarrow_{\mathcal{R}}^* u$. When s has a *unique* \mathcal{R} -normal form, this one is denoted by $\mathcal{R}(s)$.
- $s \in R$ is **\mathcal{R} -weakly normalizing**, written $s \in \mathcal{WN}_{\mathcal{R}}$, iff s has an \mathcal{R} -normal form.
- $s \in R$ is **\mathcal{R} -strongly normalizing** or **\mathcal{R} -terminating**, written $s \in \mathcal{SN}_{\mathcal{R}}$, if there is no infinite \mathcal{R} -reduction sequence starting at s , in which case the notation $\eta_{\mathcal{R}}(s)$ means the **maximal length of an \mathcal{R} -reduction sequence starting at s** . This notion is extended to lists of terms by $\eta_{\mathcal{R}}(s_1 \dots s_m) = \sum_{i=1}^m \eta_{\mathcal{R}}(s_i)$.

A **strong bisimulation between** two reduction systems (S, \rightarrow_S) and (Q, \rightarrow_Q) is a relation $\mathbf{E} \subseteq S \times Q$ s.t. for any pair $s \mathbf{E} t$:

- If $s \rightarrow_S s'$ then there is t' s.t. $t \rightarrow_Q t'$ and $s' \mathbf{E} t'$, and conversely:
- If $t \rightarrow_Q t'$ then there is s' s.t. $s \rightarrow_S s'$ and $s' \mathbf{E} t'$.

A **strong bisimulation for** (S, \rightarrow_S) is a strong bisimulation between (S, \rightarrow_S) and itself. In particular we shall make use of the following property whose proof is straightforward:

Lemma 2.1. *Let \mathbf{E} be a strong bisimulation between two reduction systems (S, \rightarrow_S) and (Q, \rightarrow_Q) .*

(1) *The relation \mathbf{E} preserves reduction lengths, i.e. for any $s \mathbf{E} t$*

- *If $s \xrightarrow{k}_S s'$ then $\exists t'$ s.t. $t \xrightarrow{k}_Q t'$ and $s' \mathbf{E} t'$.*
- *If $t \xrightarrow{k}_Q t'$ then $\exists s'$ s.t. $s \xrightarrow{k}_S s'$ and $s' \mathbf{E} t'$.*

(2) *The relation \mathbf{E} preserves strong normalization, i.e. for any $s \mathbf{E} t, s \in \mathcal{SN}_S$ if and only if $t \in \mathcal{SN}_Q$.*

Given a reduction relation \rightarrow_S and an equivalence relation \mathbf{E} both on S , the reduction relation $\rightarrow_{S/\mathbf{E}}$, called **reduction S modulo \mathbf{E}** , is defined by $t \rightarrow_{S/\mathbf{E}} u$ iff $t \mathbf{E} t' \rightarrow_S u' \mathbf{E} u$.

Lemma 2.2. *Let (S, \rightarrow_S) be a reduction system and \mathbf{E} an equivalence relation on S which is also a strong bisimulation for (S, \rightarrow_S) . Then,*

- (1) *The relation \mathbf{E} can be postponed w.r.t \rightarrow_S , i.e. $\rightarrow_{S/\mathbf{E}}^* = \rightarrow_S^* \mathbf{E}$.*
- (2) *If \rightarrow_S is confluent then $\rightarrow_{S/\mathbf{E}}$ is confluent.*
- (3) *If $t \in \mathcal{SN}_S$, then $t \in \mathcal{SN}_{S/\mathbf{E}}$.*

Proof. Point 1 is straightforward by induction on the length of $\rightarrow_{S/\mathbf{E}}^*$ using the definition of strong bisimulation. Points 2 and 3 follow from Point 1. \square

We finish this section by giving an abstract theorem used to show termination for different notions of reduction modulo appearing in the paper.

Theorem 2.3 (Termination for reduction modulo by interpretation). *Let \mathcal{A}_1 and \mathcal{A}_2 (resp. \mathbf{E}) be two reduction (resp. equivalence) relations on \mathbf{s} . Let \mathcal{A} (resp. \mathbf{F}) be a reduction (resp. equivalence) relation on \mathbf{S} and let consider a relation $\mathcal{R} \subseteq \mathbf{s} \times \mathbf{S}$. Suppose that for all u, v, U*

- (P0): *$u \mathcal{R} U$ & $u \mathbf{E} v$ imply $\exists V$ s.t. $v \mathcal{R} V$ & $U \mathbf{F} V$.*
- (P1): *$u \mathcal{R} U$ & $u \mathcal{A}_1 v$ imply $\exists V$ s.t. $v \mathcal{R} V$ & $U \mathcal{A}^* V$.*
- (P2): *$u \mathcal{R} U$ & $u \mathcal{A}_2 v$ imply $\exists V$ s.t. $v \mathcal{R} V$ & $U \mathcal{A}^+ V$.*
- (P3): *The relation \mathcal{A}_1 modulo \mathbf{E} is well-founded.*

Then, $t \mathcal{R} T$ & $T \in \mathcal{SN}_{\mathcal{A}/\mathbf{F}}$ imply $t \in \mathcal{SN}_{(\mathcal{A}_1 \cup \mathcal{A}_2)/\mathbf{E}}$.

Proof. Suppose $t \notin \mathcal{SN}_{(\mathcal{A}_1 \cup \mathcal{A}_2)/\mathbf{E}}$. Then, there is an infinite $(\mathcal{A}_1 \cup \mathcal{A}_2)/\mathbf{E}$ reduction starting at t , and since \mathcal{A}_1 modulo \mathbf{E} is a well-founded relation by (P3), this reduction has necessarily the form:

$$t \rightarrow_{\mathcal{A}_1/\mathbf{E}}^* t_1 \rightarrow_{\mathcal{A}_2/\mathbf{E}}^+ t_2 \rightarrow_{\mathcal{A}_1/\mathbf{E}}^* t_3 \rightarrow_{\mathcal{A}_2/\mathbf{E}}^+ t_4 \rightarrow_{\mathcal{A}_1/\mathbf{E}}^* \dots$$

And can be projected by (P0), (P1) and (P2) into an infinite \mathcal{A} reduction sequence as follows:

$$\begin{array}{cccccccc} t & \rightarrow_{\mathcal{A}_1/\mathbb{E}}^* & t_1 & \rightarrow_{\mathcal{A}_2/\mathbb{E}}^+ & t_2 & \rightarrow_{\mathcal{A}_1/\mathbb{E}}^* & t_3 & \rightarrow_{\mathcal{A}_2/\mathbb{E}}^+ & t_4 & \rightarrow_{\mathcal{A}_1/\mathbb{E}}^* \dots \\ \mathcal{R} & & \mathcal{R} & & \mathcal{R} & & \mathcal{R} & & \mathcal{R} & \\ T & \rightarrow_{\mathcal{A}/\mathbb{F}}^* & T_1 & \rightarrow_{\mathcal{A}/\mathbb{F}}^+ & T_2 & \rightarrow_{\mathcal{A}/\mathbb{F}}^* & T_3 & \rightarrow_{\mathcal{A}/\mathbb{F}}^+ & T_4 & \rightarrow_{\mathcal{A}/\mathbb{F}}^* \dots \end{array}$$

Since $T \in \mathcal{SN}_{\mathcal{A}/\mathbb{F}}$, then we get a contradiction. \square

3. THE STRUCTURAL λ j-CALCULUS

The **structural λ j-calculus** is given by a set of terms and a set of reduction rules. The set of λ j-terms, written \mathcal{T} , is generated by the following grammar:

$$(\mathcal{T}) \quad t, u ::= x \mid \lambda x.t \mid t u \mid t[x/u]$$

The term x is **variable**, $\lambda x.t$ an **abstraction**, tu an **application** and $t[x/u]$ a **substituted term**. The object $[x/u]$, which is not a term, is called a **jump**. The terms $\lambda x.t$ and $t[x/u]$ bind x in t , *i.e.* the sets of **free/bound variables** of a term are given by the following equations:

$$\begin{array}{ll} \mathbf{fv}(x) & := \{x\} & \mathbf{bv}(x) & := \emptyset \\ \mathbf{fv}(tu) & := \mathbf{fv}(t) \cup \mathbf{fv}(u) & \mathbf{bv}(tu) & := \mathbf{bv}(t) \cup \mathbf{bv}(u) \\ \mathbf{fv}(\lambda x.t) & := \mathbf{fv}(t) \setminus \{x\} & \mathbf{bv}(\lambda x.t) & := \mathbf{bv}(t) \cup \{x\} \\ \mathbf{fv}(t[x/u]) & := (\mathbf{fv}(t) \setminus \{x\}) \cup \mathbf{fv}(u) & \mathbf{bv}(t[x/u]) & := \mathbf{bv}(t) \cup \{x\} \cup \mathbf{bv}(u) \end{array}$$

A jump $[x/u]$ in a term $t[x/u]$ is called **void** if $x \notin \mathbf{fv}(t)$. The equivalence relation generated by the renaming of bound variables is called α -**conversion**. Thus for example $(\lambda y.x)[x/y] \equiv_{\alpha} (\lambda y'.x')[x'/y]$. We write \mathcal{T}_{λ} for the subset of \mathcal{T} only containing λ -terms. We write \bar{t}_n^1 for a set of terms $\{t_1, \dots, t_n\}$, $v\bar{t}_n^1$ for an application $(\dots(v t_1) t_2) \dots t_n$, $t_1 t_2 \dots t_n$ for $(\dots(t_1 t_2) \dots) t_n$ and $t[x_i/u_i]_m^1$ for $t[x_1/u_1] \dots [x_m/u_m]$.

The meta-level **substitution** operation is defined by induction on terms by using the following equations on α -equivalence classes:

$$\begin{array}{ll} x\{x/u\} & := u \\ y\{x/u\} & := y \\ (\lambda y.t)\{x/u\} & := \lambda y.t\{x/u\} \quad \text{if } y \notin \mathbf{fv}(u) \\ (tv)\{x/u\} & := t\{x/u\}v\{x/u\} \\ t\{y/v\}\{x/u\} & := t\{x/u\}\{y/v\{x/u\}\} \quad \text{if } y \notin \mathbf{fv}(u) \end{array}$$

Positions of terms are defined as expected; $t|_p$ denotes the **subterm of t at position p** and $\text{pos}_x(t)$ denotes the **set of all the positions p of t s.t. $t|_p = x$** .

Contexts are generated by the following grammar:

$$C ::= [\] \mid Cv \mid vC \mid v[y/C] \mid C[y/v] \mid \lambda y.C$$

We write $C[t]$ to denote the term obtained by replacing the hole $[\]$ in C by the term t . Thus for example $\lambda x.z[y/w][x] = \lambda x.z[y/wx]$ (remark that capture of variables is possible).

The **binding set** of a context $C[\cdot]$ is defined by the following equations:

$$\begin{array}{ll} \mathbf{bs}([\cdot]) & := \emptyset & \mathbf{bs}(t[x/C[\cdot]]) & := \mathbf{bs}(C[\cdot]) \\ \mathbf{bs}(t C[\cdot]) & := \mathbf{bs}(C[\cdot]) & \mathbf{bs}(C[\cdot][x/v]) & := \mathbf{bs}(C[\cdot]) \cup \{x\} \\ \mathbf{bs}(C[\cdot] v) & := \mathbf{bs}(C[\cdot]) & \mathbf{bs}(\lambda x.C[\cdot]) & := \mathbf{bs}(C[\cdot]) \cup \{x\} \end{array}$$

We use $|t|$ to denote the size of t . We write $|t|_x$ for the number of free occurrences of the variable x in the term t , called the **multiplicity of x in t** . We extend this notion to sets of variables by $|t|_\Gamma := \sum_{x \in \Gamma} |t|_x$. A key notion used to define the semantics of the λj -calculus is that of renaming: given a term t and a subset $S \subseteq \text{pos}_x(t) \cap \text{fv}(t)$, we write $R_y^{S,x}(t)$ for the term t' verifying $t'|_p = t|_p$ if $p \notin S$ and $t'|_p = y$ if $(t|_p = x \ \& \ p \in S)$. Thus for example, $R_y^{\{111,2\},x}(x \ z \ x \ x) = y \ z \ x \ y$.

When $|t|_x = n \geq 2$, we write $t_{[y]_x}$ for any **non-deterministic replacement** of i ($1 \leq i \leq n-1$) occurrences of x in t by a *fresh* variable y , *i.e.* $t_{[y]_x}$ denotes any term $R_y^{S,x}(t)$ s.t. $|S| \geq 2$ and $S \subset \text{pos}_x(t)$. Thus for example, $(x \ x \ x \ x)_{[y]_x}$ may denote $(y \ x \ y \ x)$ or $(x \ y \ y \ x)$ but not $(y \ y \ y \ y)$.

We now consider the rewriting rules in Figure 1, where L is a list $[x_1/u_1] \dots [x_k/u_k]$ of jumps with $k \in \mathbb{N}$ (so potentially $k = 0$). We close these rules by contexts, as usual: $\rightarrow_{\mathcal{R}}$ denotes the contextual closure of $\mapsto_{\mathcal{R}}$, for $\mathcal{R} \subseteq \{\text{dB}, \text{w}, \text{d}, \text{c}\}$. We write $\rightarrow_{\neg \text{w}}$ for the reduction relation $\rightarrow_{\text{dB}, \text{d}, \text{c}}$. The **reduction relation** $\rightarrow_{\lambda j}$ (resp. \rightarrow_j) is generated by all (resp. all expect **dB**) the previous rewriting rules modulo α -conversion.

$$\begin{array}{llll}
\text{(Beta at a distance)} & (\lambda x.t)L \ u & \mapsto_{\text{dB}} & t[x/u]L \\
\text{(weakening)} & t[x/u] & \mapsto_{\text{w}} & t \quad \text{if } |t|_x = 0 \\
\text{(dereliction)} & t[x/u] & \mapsto_{\text{d}} & t\{x/u\} \quad \text{if } |t|_x = 1 \\
\text{(contraction)} & t[x/u] & \mapsto_{\text{c}} & t_{[y]_x}[x/u][y/u] \quad \text{if } |t|_x > 1
\end{array}$$

Figure 1: The λj -reduction system

In the rest of this section we shall prove the following properties of λj : full composition (Lemma 3.1), simulation of one step β -reduction (Lemma 3.3), termination and uniqueness of normal forms of the substitution calculus \rightarrow_j (Lemmas 3.8 and 3.9), postponement of erasing reductions (Lemma 3.11) and confluence of λj (Theorem 3.15).

3.1. Jumps and Multiplicities. The first property we show in this section states that any jump $[x/u]$ in a substituted term $t[x/u]$ can be reduced to its implicit form $t\{x/u\}$. There are two interesting points. The first is that in, contrast with most calculi of explicit substitutions, full composition holds with no need of equivalences. The second is that the proof *has to be* by induction on $|t|_x$, since an induction on the structure of t does not work.

Lemma 3.1 (Full Composition (FC)). *Let $t, u \in \mathcal{T}$. Then $t[x/u] \rightarrow_j^+ t\{x/u\}$. Moreover, $|t|_x \geq 1$ implies $t[x/u] \rightarrow_{\text{d}, \text{c}}^+ t\{x/u\}$.*

Proof. By induction on $|t|_x$.

- If $|t|_x = 0$, then $t[x/u] \rightarrow_{\text{w}} t = t\{x/u\}$.
- If $|t|_x = 1$, then $t[x/u] \rightarrow_{\text{d}} t\{x/u\}$.
- If $|t|_x \geq 2$, then

$$\begin{array}{llll}
t[x/u] & \rightarrow_{\text{c}} & t_{[y]_x}[y/u][x/u] & \rightarrow_j^+ \text{ (i.h.)} \\
& & t_{[y]_x}\{y/u\}[x/u] & \rightarrow_j^+ \text{ (i.h.)} \\
& & t_{[y]_x}\{y/u\}\{x/u\} & = t\{x/u\}
\end{array}$$

□

Due to the very general form of the duplication rule of λ_j , we get the following corollary which together with full composition can be seen as a generalised composition property:

Corollary 3.2. *Given $S \subset \text{pos}_x(t)$ s.t. $|S| \geq 2$, then $t[x/u] \rightarrow_j^+ R_y^{S,x}(t)\{y/u\}[x/u]$, where y is a fresh variable.*

Proof. The term $t[x/u]$ c-reduces to the term $R_y^{S,x}(t)[y/u][x/u]$. Then we conclude by full composition. \square

Thus for example $(x (x x))[x/u] \rightarrow_{\lambda_j}^+ (x (u x))[x/u]$. Note that this property is not enjoyed by traditional explicit substitution calculi: for instance, in λx [4], the term $(x (x x))[x/u]$ cannot be reduced to $(x (u x))[x/u]$. However, it holds in calculi with partial substitutions, as Milner's calculus λ_{sub} [30]. It is not difficult (see e.g. [19]) to define a translation T on terms such that $t \rightarrow_{\lambda_{\text{sub}}} t'$ implies $T(t) \rightarrow_{\lambda_j}^+ T(t')$. This property allows in particular to deduce normalisation properties for λ_{sub} from those of λ_j .

The one-step simulation of λ -calculus follows directly from full composition:

Lemma 3.3 (Simulation of λ -calculus). *Let $t \in \mathcal{T}_\lambda$. If $t \rightarrow_\beta t'$ then $t \rightarrow_{\lambda_j}^+ t'$.*

Proof. By induction on $t \rightarrow_\beta t'$. Let $t = (\lambda x.u)v \rightarrow_\beta u\{x/v\}$, then $t \rightarrow_{\text{aB}} u[x/v] \rightarrow_j^+ (L. 3.1) u\{x/v\}$. All the other cases are straightforward. \square

The following notion will be useful in various proofs. The idea is that it counts the maximal number of free occurrences of a variable x that may appear during a j -reduction sequence from a term t .

The **potential multiplicity** of the variable x in the term t , written $P_x(t)$, is defined on α -equivalence classes as follows: if $x \notin \text{fv}(t)$, then $P_x(t) := 0$; otherwise:

$$\begin{aligned} P_x(x) &:= 1 \\ P_x(\lambda y.u) &:= P_x(u) \\ P_x(u v) &:= P_x(u) + P_x(v) \\ P_x(u[y/v]) &:= P_x(u) + \max(1, P_y(u)) \cdot P_x(v) \end{aligned}$$

We can formalise the intuition behind $P_x(t)$.

Lemma 3.4. *Let $t \in \mathcal{T}$. Then*

- $|t|_x \leq P_x(t)$.
- *If t is a c-nf then $|t|_x = P_x(t)$.*

Proof. Both points are by induction on the definition of $P_x(t)$. The only interesting case is when $t = u[y/v]$: the i.h. gives $|u|_x \leq P_x(u)$, $|u|_y \leq P_y(u)$ and $|v|_x \leq P_x(u)$, from which we conclude with the first point. For the second one, if t is a c-nf every relation given by the i.h. is an equality and $|u|_y = P_y(u) \leq 1$, otherwise there would be a c-redex. Then we get $P_x(t) = P_x(u) + \max(1, P_y(u)) \cdot P_x(v) = |u|_x + |v|_x = |t|_x$. \square

Potential multiplicities enjoy the following properties.

Lemma 3.5. *Let $t \in \mathcal{T}$.*

- (1) *If $u \in \mathcal{T}$ and $y \notin \text{fv}(u)$, then $P_y(t) = P_y(t\{x/u\})$.*
- (2) *If $|t|_x \geq 2$, then $P_z(t) = P_z(t_{[y]_x})$ and $P_x(t) = P_x(t_{[y]_x}) + P_y(t_{[y]_x})$.*
- (3) *If $t \rightarrow_j t'$, then $P_y(t) \geq P_y(t')$.*

Proof. By induction on t . □

By exploiting potential multiplicities we can measure the global degree of sharing of a given term, then prove that the j-reduction subsystem terminates.

We consider multisets of integers. We use \emptyset to denote the empty multiset, \sqcup to denote multiset union and $n \cdot [a_1, \dots, a_n]$ to denote $[n \cdot a_1, \dots, n \cdot a_n]$.

The **j-measure** of $t \in \mathcal{T}$, written $\mathbf{jm}(t)$, is given by:

$$\begin{aligned} \mathbf{jm}(x) &:= \emptyset \\ \mathbf{jm}(\lambda x.t) &:= \mathbf{jm}(t) \\ \mathbf{jm}(tu) &:= \mathbf{jm}(t) \sqcup \mathbf{jm}(u) \\ \mathbf{jm}(t[x/u]) &:= [\mathbf{P}_x(t)] \sqcup \mathbf{jm}(t) \sqcup \max(1, \mathbf{P}_x(t)) \cdot \mathbf{jm}(u) \end{aligned}$$

Potential multiplicities and the j-measure are stable by j-reduction, but can be incremented by dB-steps. For example, consider $t = (\lambda x.x \ x) \ y \rightarrow_{\text{dB}} (x \ x)[x/y] = t'$. We get $\mathbf{P}_y(t) = 1$, $\mathbf{P}_y(t') = 2$, $\mathbf{jm}(t) = \emptyset$ and $\mathbf{jm}(t') = [2]$.

Stability of potential multiplicities and the j-measure by j-reduction is proved as follows:

Lemma 3.6. *Let $t \in \mathcal{T}$. Then,*

- (1) $\mathbf{jm}(t) = \mathbf{jm}(t_{[y]_x})$.
- (2) If $u \in \mathcal{T}$, then $\mathbf{jm}(t) \sqcup \mathbf{jm}(u) \geq \mathbf{jm}(t\{x/u\})$.

Proof. By induction on t . The first property is straightforward so that we only show the second one.

- $t = x$. Then $\mathbf{jm}(x) \sqcup \mathbf{jm}(u) = \emptyset \sqcup \mathbf{jm}(u) = \mathbf{jm}(x\{x/u\})$.
- $t = y \neq x$. Then $\mathbf{jm}(y) \sqcup \mathbf{jm}(u) = \emptyset \sqcup \mathbf{jm}(u) \geq \emptyset = \mathbf{jm}(y\{x/u\})$.
- $t = t_1[y/t_2]$. W.l.g we assume $y \notin \text{fv}(u)$. Then,

$$\begin{aligned} &\mathbf{jm}(t_1[y/t_2]) \sqcup \mathbf{jm}(u) &&= \\ &[\mathbf{P}_y(t_1)] \sqcup \mathbf{jm}(t_1) \sqcup \max(1, \mathbf{P}_y(t_1)) \cdot \mathbf{jm}(t_2) \sqcup \mathbf{jm}(u) &&\geq_{i.h. \ \& \ L.3.5:1} \\ &[\mathbf{P}_y(t_1\{x/u\})] \sqcup \mathbf{jm}(t_1\{x/u\}) \sqcup \max(1, \mathbf{P}_y(t_1\{x/u\})) \cdot \mathbf{jm}(t_2\{x/u\}) &&= \\ &\mathbf{jm}(t_1\{x/u\}[y/t_2\{x/u\}]) && \end{aligned}$$

- All the other cases are straightforward. □

Lemma 3.7. *Let $t_0 \in \mathcal{T}$. Then,*

- (1) $t_0 \equiv_{\alpha} t_1$ implies $\mathbf{jm}(t_0) = \mathbf{jm}(t_1)$.
- (2) $t_0 \rightarrow_j t_1$ implies $\mathbf{jm}(t_0) > \mathbf{jm}(t_1)$.

Proof. By induction on the relations. The first point is straightforward, hence we only show the second one.

- $t_0 = t[x/u] \rightarrow_w t = t_1$, with $|t|_x = 0$. Then $\mathbf{jm}(t_0) = \mathbf{jm}(t) \sqcup 1 \cdot \mathbf{jm}(u) \sqcup [0] > \mathbf{jm}(t) = \mathbf{jm}(t_1)$.
- $t_0 = t[x/u] \rightarrow_d t\{x/u\} = t_1$, with $|t|_x = 1$.
Then $\mathbf{jm}(t_0) = \mathbf{jm}(t) \sqcup 1 \cdot \mathbf{jm}(u) \sqcup [1] > \mathbf{jm}(t) \sqcup \mathbf{jm}(u) \geq_{L. \ 3.6:2} \mathbf{jm}(t\{x/u\}) = \mathbf{jm}(t_1)$.

- $t_0 = t[x/u] \rightarrow_c t_{[y]x}[x/u][y/u] = t_1$, with $|t|_x \geq 2$ and y fresh. Then,

$$\begin{aligned}
& \mathbf{jm}(t_0) & & = \\
& \mathbf{jm}(t) \sqcup \mathbf{P}_x(t) \cdot \mathbf{jm}(u) \sqcup [\mathbf{P}_x(t)] & & = \\
& \mathbf{jm}(t) \sqcup (\mathbf{P}_x(t_{[y]x}) + \mathbf{P}_y(t_{[y]x})) \cdot \mathbf{jm}(u) \sqcup [\mathbf{P}_x(t)] & & =_{L.3.6:1} \\
& \mathbf{jm}(t_{[y]x}) \sqcup (\mathbf{P}_x(t_{[y]x}) + \mathbf{P}_y(t_{[y]x})) \cdot \mathbf{jm}(u) \sqcup [\mathbf{P}_x(t)] & & > \\
& \mathbf{jm}(t_{[y]x}) \sqcup \mathbf{P}_x(t_{[y]x}) \cdot \mathbf{jm}(u) \sqcup [\mathbf{P}_x(t_{[y]x})] \sqcup \mathbf{P}_y(t_{[y]x}) \cdot \mathbf{jm}(u) \sqcup [\mathbf{P}_y(t_{[y]x})] & & = \\
& \mathbf{jm}(t_{[y]x}) \sqcup \mathbf{P}_x(t_{[y]x}) \cdot \mathbf{jm}(u) \sqcup [\mathbf{P}_x(t_{[y]x})] \sqcup \mathbf{P}_y(t_{[y]x}[x/u]) \cdot \mathbf{jm}(u) \sqcup [\mathbf{P}_y(t_{[y]x}[x/u])] & & = \\
& \mathbf{jm}(t_{[y]x}[x/u]) \sqcup \mathbf{P}_y(t_{[y]x}[x/u]) \cdot \mathbf{jm}(u) \sqcup [\mathbf{P}_y(t_{[y]x}[x/u])] & & = \mathbf{jm}(t_1)
\end{aligned}$$

- $t_0 = t[x/u] \rightarrow t'[x/u] = t_1$, where $t \rightarrow t'$. Then

$$\begin{aligned}
& \mathbf{jm}(t_0) & & = \\
& \mathbf{jm}(t) \sqcup \max(1, \mathbf{P}_x(t)) \cdot \mathbf{jm}(u) \sqcup [\mathbf{P}_x(t)] & & >_{i.h.} \\
& \mathbf{jm}(t') \sqcup \max(1, \mathbf{P}_x(t)) \cdot \mathbf{jm}(u) \sqcup [\mathbf{P}_x(t)] & & \geq_{L. 3.5:3} \\
& \mathbf{jm}(t') \sqcup \max(1, \mathbf{P}_x(t')) \cdot \mathbf{jm}(u) \sqcup [\mathbf{P}_x(t')] & & = \mathbf{jm}(t_1)
\end{aligned}$$

- All the other cases are straightforward

□

The last lemma obviously implies:

Lemma 3.8. *The j-calculus terminates.*

Moreover:

Lemma 3.9. *Let $t \in \mathcal{T}$. Then, $\mathbf{j}(t)$ is the unique j-nf of t . Moreover, j-normal forms are λ -terms and verify the following properties:*

$$\begin{aligned}
\mathbf{j}(x) &= x & \mathbf{j}(u v) &= \mathbf{j}(u) \mathbf{j}(v) \\
\mathbf{j}(\lambda x.u) &= \lambda x.\mathbf{j}(u) & \mathbf{j}(u[x/v]) &= \mathbf{j}(u)\{x/\mathbf{j}(v)\}
\end{aligned}$$

Proof. One easily shows that \rightarrow_j is locally confluent, then Lemma 3.8 allows to apply Newman's Lemma to conclude with the first part of the statement. The second part can be shown by induction on the structure of terms. Particularly, when $t = u[x/v]$ one has $u[x/v] \rightarrow_j^* \mathbf{j}(u)[x/\mathbf{j}(v)] \rightarrow_j^+ (L. 3.1) \mathbf{j}(u)\{x/\mathbf{j}(v)\}$. It is then sufficient to notice that j-normal forms are stable by substitutions of j-normal forms. □

We conclude this section by showing another important property of $\lambda\mathbf{j}$ concerning the postponement of erasing steps. We need the following lemma:

Lemma 3.10. *Let $t \in \mathcal{T}$. Then:*

- (1) $t \rightarrow_{\mathbf{w}} \rightarrow_{\neg \mathbf{w}} t'$ implies $t \rightarrow_{\neg \mathbf{w}} \rightarrow_{\mathbf{w}}^+ t'$.
- (2) $t \rightarrow_{\mathbf{w}}^+ \rightarrow_{\neg \mathbf{w}} t'$ implies $t \rightarrow_{\neg \mathbf{w}} \rightarrow_{\mathbf{w}}^+ t'$

Proof. Point 1 is by induction on the relations and case analysis. Point 2 is by induction on the length of $\rightarrow_{\mathbf{w}}^+$ using Point 1. □

Let us use $\tau : t \rightarrow^* t'$ as a notation for a reduction sequence, the symbol ';' for the concatenation of reduction sequences and $|\tau|_{\neg w}$ for the number of $\rightarrow_{\neg w}$ steps in τ . Then we obtain:

Lemma 3.11 (w -postponement). *Let $t \in \mathcal{T}$. If $\tau : t \rightarrow_{\lambda_j}^* t'$ then $\exists \tau' : t \rightarrow_{\neg w}^* \rightarrow_w^* t'$ s.t. $|\tau|_{\neg w} = |\tau'|_{\neg w}$.*

Proof. By induction on $k = |\tau|_{\neg w}$. The case $k = 0$ is straightforward. Let $k > 0$. If $\tau : t \rightarrow_{\neg w} u \rightarrow_{\lambda_j}^* t'$ then simply conclude using the i.h. on the sub-reduction $\rho : u \rightarrow_{\lambda_j}^* t'$. Otherwise τ starts with a w -step. If all the steps in τ are w , then we trivially conclude. Otherwise $\tau = \tau_w; \rightarrow_{\neg w}; \rho$ where τ_w is the maximal prefix of τ made out of weakening steps only. By Lemma 3.10.4 we get that $t \rightarrow_{\neg w} \rightarrow_w^+; \rho t'$ and we conclude by applying the i.h. to $\rightarrow_w^+; \rho$. \square

3.2. Confluence. Confluence of calculi with ES can be easily proved by using Tait and Martin L of's technique (see for example the case of λ_{es} [17]). This technique is based on the definition of a simultaneous reduction relation which enjoys the diamond property. It is completely standard so we give the statements of the lemmas and omit the proofs.

The **simultaneous reduction relation** \Rightarrow_{λ_j} is defined on terms in j -normal form as follows:

- $x \Rightarrow_{\lambda_j} x$
- If $t \Rightarrow_{\lambda_j} t'$, then $\lambda x.t \Rightarrow_{\lambda_j} \lambda x.t'$
- If $t \Rightarrow_{\lambda_j} t'$ & $u \Rightarrow_{\lambda_j} u'$, then $t u \Rightarrow_{\lambda_j} t' u'$
- If $t \Rightarrow_{\lambda_j} t'$ and $u \Rightarrow_{\lambda_j} u'$, then $(\lambda x.t) u \Rightarrow_{\lambda_j} j(t'[x/u'])$

A first lemma ensures that \Rightarrow_{λ_j} can be simulated by \rightarrow_{λ_j} .

Lemma 3.12. *If $t \Rightarrow_{\lambda_j} t'$, then $t \rightarrow_{\lambda_j}^* t'$.*

Proof. By induction on $t \Rightarrow_{\lambda_j} t'$. \square

A second lemma ensures that \rightarrow_{λ_j} can be projected through $j(\cdot)$ on \Rightarrow_{λ_j} .

Lemma 3.13. *If $t \rightarrow_{\lambda_j} t'$, then $j(t) \Rightarrow_{\lambda_j} j(t')$.*

Proof. By induction on $t \rightarrow_{\lambda_j} t'$. \square

The two lemmas combined essentially say that \Rightarrow_{λ_j} is confluent if and only if $\rightarrow_{\lambda_j}^*$ is confluent. Then we show the diamond property for \Rightarrow_{λ_j} , which implies that \rightarrow_{λ_j} is confluent:

Lemma 3.14. *The relation \Rightarrow_{λ_j} enjoys the diamond property.*

Proof. By induction on \Rightarrow_{λ_j} and case analysis. \square

Then we conclude:

Theorem 3.15 (Confluence). *For all $i \in \{1, 2\}$, for all $t, u_i \in \mathcal{T}$ s.t. $t \rightarrow_{\lambda_j}^* u_i$, $\exists v$ s.t. $u_i \rightarrow_{\lambda_j}^* v$.*

Proof. Let $t \rightarrow_{\lambda_j}^* t_i$ for $i = 1, 2$. Lemma 3.13 gives $j(t) \Rightarrow_{\lambda_j}^* j(t_i)$ for $i = 1, 2$. Lemma 3.14 implies \Rightarrow_{λ_j} is confluent so that $\exists s$ such that $j(t_i) \Rightarrow_{\lambda_j}^* s$ for $i = 1, 2$. We can then close the diagram with $t_i \rightarrow_j^* j(t_i) \rightarrow_{\lambda_j}^* s$ by Lemma 3.12. \square

While confluence holds for all calculi with explicit substitutions, **metaconfluence** does not. The idea is to switch to an enriched language with a new kind of (meta)variable of the form X^Δ , to be intended as a named context hole accepting to be replaced by terms whose free variables are among Δ . This form of metaterms are for example used in the framework of higher-order unification [15]. In presence of meta-variables not all the substitutions can be computed. Thus for instance in the metaterm $X^y[y/z]$ the jump $[y/z]$ is blocked. Consider:

$$(X^{\{z_1\}} Y^{\{z_2\}})[z_1/x][z_2/x] \leftarrow_c (X^{\{z\}} Y^{\{z\}})[z/x] \rightarrow_c (X^{\{z_1\}} Y^{\{z_2\}})[z_2/x][z_1/x]$$

These metaterms are different normal forms. However, it is enough to add the following equation to recover confluence.

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

A proof of confluence of λj modulo **CS** for metaterms can be found in [34].

4. PRESERVATION OF β -STRONG NORMALIZATION FOR λj

A reduction system \mathcal{R} for a language containing the set \mathcal{T}_λ of all λ -terms is said to enjoy the **PSN property** iff every λ -term which is β -strongly normalizing is also \mathcal{R} -strongly normalizing. Formally, for all $t \in \mathcal{T}_\lambda$, if $t \in \mathcal{SN}_\beta$, then $t \in \mathcal{SN}_\mathcal{R}$.

The PSN property, when it holds, is usually non-trivial to prove. We are going to show that λj enjoys PSN by giving a particularly compact proof. This proof technique has been developed by D. Kesner, it reduces PSN to a property called **IE**, which relates termination of **Implicit** substitution to termination of **Explicit** substitution. It is an abstract technique not depending on the particular rules of the calculus with explicit substitutions.

A reduction system \mathcal{R} for a language $\mathcal{T}_\mathcal{R}$ containing the set \mathcal{T}_λ is said to enjoy the **IE property** iff for $n \geq 0$ and for all $t, u, \bar{v}_n^1 \in \mathcal{T}_\lambda$:

$$u \in \mathcal{SN}_\mathcal{R} \ \& \ t\{x/u\}\bar{v}_n^1 \in \mathcal{SN}_\mathcal{R} \ \& \ t[x/u]\bar{v}_n^1 \in \mathcal{T}_\mathcal{R} \quad \text{imply} \quad t[x/u]\bar{v}_n^1 \in \mathcal{SN}_\mathcal{R}$$

Of course one generally considers a system \mathcal{R} which can simulate the λ -calculus, so that the following properties seem to be natural requirements to get PSN.

Theorem 4.1 (Natural Requirements for PSN). *Let \mathcal{R} be a calculus verifying the following facts:*

- (F0) *If $\bar{t}_n^1 \in \mathcal{T}_\lambda \cap \mathcal{SN}_\mathcal{R}$, then $x \bar{t}_n^1 \in \mathcal{SN}_\mathcal{R}$.*
- (F1) *If $u \in \mathcal{T}_\lambda \cap \mathcal{SN}_\mathcal{R}$, then $\lambda x.u \in \mathcal{SN}_\mathcal{R}$.*
- (F2) *If $v \in \mathcal{T}_\lambda \cap \mathcal{SN}_\mathcal{R}$ & $u\{x/v\}\bar{t}_n^1 \in \mathcal{T}_\lambda \cap \mathcal{SN}_\mathcal{R}$, then $(\lambda x.u)v\bar{t}_n^1 \in \mathcal{SN}_\mathcal{R}$.*

Then, \mathcal{R} enjoys PSN.

Proof. We show that $t \in \mathcal{SN}_\beta$ implies $t \in \mathcal{SN}_\mathcal{R}$ by induction

the pair $\langle \eta_\beta(t), |t| \rangle$, using the lexicographic ordering. We reason by cases.

- If $t = x \bar{t}_n^1$, then $t_i \in \mathcal{SN}_\beta$ and $\langle \eta_\beta(t_i), |t_i| \rangle < \langle \eta_\beta(t), |t| \rangle$. We have $t_i \in \mathcal{SN}_\mathcal{R}$ by the i.h. and thus $x \bar{t}_n^1 \in \mathcal{SN}_\mathcal{R}$ by fact **F0**.
- If $t = \lambda x.u$, then $u \in \mathcal{SN}_\beta$ and $\langle \eta_\beta(u), |u| \rangle < \langle \eta_\beta(t), |t| \rangle$. We have $u \in \mathcal{SN}_\mathcal{R}$ by the i.h. and thus $\lambda x.u \in \mathcal{SN}_\mathcal{R}$ by fact **F1**.
- If $t = (\lambda x.u)v \bar{t}_n^1 \in \mathcal{SN}_\beta$, then $u\{x/v\}\bar{t}_n^1 \in \mathcal{SN}_\beta$ and $v \in \mathcal{SN}_\beta$. Indeed, $\eta_\beta(u\{x/v\}\bar{t}_n^1) < \eta_\beta(t)$ and $\eta_\beta(v) < \eta_\beta(t)$. We have that both terms are in $\mathcal{SN}_\mathcal{R}$ by the i.h. Then **F2** guarantees that $t \in \mathcal{SN}_\mathcal{R}$.

□

Now we show that λ_j satisfies the three natural requirements of the last theorem, and thus it satisfies PSN.

Lemma 4.2 (Adequacy of **IE**). *If λ_j verifies **IE**, then λ_j satisfies PSN.*

Proof. By Theorem 4.1 it is sufficient to show **F0**, **F1** and **F2**. The first two properties are straightforward. For the third one, assume $v \in \mathcal{T}_\lambda \cap \mathcal{SN}_{\lambda_j}$ and $u\{x/v\} \bar{t}_n^1 \in \mathcal{T}_\lambda \cap \mathcal{SN}_{\lambda_j}$. Then in particular $u, v, \bar{t}_n^1 \in \mathcal{T}_\lambda \cap \mathcal{SN}_{\lambda_j}$. We show that $t = (\lambda x.u) v \bar{t}_n^1 \in \mathcal{SN}_{\lambda_j}$ by induction on $\eta_{\lambda_j}(u) + \eta_{\lambda_j}(v) + \sum_i \eta_{\lambda_j}(t_i)$. For that, it is sufficient to show that every λ_j -reduct of t is in \mathcal{SN}_{λ_j} . If the λ_j -reduct of t is internal we conclude by the i.h. Otherwise $t = u[x/v]\bar{t}_n^1$ which is in \mathcal{SN}_{λ_j} by the **IE** property. □

As a consequence, in order to get PSN for λ_j we only need to prove the **IE** property. For that, we first generalise the **IE** property in order to deal with possibly many substitutions.

A reduction system \mathcal{R} for a language $\mathcal{T}_\mathcal{R}$ containing the set \mathcal{T}_λ is said to enjoy the **Generalised IE property**, written **GIE**, iff for all t, \bar{u}_m^1 ($m \geq 1$), \bar{v}_n^1 ($n \geq 0$) in $\mathcal{T}_\mathcal{R}$, if $\bar{u}_m^1 \subseteq \mathcal{SN}_\mathcal{R}$ & $t\{x_i/u_i\}_m^1 \bar{v}_n^1 \in \mathcal{SN}_\mathcal{R}$, then $t[x_i/u_i]_m^1 \bar{v}_n^1 \in \mathcal{SN}_{\lambda_j}$, where $x_i \neq x_j$ for $i, j = 1 \dots m$ and $x_i \notin \text{fv}(u_j)$ for $i, j = 1 \dots m$.

Theorem 4.3 (**GIE** for λ_j). *The λ_j -calculus enjoys the **GIE** property.*

Notation: To improve readability of the proof we shall abbreviate the notation $[x_i/u_i]_m^1$ by $[\cdot]_m^1$. Similarly for implicit substitutions.

Proof. Suppose $\bar{u}_m^1 \in \mathcal{SN}_{\lambda_j}$ & $t\{x_i/u_i\}_m^1 \bar{v}_n^1 \in \mathcal{SN}_{\lambda_j}$. We show $t_0 = t[x_i/u_i]_m^1 \bar{v}_n^1 \in \mathcal{SN}_{\lambda_j}$ by induction on

$$\langle \eta_{\lambda_j}(t\{x_i/u_i\}_m^1 \bar{v}_n^1), \mathbf{v}_{\bar{x}_m^1}(t), \eta_{\lambda_j}(\bar{u}_m^1) \rangle$$

where $\mathbf{v}_{x_i}(t) = 3^{|t|_{x_i}}$ and $\mathbf{v}_{\bar{x}_m^1}(t) = \sum_{i \in m} \mathbf{v}_{x_i}(t)$.

To show $t_0 \in \mathcal{SN}_{\lambda_j}$ it is sufficient to show that every λ_j -reduct of t_0 is in \mathcal{SN}_{λ_j} .

- $t_0 \rightarrow_{\lambda_j} t[\cdot]_{j-1}^1 [x_j/u_j][\cdot]_m^{j+1} \bar{v}_n^1 = t'_0$ with $u_j \rightarrow_{\lambda_j} u'_j$. Then we get:
 - $\eta_{\lambda_j}(t\{\cdot\}_{j-1}^1 \{x_j/u'_j\} \{\cdot\}_m^{j+1} \bar{v}_n^1) \leq \eta_{\lambda_j}(t\{\cdot\}_m^1 \bar{v}_n^1)$,
 - $\mathbf{v}_{\bar{x}_m^1}$ does not change, and
 - $\eta_{\lambda_j}(\bar{u}_{j-1}^1 u'_j \bar{u}_m^{j+1}) < \eta_{\lambda_j}(\bar{u}_m^1)$.

We conclude by the i.h. since $\bar{u}_{j-1}^1 u'_j \bar{u}_m^{j+1} \in \mathcal{SN}_{\lambda_j}$ and our hypothesis $t\{x_i/u_i\}_m^1 \bar{v}_n^1 \in \mathcal{SN}_{\lambda_j}$ is equal or reduces to $t\{\cdot\}_{j-1}^1 \{x_j/u'_j\} \{\cdot\}_m^{j+1} \bar{v}_n^1 \in \mathcal{SN}_{\lambda_j}$ (depending on $|t|_{x_j}$).

- $t_0 \rightarrow_{\lambda_j} t'[\cdot]_m^1 \bar{v}_n^1 = t'_0$ with $t \rightarrow_{\lambda_j} t'$. Then we have that

$$\eta_{\lambda_j}(t'\{\cdot\}_m^1 \bar{v}_n^1) < \eta_{\lambda_j}(t\{\cdot\}_m^1 \bar{v}_n^1)$$

We conclude by the i.h. since $t'\{\cdot\}_m^1 \bar{v}_n^1 \in \mathcal{SN}_{\lambda_j}$.

- $t_0 \rightarrow_{\lambda_j} t[\cdot]_m^1 v_1 \dots v'_i \dots v_n = t'_0$ with $v_i \rightarrow_{\lambda_j} v'_i$. Then we have that

$$\eta_{\lambda_j}(t\{\cdot\}_m^1 v_1 \dots v'_i \dots v_n) < \eta_{\lambda_j}(t\{\cdot\}_m^1 \bar{v}_n^1)$$

We conclude by the i.h. since $t\{\cdot\}_m^1 v_1 \dots v'_i \dots v_n \in \mathcal{SN}_{\lambda_j}$.

- $t_0 \rightarrow_w t[\cdot]_{j-1}^1 [\cdot]_m^{j+1} \bar{v}_n^1 = t'_0$, with $|t|_{x_j} = 0$. Then we have that

$$\eta_{\lambda_j}(t\{\cdot\}_{j-1}^1 \{\cdot\}_m^{j+1} \bar{v}_n^1) = \eta_{\lambda_j}(t\{\cdot\}_m^1 \bar{v}_n^1)$$

But $\mathbf{v}_{\bar{x}_{j-1}^1 \bar{x}_m^{j+1}}(t) < \mathbf{v}_{\bar{x}_m^1}(t)$. We conclude by the i.h. since $t\{\cdot\}_{j-1}^1 \{\cdot\}_m^{j+1} \bar{v}_n^1 = t\{\cdot\}_m^1 \bar{v}_n^1 \in \mathcal{SN}_{\lambda_j}$ by hypothesis.

- $t_0 \rightarrow_a t[\cdot]_{j-1}^1 \{x_j/u_j\} [\cdot]_m^{j+1} \bar{v}_n^1 = t'_0$ with $|t|_{x_j} = 1$. Notice that $t'_0 = t\{x_j/u_j\} [\cdot]_{j-1}^1 [\cdot]_m^{j+1} \bar{v}_n^1$. Then we get

$$\eta_{\lambda_j}(t\{x_j/u_j\} \{\cdot\}_{j-1}^1 \{\cdot\}_m^{j+1} \bar{v}_n^1) = \eta_{\lambda_j}(t\{\cdot\}_m^1 \bar{v}_n^1)$$

Since the jumps are independent, then $(\bar{x}_{j-1}^1 \cup \bar{x}_m^{j+1}) \cap \mathbf{fv}(u_j) = \emptyset$ implies $\mathbf{v}_{\bar{x}_{j-1}^1 \bar{x}_m^{j+1}}(t\{x_j/u_j\}) < \mathbf{v}_{\bar{x}_m^1}(t)$. We conclude since $t\{\cdot\}_{j-1}^1 \{x_j/u_j\} \{\cdot\}_m^{j+1} \bar{v}_n^1 = t\{\cdot\}_m^1 \bar{v}_n^1 \in \mathcal{SN}_{\lambda_j}$ by hypothesis.

- $t_0 \rightarrow_c t_{[y]_{x_j}} [\cdot]_{j-1}^1 [x_j/u_j] [y/u_j] [\cdot]_m^{j+1} \bar{v}_n^1 = t'_0$ with $|t|_{x_j} \geq 2$ and y fresh. Then,

$$\eta_{\lambda_j}(t_{[y]_{x_j}} \{\cdot\}_{j-1}^1 \{x_j/u_j\} \{y/u_j\} \{\cdot\}_m^{j+1} \bar{v}_n^1) = \eta_{\lambda_j}(t\{\cdot\}_m^1 \bar{v}_n^1)$$

$\mathbf{v}_{\bar{x}_{j-1}^1 x_j y \bar{x}_m^{j+1}}(t_{[y]_{x_j}}) < \mathbf{v}_{\bar{x}_m^1}(t)$. In order to apply the i.h. to $t_{[y]_{x_j}}$ we need:

- $\bar{u}_{j-1}^1, u_j, u_j, \bar{u}_m^{j+1} \in \mathcal{SN}_{\lambda_j}$. This holds by hypothesis.
- $t_{[y]_{x_1}} \{\cdot\}_{j-1}^1 \{x_j/u_j\} \{y/u_j\} \{\cdot\}_m^{j+1} \bar{v}_n^1 \in \mathcal{SN}_{\lambda_j}$. This holds since the term is equal to $t\{\cdot\}_m^1 \bar{v}_n^1$ which is \mathcal{SN}_{λ_j} by hypothesis.

Notice that this is the case that forces the use of a generalised sequence of substitutions: if we were proving the statement for $t[x/u] \bar{v}_n^1$ using as hypothesis $u \in \mathcal{SN}_{\lambda_j}$ & $t\{x/u\} \bar{v}_n^1 \in \mathcal{SN}_{\lambda_j}$ then there would be no way to use the i.h. to get $t_{[y]_x}[x/u][y/u] \bar{v}_n^1 \in \mathcal{SN}_{\lambda_j}$.

- $t_0 = (\lambda x.t') [\cdot]_m^1 v_1 \bar{v}_n^2 \rightarrow_{dB} t'[x/v_1] [\cdot]_m^1 \bar{v}_n^2 = t'_0$. We have that

$$u_0 = (\lambda x.t') \{\cdot\}_m^1 v_1 \bar{v}_n^2 \in \mathcal{SN}_{\lambda_j}$$

holds by hypothesis. Using full composition we obtain

$$u_0 \rightarrow_{dB} t' \{\cdot\}_m^1 [x/v_1] \bar{v}_n^2 \rightarrow_{\lambda_j}^+ t' \{\cdot\}_m^1 \{x/v_1\} \bar{v}_n^2 = t' \{x/v_1\} \{\cdot\}_m^1 \bar{v}_n^2 = u'_0$$

Thus $\eta_{\lambda_j}(u'_0) < \eta_{\lambda_j}(u_0)$. To conclude $t'_0 \in \mathcal{SN}_{\lambda_j}$ by the i.h. we then need

- $v_1, \bar{u}_m^1 \in \mathcal{SN}_{\lambda_j}$. But $\bar{u}_m^1 \in \mathcal{SN}_{\lambda_j}$ holds by hypothesis and $t\{x_i/u_i\}_m^1 \bar{v}_n^1 \in \mathcal{SN}_{\lambda_j}$ implies $v_1 \in \mathcal{SN}_{\lambda_j}$.
- $u'_0 = t' \{x/v_1\} \{\cdot\}_m^1 \bar{v}_n^2 \in \mathcal{SN}_{\lambda_j}$ which holds since $\eta_{\lambda_j}(u'_0) < \eta_{\lambda_j}(u_0)$.

□

The following is a consequence of Theorem 4.3: just take the number of substitutions m to be 1 and consider only the **GIE** property for $\mathcal{T}_\lambda \subset \mathcal{T}$.

Corollary 4.4 (IE for λ_j). *The λ_j -calculus enjoys the **IE** property.*

Corollary 4.4, then Lemma 4.2 and finally Theorem 4.1 imply:

Corollary 4.5 (PSN for λ_j). *The λ_j -calculus enjoys PSN, i.e. if $t \in \mathcal{T}_\lambda \cap \mathcal{SN}_\beta$, then $t \in \mathcal{SN}_{\lambda_j}$.*

5. AN EQUATIONAL THEORY FOR λ_j

Sections 3 and 4 show that the basic theory of λ_j enjoys good properties such as full composition, confluence and PSN. In most calculi with explicit substitutions, where substitutions are propagated through constructors and do not act at distance, full composition can only be obtained by adding an equivalence relation \equiv_{cs} defined as the contextual and reflexive-transitive closure of the following equation:

$$t[x/s][y/v] \sim_{cs} t[y/v][x/s] \quad \text{if } x \notin \mathbf{fv}(v) \text{ \& } y \notin \mathbf{fv}(s)$$

Otherwise a term like $x[y/z][x/w]$ cannot reduce to its implicit form $w[y/z] = x[y/z]\{x/w\}$ (and so full composition does not hold). Interestingly, λj enjoys full composition without using equation **CS**, which is remarkable since plain rewriting is much easier than rewriting modulo an equivalence relation.

However, as mentioned at the end of Section 3.2, the equation **CS** is necessary to recover confluence on metaterms. It is then natural to wonder what happens when \equiv_{CS} is added to λj . The answer is extremely positive since \equiv_{CS} preserves all the good properties of λj , and this holds in a very strong sense. In fact, \equiv_{CS} is a strong bisimulation for $(\mathcal{T}, \rightarrow_{\lambda j})$ (cf. Lemma 5.2), so that \equiv_{CS} can be postponed w.r.t. λj (cf. Lemma 2.2).

As already mentioned in the introduction, λj -terms and λj -dags [1] are strongly bisimilar, but the translation of λj -terms to λj -dags is not injective, *i.e.* there are different λj -terms which are mapped to the *same* λj -dag. It is then interesting to characterise the quotient induced by the translation [1], which turns out to be \equiv_{CS} : indeed $t \equiv_{\text{CS}} u$ if and only if t and u are mapped to the same λj -dag G , and since they both behave like G (*i.e.* are strongly bisimilar to G), then they behave the same (*i.e.* they are strongly bisimilar).

The λj calculus is also interesting since it can be mapped to another graphical language, Danos' and Regnier's **Pure Proof-Nets**, being able to capture *untyped* λ -calculus. It is possible to endow Pure Proof-Nets with an operational semantics¹ which makes them strongly bisimilar to λj . The quotient induced by the translation of λj -terms to Pure Proof-Nets is given by the **graphical equivalence** \equiv_{\circ} given by (the contextual and reflexive-transitive closure of) the equations in Figure 2.

$$\begin{array}{llll} t[x/s][y/v] & \sim_{\text{CS}} & t[y/v][x/s] & \text{if } x \notin \text{fv}(v) \ \& \ y \notin \text{fv}(s) \\ \lambda y.(t[x/s]) & \sim_{\sigma_1} & (\lambda y.t)[x/s] & \text{if } y \notin \text{fv}(s) \\ t[x/s] v & \sim_{\sigma_2} & (t v)[x/s] & \text{if } x \notin \text{fv}(v) \end{array}$$

Figure 2: The graphical equivalence \equiv_{\circ} .

This means that Pure Proof-Nets quotient more than λj -dags². As before, \equiv_{\circ} is a strong bisimulation (cf. Lemma 5.2), and thus confluence and PSN automatically lifts to $\rightarrow_{\lambda j/\circ}$ (cf. Theorem 5.3), which is the reduction relation λj modulo \equiv_{\circ} .

The equations defining \equiv_{\circ} permute a jump with a linear constructor. It is then natural to wonder if \equiv_{\circ} can be extended with additional equations for the missing permutations, still having confluence and PSN. The answer is *yes* and the obtained extension is called the **substitution equivalence** \equiv_{obox} . The fact that λj modulo \equiv_{obox} enjoys PSN is the main result of this paper.

Indeed, such an extension is very delicate. First, we explain how a naïve extension of \equiv_{\circ} breaks PSN. Then we define the equivalence \equiv_{obox} which, unfortunately, is not a strong bisimulation. As a consequence, there is no easy way of getting rid of \equiv_{obox} : erasures cannot be postponed and there is no canonical representant of \equiv_{obox} -classes which is stable by reduction.

Section 5.1 starts over by explaining the equivalence \equiv_{\circ} in terms of Regnier's σ -equivalence [33], thus providing a different point of view with respect to what was already told. Subsection 5.2 discusses how to extend \equiv_{\circ} to \equiv_{obox} by showing the difficulties to prove PSN for the obtained extension.

¹Danos' and Regnier's original operational semantics does not match exactly λj because they use a big-steps rule for eliminating exponential cuts, which corresponds to use just one substitution rule $t[x/u] \rightarrow t\{x/u\}$. However, the refinement of Pure Proof-Nets where duplications are done small-steps is very natural from an explicit substitution point of view, although - to our knowledge - it has never been considered before.

² λj -dags can be mapped on Pure Proof-Nets, and once again the map is a strong bisimulation.

Section 6 develops the proof of PSN for λj modulo \equiv_{obox} .

5.1. **The graphical equivalence. Regnier's equivalence** $\equiv_{\hat{\sigma}}$ is the smallest equivalence on λ -terms closed by contexts and containing the equations in Figure 3:

$$\begin{array}{ll} (\lambda x. \lambda y. t) u \sim_{\hat{\sigma}_1} \lambda y. ((\lambda x. t) u) & \text{if } y \notin \text{fv}(u) \\ (\lambda x. t v) u \sim_{\hat{\sigma}_2} (\lambda x. t) u v & \text{if } x \notin \text{fv}(v) \end{array}$$

Figure 3: The equivalence relation $\hat{\sigma}$

Regnier proved that two $\hat{\sigma}$ -equivalent terms have essentially the same *operational* behavior: $\equiv_{\hat{\sigma}}$ is contained in the equational theory generated by β -reduction, *i.e.* $\equiv_{\hat{\sigma}} \subset \equiv_{\beta}$, and if $t \equiv_{\hat{\sigma}} t'$ then the maximal β -reduction sequences from t and t' have the same length (the so-called *Barendregt's norm*). That is why he calls $\equiv_{\hat{\sigma}}$ an *operational equivalence*.

It is then natural to expect that the previous property can be *locally* reformulated in terms of a strong bisimulation, namely,

$$\begin{array}{ccc} t & \xrightarrow{\beta} & u \\ \equiv_{\hat{\sigma}} & & \\ t' & & \end{array} \quad \text{implies} \quad \begin{array}{ccc} t & \xrightarrow{\beta} & u \\ \equiv_{\hat{\sigma}} & & \equiv_{\hat{\sigma}} \\ t' & \xrightarrow{\beta} & u' \end{array}$$

Unfortunately, this is not the case. Consider the following example, where grey boxes are used to help the identification of redexes and their reductions.

$$\begin{array}{ccc} t = \lambda y. ((\lambda x. y) z_1) z_2 & \xrightarrow{\beta} & (\lambda x. z_2) z_1 = u \\ \equiv_{\hat{\sigma}_1} & & \not\equiv_{\hat{\sigma}} \end{array}$$

$$t' = ((\lambda x. (\lambda y. y)) z_1) z_2 \xrightarrow{\beta} (\lambda y. y) z_2 = u'$$

The term t' has only one redex whose reduction gives u' which is not $\equiv_{\hat{\sigma}}$ -equivalent to u , the reduct of t . The diagram can be completed only by unfolding the whole reduction:

$$\begin{array}{ccc} t = \lambda y. ((\lambda x. y) z_1) z_2 & \xrightarrow{\beta} & (\lambda x. z_2) z_1 \xrightarrow{\beta} z_2 \\ \equiv_{\hat{\sigma}_1} & & = (\subseteq \equiv_{\hat{\sigma}}) \end{array}$$

$$t' = ((\lambda x. (\lambda y. y)) z_1) z_2 \xrightarrow{\beta} (\lambda y. y) z_2 \xrightarrow{\beta} z_2$$

Note that the second step from t' reduces a *created* redex.

We are now going to analyze $\equiv_{\hat{\sigma}}$ in the framework of λj . For that, Regnier's equivalence can be understood on λj -terms by first removing the dB-redexes. Indeed, let us take the clauses defining $\equiv_{\hat{\sigma}}$

and let us make a dB -reduction step on both sides, thus eliminating the multiplicative redexes as in Regnier's definition.

$$\begin{array}{ccc}
(\lambda x. \lambda y. t) u & \sim_{\hat{\sigma}_1} & \lambda y. ((\lambda x. t) u) & & ((\lambda x. t) u) v & \sim_{\hat{\sigma}_2} & (\lambda x. (t v)) u \\
\downarrow_{\text{dB}} & & \downarrow_{\text{dB}} & & \downarrow_{\text{dB}} & & \downarrow_{\text{dB}} \\
(\lambda y. t)[x/u] & & \lambda y. (t[x/u]) & & (t[x/u]) v & & (t v)[x/u]
\end{array}$$

Now, $\equiv_{\hat{\sigma}}$ can be seen as a change of the positions of jumps in a given term and particularly as a permutation equivalence of jumps concerning the linear constructors of the calculus.

This is not so surprising since such permutations turn into simple equalities when one extends the standard translation of λ -calculus into Linear Logic Proof-Nets to λj -terms (see for example [21]). Another interesting observation is the relationship existing between $\equiv_{\hat{\sigma}}$ and the equivalence \equiv_{CS} introduced in Section 3.2. To understand this point we proceed the other way around by expanding jumps into β -redexes:

$$\begin{array}{ccc}
t[y/v][x/u] & \equiv_{\text{CS}} & t[x/u][y/v] \\
\uparrow_{\text{dB}} & & \uparrow_{\text{dB}} \\
((\lambda y. t) v)[x/u] & & (\lambda y. t[x/u]) v \\
\uparrow_{\text{dB}} & & \uparrow_{\text{dB}} \\
(\lambda x. ((\lambda y. t) v)) u & & (\lambda y. ((\lambda x. t) u)) v
\end{array}$$

It is interesting to notice that the relation between the resulting terms is contained in $\equiv_{\hat{\sigma}}$, that is why it was not visible in λ -calculus:

$$(\lambda x. ((\lambda y. t) v)) u \sim_{\hat{\sigma}_2} (\lambda x. \lambda y. t) u v \sim_{\hat{\sigma}_1} (\lambda y. ((\lambda x. t) u)) v$$

In [1] it has been proved that two λj -terms t and t' are translated to the same Pure Proof-Net if and only if $t \equiv_{\sigma, \text{CS}} t'$. More precisely, this relation can be given by the **graphical equivalence** \equiv_{\circ} already defined in Figure 2.

The equations defining \equiv_{\circ} are given by *local* permutations, but it is interesting to notice that it is also possible to define \equiv_{\circ} in terms of global permutations. First define a **spine context** $S[\]$ as:

$$S ::= \lambda x. S \mid St \mid S[x/t]$$

and then define \equiv_{\circ} as the context closure of the following equation \sim_{\circ} :

$$S[t[x/u]] \sim_{\circ} S[t][x/u] \quad \text{if } \text{bs}(S) \cap \text{fv}(u) = \emptyset \text{ and } |t|_x = |S[t]|_x$$

The two definitions are easily seen to be equivalent. We shall now prove that \equiv_{\circ} is a strong bisimulation, which will immediately imply (Lemma 2.1) that \equiv_{\circ} preserves reduction lengths. This

property is stronger than the one proved by Regnier for \equiv_{δ} , since it holds for any reduction sequence, not only for the maximal ones.

Lemma 5.1. *Let \mathbb{E} be the equivalence relation CS or \circ , and $t, t' \in \mathcal{T}$ s.t. $t \equiv_{\mathbb{E}} t'$. Let $u \in \mathcal{T}$. Then:*

- (1) $|t|_x = |t'|_x$.
- (2) For all $S \subseteq \text{pos}_x(t)$ there is $S' \subseteq \text{pos}_x(t')$ s.t. $|S| = |S'|$ and $R_y^{S,x}(t) \equiv_{\mathbb{E}} R_y^{S',x}(t')$.
- (3) $t\{x/u\} \equiv_{\mathbb{E}} t'\{x/u\}$.
- (4) $u\{x/t\} \equiv_{\mathbb{E}} u\{x/t'\}$.

Proof. Straightforward inductions. □

Lemma 5.2. *The relations \equiv_{CS} and \equiv_{\circ} are strong bisimulations for λj .*

Proof. We prove the statement for \equiv_{\circ} . The proof for \equiv_{CS} is obtained by forgetting the cases $\{\sim_{\sigma_1}, \sim_{\sigma_2}\}$. Assume $t_0 \equiv_{\circ} t_1$ holds in n -steps, which is written as $t_0 \equiv_{\circ}^n t_1$, and let $t_1 \rightarrow_{\lambda j} s_1$. We show $\exists s_0$ s.t. $t_0 \rightarrow_{\lambda j} s_0 \equiv_{\circ} s_1$ by induction on n .

The inductive step for $n > 1$ is straightforward. For the base case when $n = 1$ we reason by induction on the definition of $t_0 \equiv_{\circ}^1 t_1$, given by the closure under contexts of the equations $\{\sim_{\text{CS}}, \sim_{\sigma_1}, \sim_{\sigma_2}\}$.

We only show here the cases where $t_0 \equiv_{\circ} t_1$ is contextual, all the other ones being straightforward.

- If $t_0 = t[x/u] \equiv_{\circ} t'[x/u] = t_1 \rightarrow_{\lambda j} t'[x/u'] = s_1$, where $t \equiv_{\circ} t'$ and $u \rightarrow_{\lambda j} u'$, then we close the diagram by $t_0 \rightarrow_{\lambda j} t[x/u'] \equiv_{\circ} s_1$.
- The case $t_0 = t[x/u] \equiv_{\circ} t[x/u'] = t_1 \rightarrow_{\lambda j} t'[x/u'] = s_1$, where $u \equiv_{\circ} u'$ and $t \rightarrow_{\lambda j} t'$ is analogous to the previous one.
- If $t_0 = t[x/u] \equiv_{\circ} t'[x/u] = t_1 \rightarrow_{\lambda j} t''[x/u] = s_1$, where $t \equiv_{\circ} t' \rightarrow_{\lambda j} t''$, then by the i.h. $\exists t'''$ s.t. $t \rightarrow_{\lambda j} t''' \equiv_{\circ} t''$. We close the diagram by $t_0 \rightarrow_{\lambda j} t'''[x/u] \equiv_{\circ}^* s_1$.
- The case $t_0 = t[x/u] \equiv_{\circ} t[x/u'] = t_1 \rightarrow_{\lambda j} t[x/u''] = s_1$, where $u \equiv_{\circ} u' \rightarrow_{\lambda j} u''$ is analogous to the previous one.
- If $t_0 = t[x/u] \equiv_{\circ} t[x/u'] = t_1 \rightarrow_{\mathfrak{w}} t = s_1$, where $u \equiv_{\circ} u'$ and $|t|_x = 0$, then $t_0 \rightarrow_{\mathfrak{w}} t = s_1$.
- If $t_0 = t[x/u] \equiv_{\circ} t'[x/u] = t_1 \rightarrow_{\mathfrak{w}} t = s_1$, where $t \equiv_{\circ} t'$ and $|t|_x = 0$, then the previous remark implies $|t|_x = 0$ and we close the diagram by $t_0 \rightarrow_{\mathfrak{w}} t \equiv_{\circ} t' = s_1$.
- If $t_0 = t[x/u] \equiv_{\circ} t[x/u'] = t_1 \rightarrow_{\mathfrak{c}} t_{[y]_x}[x/u'] [y/u'] = s_1$, where $u \equiv_{\circ} u'$ and $|t|_x > 1$, then we close the diagram by $t_0 \rightarrow_{\mathfrak{c}} t = t_{[y]_x}[x/u][y/u] \equiv_{\circ}^2 t_{[y]_x}[x/u'] [y/u']$.
- If $t_0 = t[x/u] \equiv_{\circ} t'[x/u] = t_1 \rightarrow_{\mathfrak{c}} t'_{[y]_x}[x/u][y/u] = s_1$, where $t \equiv_{\circ} t'$ and $|t|_x > 1$, then we first write $t'_{[y]_x}$ as $R_y^{S',x}(t')$, where $S' \subset \text{pos}_{t'}(x)$ and $|S'| \geq 2$. Lemma 5.1:1 gives $|t|_x > 1$ and Lemma 5.1:2 gives $S \subset \text{pos}_t(x)$ verifying $|S| = |S'|$ and $R_y^{S',x}(t') \equiv_{\circ} R_y^{S,x}(t)$. Then, we close the diagram with $t_0 \rightarrow_{\mathfrak{c}} R_y^{S,x}(t)[x/u][y/u] \equiv_{\circ} t'_{[y]_x}[x/u][y/u]$.
- If $t_0 = t[x/u] \equiv_{\circ} t[x/u'] = t_1 \rightarrow_{\mathfrak{d}} t\{x/u'\} = s_1$, where $u \equiv_{\circ} u'$ and $|t|_x = 1$, then $t[x/u] \rightarrow_{\mathfrak{d}} t\{x/u\} \equiv_{\circ} t\{x/u'\}$, where the last equivalence holds by Lemma 5.1:4.
- If $t_0 = t[x/u] \equiv_{\circ} t'[x/u] = t_1 \rightarrow_{\mathfrak{d}} t'\{x/u\} = s_1$, where $t \equiv_{\circ} t'$ and $|t|_x = 1$. Then, $t[x/u] \rightarrow_{\mathfrak{d}} t\{x/u\} \equiv_{\circ} t'\{x/u\}$ where the last equivalence holds by Lemma 5.1:1-3.

□

A consequence (cf. Lemma 2.2) of the previous lemma is that both \equiv_{CS} and \equiv_{\circ} can be postponed, which implies in particular the following.

Theorem 5.3. *The reduction systems $(\mathcal{T}, \rightarrow_{\lambda j/\text{CS}})$ and $(\mathcal{T}, \rightarrow_{\lambda j/\circ})$ are both confluent and enjoy PSN.*

Proof. Confluence follows from Lemma 5.2 and Theorem 3.15 by application of Lemma 2.2:2, while PSN follows from Lemma 5.2 and Corollary 4.5 by application of Lemma 2.1. \square

Actually, $\rightarrow_{\lambda_j/o}$ is equal to $\equiv_o \rightarrow_{\lambda_j} \equiv_o$. In the framework of rewriting modulo an equivalence relation there are various, non-equivalent, forms of confluence. The one we showed is the weakest one, but the *Church-Rosser modulo* property also holds in our framework:

Theorem 5.4. *Let E be the equivalence relation CS or o . If $t_0 \equiv_E t_1$, $t_0 \rightarrow_{\lambda_j/E}^* u_0$ and $t_1 \rightarrow_{\lambda_j/E}^* u_1$, then $\exists v_0, v_1$ s.t. $u_0 \rightarrow_{\lambda_j/E}^* v_0$, $u_1 \rightarrow_{\lambda_j/E}^* v_1$ and $v_0 \equiv_E v_1$.*

Proof. By Lemma 5.2 and Lemma 2.2. \square

We finish this section with the following interesting property.

Lemma 5.5. *The reduction relation j/o is strongly normalizing.*

Proof. The proof uses the measure $\mathbf{jm}()$ used to prove Lemma 3.8 and the fact that $t \equiv_o t'$ implies $\mathbf{jm}(t) = \mathbf{jm}(t')$. \square

5.2. The substitution equivalence. Composition of explicit substitutions is a sensible topic in the literature, it is interesting to know if λ_j can be extended with a safe notion of (structural) composition.

The structural λ -calculus is peculiar since composition of substitution is provided natively, but only *implicitly* and at distance. Indeed, a term $t[x/u][y/v]$ s.t. $y \in \mathbf{fv}(u)$ & $y \in \mathbf{fv}(t)$ 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 paper is to prove that adding explicit composition to λ_j preserves PSN and confluence.

The second aim of this section concerns *explicit decomposition*. Indeed, some calculi [32, 28, 35, 13, 12] explicitly *decompose* substitutions, *i.e.* reduce $t[x/u[y/v]]$ to $t[x/u][y/v]$. We show that PSN and confluence hold even when extending λ_j with such a rule.

More generally, having a core system, λ_j , whose operational semantics does not depend on propagations, we study how to modularly add propagations by keeping the good properties. We have already shown that λ_j is stable with respect to the graphical equivalence, which can be seen as handling propagations of jumps with respect to *linear* constructors. We proved that λ_j/o is confluent and enjoys PSN (Theorem 5.3). What we investigate here is if we can extend it to propagations with respect to *non-linear* constructors.

The idea is to extend \equiv_o to \equiv_n , where \equiv_n is the contextual and reflexive-transitive closure of the relation generated by $\{\mathbf{CS}, \sigma_1, \sigma_2\}$ plus:

$$\begin{aligned} (t \ v)[x/u] &\sim_{\mathbf{box}_1^0} t \ v[x/u] && \text{if } x \notin \mathbf{fv}(t) \\ t[y/v][x/u] &\sim_{\mathbf{box}_2^0} t[y/v[x/u]] && \text{if } x \notin \mathbf{fv}(t) \end{aligned}$$

In terms of global permutations \equiv_n can be defined as the context closure of $C[[t[x/u]]] \sim_n C[[t]][x/u]$ where $|t|_x = |C[[t]]|_x$, and $C[[\]]$ is *any* context (not just a spine context) which does not capture the variables of u . These equations are constructor preserving (same kind and number of constructors), in contrast to more traditional explicit substitution calculi containing for instance the following rule:

$$(t \ u)[y/v] \rightarrow_{\textcircled{a}} t[y/v] \ u[y/v]$$

which achieves two actions at the same time: duplication and propagation of a jump. In $\lambda\mathbf{j}/\mathbf{n}$ there is a neat separation between propagations and duplications, so that no propagation affects the number of constructors. The rule $\rightarrow_{\textcircled{a}}$ can be simulated in $\lambda\mathbf{j}/\mathbf{n}$ only in the very special case where t and u both have occurrences of y . In our opinion this is not a limitation: the rule $\rightarrow_{\textcircled{a}}$ is particularly inefficient since it duplicates even if there is no occurrence of y at all, thus it is rather a good sign that $\lambda\mathbf{j}/\mathbf{n}$ cannot simulate $\rightarrow_{\textcircled{a}}$.

The reduction relation $\lambda\mathbf{j}/\mathbf{n}$ does not enjoy PSN, since it is a bit naïve on the way it handles void substitutions. The following counter-example has been found by Stefano Guerrini. Let $u = (z z)[z/y]$, then

$$\begin{aligned} t = u[x/u] = (z z)[z/y][x/u] &\equiv_{\text{box}_2^0} (z z)[z/y[x/u]] \rightarrow_c \\ (z_1 z_2)[z_1/y[x/u]][z_2/y[x/u]] &\rightarrow_d^+ y[x/u] (y[x/u]) \equiv_{\sigma_2, \text{box}_1^0, \alpha} \\ (y y)[x_1/u][x/u] &\equiv_{\text{box}_2^0} (y y)[x_1/u[x/u]] \end{aligned}$$

The term t reduces to a term containing t and so there is a loop of the form $t \rightarrow^+ C_0[t] \rightarrow^+ C_0[C_1[t]] \rightarrow^+ \dots$. Now, take $t_0 = (\lambda x.((\lambda z.z z) y)) ((\lambda z.z z) y)$, which is strongly normalizing in the λ -calculus. Since t_0 $\lambda\mathbf{j}/\mathbf{n}$ -reduces to t , t_0 is not $\lambda\mathbf{j}/\mathbf{n}$ -strongly normalizing and thus $\lambda\mathbf{j}/\mathbf{n}$ does not enjoy PSN. It is worth to note that, in contrast to Melliès counterexample for λ_σ , the dB -rule has no role in building the diverging reduction: the fault comes only from the jump subsystem \mathbf{j} modulo $\equiv_{\mathbf{n}}$.

The key point of the previous counter-example is that the jump $[x/u]$ is free to float everywhere in the term since x has no occurrence in t . Such behavior can be avoided by imposing the constraint " $x \in \text{fv}(v)$ " to box_1^0 and box_2^0 . This has also a natural graphical justification in terms of Pure Proof-Nets ([1], Chapter 6, page 149), since it turns box_1^0 and box_2^0 into the exact analogous form of the commutative box-box rule of Linear Logic Proof-Nets, but used here as an equivalence relation. We then modify $\sim_{\text{box}_1^0}$ and $\sim_{\text{box}_2^0}$ by introducing the equivalence \equiv_{box} as the contextual and reflexive-transitive closure of the equations in Figure 4.

$$\begin{aligned} (t v)[x/u] &\sim_{\text{box}_1} t v[x/u] && \text{if } x \notin \text{fv}(t) \ \& \ x \in \text{fv}(v) \\ t[y/v][x/u] &\sim_{\text{box}_2} t[y/v][x/u] && \text{if } x \notin \text{fv}(t) \ \& \ x \in \text{fv}(v) \end{aligned}$$

Figure 4: The equivalence \equiv_{box}

Now, we redefine $\equiv_{\mathbf{n}}$ in the following way. The **substitution equivalence** \equiv_{obox} is the smallest equivalence closed by contexts containing all the equations in Figure 5.

$$\begin{aligned} t[x/s][y/v] &\sim_{\text{CS}} t[y/v][x/s] && \text{if } x \notin \text{fv}(v) \ \& \ y \notin \text{fv}(s) \\ \lambda y.(t[x/s]) &\sim_{\sigma_1} (\lambda y.t)[x/s] && \text{if } y \notin \text{fv}(s) \\ t[x/s] v &\sim_{\sigma_2} (t v)[x/s] && \text{if } x \notin \text{fv}(v) \\ (t v)[x/u] &\sim_{\text{box}_1} t v[x/u] && \text{if } x \notin \text{fv}(t) \ \& \ x \in \text{fv}(v) \\ t[y/v][x/u] &\sim_{\text{box}_2} t[y/v][x/u] && \text{if } x \notin \text{fv}(t) \ \& \ x \in \text{fv}(v) \end{aligned}$$

Figure 5: The substitution equivalence \equiv_{obox}

Alternatively, \equiv_{obox} can be defined by the context closure of the following global permutating equations:

$$\begin{array}{l} C[[t[x/u]]] \sim_{\text{obox}} C[[t][x/u]] \text{ if } \mathbf{bs}(C) \cap \mathbf{fv}(u) = \emptyset \text{ and } |t|_x = |H[[t]]|_x > 0 \\ S[[t[x/u]]] \sim_{\text{obox}} S[[t][x/u]] \text{ if } \mathbf{bs}(S) \cap \mathbf{fv}(u) = \emptyset \text{ and } |t|_x = |S[[t]]|_x = 0 \end{array}$$

where C is any context and S is a spine context.

It is now natural to study λj -reduction modulo \equiv_{obox} . It is easy to prove that the jump calculus terminates with respect to the new equivalence \equiv_{obox} so that the previous counterexample to PSN is ruled out.

Lemma 5.6. *The reduction relation j/obox is terminating.*

Proof. The proof uses the measure $\mathbf{jm}()$ used to prove Lemma 3.8 and the fact that $t \equiv_{\text{obox}} t'$ implies $\mathbf{jm}(t) = \mathbf{jm}(t')$. Notice in particular that $t \equiv_{\text{box}_1, \text{box}_2, \text{box}_1^0} t'$ implies $\mathbf{jm}(t) = \mathbf{jm}(t')$ but $t \equiv_{\text{box}_2^0} t'$ does not imply $\mathbf{jm}(t) = \mathbf{jm}(t')$. \square

However, λj modulo \equiv_{obox} is an incredibly subtle and complex rewriting system, and proving PSN is far from being trivial. Some of the difficulties are:

- *The relation \equiv_{obox} is not a strong bisimulation.* It is not difficult to see that λj is confluent modulo \equiv_{obox} (essentially the same proof than for λj). However, \equiv_{obox} does not preserve reduction lengths to normal form, *i.e.* it is not a strong bisimulation. Two examples can be given by analysing the interaction between \equiv_{obox} with erasure and duplication. Here is an example for erasure:

$$\begin{array}{ccc} z[x/y][y/u] & \rightarrow_w & z[y/u] \\ \equiv_{\text{box}_2} & & \downarrow_w \\ z[x/y[y/u]] & \rightarrow_w & z \end{array}$$

and here another one for duplication:

$$\begin{array}{ccc} (x x)[x/y][y/z] & \rightarrow_c & (x x_1)[x/y][x_1/y][y/z] & \rightarrow_c & (x x_1)[x/y_1][x_1/y_2][y_1/z][y_2/z] \\ \equiv_{\text{box}_2} & & & & \equiv_{\text{obox}} \\ (x x)[x/y[y/z]] & \rightarrow_c & (x x_1)[x/y[y/z]][x_1/y[y/z]] & \equiv_\alpha & (x x_1)[x/y_1[y_1/z]][x_1/y_2[y_2/z]] \end{array}$$

Indeed, if \equiv_{obox} would have been a strong bisimulation, then in both diagrams the two terms of the second column would be \equiv_{obox} -equivalent, while they are not (remark that \equiv_{obox} preserves the number of constructors so that those terms cannot be \equiv_{obox} -equivalent).

- *The relation \equiv_{obox} cannot be postponed.* The last example shows also that \equiv_{obox} cannot be postponed. This is illustrated by the upper left corner of the previous figure:

$$\begin{array}{ccc} (x x)[x/y][y/z] & \rightarrow_c & (x x_1)[x/y][x_1/y][y/z] \\ \equiv_{\text{box}_2} & & \\ (x x)[x/y[y/z]] & & \end{array}$$

Observe that this phenomenon is caused by the equation \sim_{box_2} . Remark that both composition (*i.e.* $\rightarrow_{\text{box}_2}$) and decomposition ($\leftarrow_{\text{box}_2}$) are used in Guerrini's counterexample.

- *There is no canonical representant of equivalence classes which is stable by reduction.* Indeed, there are two natural canonical representants in $\lambda j/\text{obox}$. Given t we can define $\mathbf{in}(t)$ as the term obtained by moving all substitution towards the variables as much as possible and $\mathbf{out}(t)$ the term obtained moving substitutions far from variables as much as possible. Consider

$t = x[x/(\lambda y.z[z/y])x'] \rightarrow_{\text{aB}} x[x/z[z/y][y/x']] = t'$, then $\text{out}(t) = x[x/(\lambda y.z[z/y])x']$ does not reduce to $\text{out}(t') = x[x/z[z/y][y/x']]$. Similarly, for the other representative since $\text{in}(t) = (x[y/z] z)[z/z']$ does not reduce to $\text{in}(t') = x z[z/z']$.

Next section proves that $\lambda\text{j}/\text{obox}$ enjoys PSN. Partial results were already proved [3] in this direction, *i.e.* PSN in the cases where \sim_{box_1} and \sim_{box_2} are both oriented from left to right or from right to left. Surprisingly, the proof of the more general result we present here is relatively more compact and concise than the one(s) in [3]. Indeed, even if we need to pass through an auxiliary calculus, such calculus can be proved to enjoy PSN without using labels, in contrast to our previous result and proof. In our opinion the intermediate calculus we use here is interesting on its own as a tool for proving termination results.

6. PRESERVATION OF β -STRONG NORMALIZATION FOR $\lambda\text{j}/\text{obox}$

Even if there is no canonical representative form of an obox -equivalence class which is stable by reduction (*cf.* Section 5.2), there is an even more natural way to reason about PSN in the presence of the non-trivial equations $\{\text{box}_1, \text{box}_2\}$ which consists in projecting $\lambda\text{j}/\text{obox}$ over a simpler equational calculus. Since both the calculus and the projection are quite peculiar we introduce them gradually.

A usual naïve idea consists in projecting $\lambda\text{j}/\text{obox}$ into the λ -calculus by means of a function computing the complete unfolding of jumps. This gives the following diagram:

$$\begin{array}{ccc} t & \rightarrow_{\lambda\text{j}} & u \\ \downarrow_{\text{j}}^* & & \downarrow_{\text{j}}^* \\ \text{j}(t) & \rightarrow_{\beta}^* & \text{j}(u) \end{array} \quad (6.1)$$

This principle could be easily exploited in order to prove some properties of $\lambda\text{j}/\text{obox}$ (such as confluence), however, since such projection does not preserve all divergent terms, then it cannot be used to show PSN. For instance, let consider $t = x[y/\Omega]$ (where Ω is a non-terminating term), which is only λj -weakly normalizing, whereas $\text{j}(t) = x$ is in normal form. It is easy to show that projection of terms without void jumps preserves divergence and thus PSN. Unfortunately, erasures cannot be postponed in $\lambda\text{j}/\text{obox}$.

Roughly speaking, projection gives $\text{j}(t) \rightarrow_{\beta}^* \text{j}(u)$ so that there are some (eventually erasing) steps $t \rightarrow_{\lambda\text{j}} u$ s.t. $\text{j}(t) = \text{j}(u)$. It is not really a problem if such (erased) steps are finite, but here there may be infinite sequences of such (erased) steps. It is then quite natural to change the complete unfolding j into a *non-erasing* unfolding wj , which does not project void jumps:

$$\begin{aligned} \text{wj}(x) &:= x \\ \text{wj}(\lambda x.u) &:= \lambda x.\text{wj}(u) \\ \text{wj}(u v) &:= \text{wj}(u) \text{wj}(v) \\ \text{wj}(t[x/u]) &:= \begin{cases} \text{wj}(t)\{x/\text{wj}(u)\} & \text{if } x \in \text{fv}(t) \\ \text{wj}(t)[x/\text{wj}(u)] & \text{if } x \notin \text{fv}(t) \end{cases} \end{aligned} \quad (6.2)$$

Notice that there are still some erased steps, as for instance $t = x[x/y] \rightarrow_{\text{w}} y = u$, where $\text{wj}(t) = y = \text{wj}(u)$, but intuition tells that wj preserves divergence, because diverging terms are no longer erased by the projection. Notice also that the image of the projection of the previous reduction step $t \rightarrow_{\text{w}} u$ is no longer a reduction step in the λ -calculus, so that we need to specify which are the rewriting rules and the equations of the image of the translation.

For didactic purpose let us assume that we are able to turn the image of the projection into a calculus - let us say $\lambda\text{void}/o$ - such that \mathbf{wj} projects $\lambda\mathbf{j}/o\text{box}$ into $\lambda\text{void}/o$ and preserves divergence. Two important remarks are: since $\mathbf{wj}(\cdot)$ preserves divergence, then PSN for $\lambda\text{void}/o$ implies PSN for $\lambda\mathbf{j}/o\text{box}$; also, the $\lambda\text{void}/o$ -calculus does not contain the equations $\{\text{box}_1, \text{box}_2\}$ because they were turned into equalities thanks to their side conditions.

It is then reasonable to expect that proving PSN for $\lambda\text{void}/o$ is easier than PSN for $\lambda\mathbf{j}/o\text{box}$. Our proof technique can then be summarised as follows:

- (1) Introduce $\lambda\text{void}/o$;
- (2) Prove PSN for $\lambda\text{void}/o$;
- (3) Show that $\mathbf{wj}(\cdot)$ preserves divergence from $\lambda\mathbf{j}/o\text{box}$ to $\lambda\text{void}/o$;
- (4) Conclude PSN for $\lambda\mathbf{j}/o\text{box}$.

Section 6.1 presents the rewriting rules of $\lambda\text{void}/o$, thus completing point 1. Section 6.2 deals with point 2 and Section 6.3 with points 3 and 4.

We believe that the isolation of $\lambda\text{void}/o$ is an important contribution of this paper. Indeed, it is easy to see that $\lambda\text{void}/o$ should contain at least the three following rewriting rules:

$$\begin{array}{llll} (\lambda x.t)\mathbf{L} u & \mapsto_{\beta} & t\{x/u\}\mathbf{L} & \text{if } x \in \text{fv}(t) \\ (\lambda x.t)\mathbf{L} u & \mapsto_{\text{dB}} & t[x/u]\mathbf{L} & \text{if } x \notin \text{fv}(t) \\ t[x/u] & \mapsto_{\mathbf{w}} & t & \text{if } x \notin \text{fv}(t) \end{array}$$

More precisely,

- The reduction step $t = (\lambda x.x) y \rightarrow_{\text{dB}} x[x/y] = u$ projects into $\mathbf{wj}(t) = (\lambda x.x) y \rightarrow_{\beta} y = \mathbf{wj}(u)$.
- The reduction step $t = (\lambda x.z) y \rightarrow_{\text{dB}} z[x/y] = u$ should map to itself, *i.e.* $\mathbf{wj}(t) = (\lambda x.z) y \rightarrow_{\text{dB}} z[x/y] = \mathbf{wj}(u)$.
- The reduction step $t = z[x/y] \rightarrow_{\mathbf{w}} z = u$ should map to itself, *i.e.* $\mathbf{wj}(t) = z[x/y] \rightarrow_{\mathbf{w}} z = \mathbf{wj}(t)$.

However, projecting on such a simple calculus does not still work. There are three phenomena we should take care of:

- (1) **Equations.** As we already told \equiv_{box_1} and \equiv_{box_2} vanish, that is, $t \equiv_{\text{box}_1, \text{box}_2} u$ implies $\mathbf{wj}(t) = \mathbf{wj}(u)$. The graphical equivalence, instead, do not vanish, and must be added to the intermediate calculus, thus getting the reduction relation to be considered modulo \equiv_o .
- (2) **Generalised erasure.** Consider:

$$t = z[x/y_1 y_2][y_1/v_1][y_2/v_2] \rightarrow_{\mathbf{w}} z[y_1/v_1][y_2/v_2] = u$$

where $\mathbf{wj}(t) = z[x/v_1 v_2]$ and $\mathbf{wj}(u) = z[y_1/v_1][y_2/v_2]$. Hence the \mathbf{w} -rule $t[x/s] \rightarrow t$ if $|t|_x = 0$ must be generalised in order to replace the jump $[x/s]$ by many (eventually none) jumps containing subterms of s . We shall then use the following (Hydra like) rule :

$$(h) \quad t[x/u] \rightarrow t[x_1/u_1] \dots [x_n/u_n] \quad \forall i (x_i \text{ fresh } \& u_i < u \& \text{fv}(u_i) \subseteq \text{fv}(u) \& n \geq 0)$$

Where $u_i < u$ means that u_i is a subterm of u . The condition upon free variables is necessary in order to avoid unwanted captures inducing degenerate behavior. Notice that the particular case $n = 0$ gives the \mathbf{w} -rule.

- (3) **Unboxing:** An erasing step $t \rightarrow_{\mathbf{w}} u$ can cause jumps to *move* towards the *root* of the term. Consider:

$$t = (z z[x/y])[y/v] \rightarrow_{\mathbf{w}} (z z)[y/v] = u$$

where $\mathbf{w}j(t) = z \ z[x/v]$ and $\mathbf{w}j(u) = (z \ z)[y/v] =_{\alpha} (z \ z)[x/v]$. Hence, to project this step over $\lambda\mathbf{void}/\mathbf{o}$ we need a rule moving jumps towards the *root* of the term, which could have in principle the general form

$$C[[t[x/u]]] \rightarrow C[[t]][x/u]$$

This rule is the one that shall demand a more involved - but still reasonable - technical development. Indeed, reduction that moves *any* jump towards the root modulo $\equiv_{\mathbf{o}}$ may cause non-termination:

$$\lambda x.x[y/z] \rightarrow (\lambda x.x)[y/z] \equiv_{\mathbf{o}} \lambda x.x[y/z] \rightarrow \dots$$

In order to avoid this problem we restrict the general form of the rule to a certain kind of contexts, those whose hole is contained in at least one *box*, *i.e.* the argument of an application or the argument of a jump.

We now develop a PSN proof for $\lambda\mathbf{j}/\mathbf{o}\mathbf{box}$. Section 6.1 formally defines the intermediate calculus $\lambda\mathbf{void}/\mathbf{o}$, while Section 6.2 proves PSN for the intermediate calculus $\lambda\mathbf{void}/\mathbf{o}$ and Section 6.3 proves the properties of the projection which allows us to conclude PSN for $\lambda\mathbf{j}/\mathbf{o}\mathbf{box}$.

Let us conclude this section by observing that the generalised erasure and the unboxing rules are introduced to project the \mathbf{w} -rule and not the equations $\{\mathbf{box}_1, \mathbf{box}_2\}$. Said in other words, to prove PSN of the simpler calculus $\lambda\mathbf{j}$ (resp. $\lambda\mathbf{j}/\mathbf{o}$) through the $\mathbf{w}j$ projection into $\lambda\mathbf{void}$ (resp. $\lambda\mathbf{void}/\mathbf{o}$), one still needs the generalised erasure and the unboxing rules. That is why we believe that the technique developed here is really interesting by itself.

6.1. The $\lambda\mathbf{void}/\mathbf{o}$ -calculus. The $\lambda\mathbf{void}/\mathbf{o}$ -calculus can be understood as a memory calculus based on *void* jumps. It is given by a set of terms, written $\mathcal{T}_{\mathbf{v}}$, generated by the following grammar, where only void jumps are allowed:

$$(\mathcal{T}_{\mathbf{v}}) \quad t, u ::= x \mid \lambda x.t \mid t \ u \mid t[-/u]$$

The notation $t[-/u]$ just means that the constant $-$ has no (free) occurrence in the term t and $\overline{[-/s]}$ denotes a list of void jumps $[-/s_1] \dots [-/s_n]$.

To define the operational semantics we need to define a particular kind of context. More precisely, if C denotes a context then a **boxed context** B is given by the following grammar:

$$B ::= t \ C \mid t[-/C] \mid B \ t \mid B[-/t] \mid \lambda y.B$$

We now consider the reduction rules and equations in Figure 6.

$$\begin{array}{lll} (\lambda x.t)\mathbf{L} \ u & \mapsto_{\beta} & t\{x/u\}\mathbf{L} & \text{if } x \in \mathbf{fv}(t) \\ (\lambda x.t)\mathbf{L} \ u & \mapsto_{\mathbf{ab}} & t[-/u]\mathbf{L} & \text{if } x \notin \mathbf{fv}(t) \\ t[-/u] & \mapsto_{\mathbf{h}} & t[-/u_1] \dots [-/u_n] & \forall i (u_i < u \ \& \ \mathbf{fv}(u_i) \subseteq \mathbf{fv}(u) \ \& \ n \geq 0) \\ B[[t[-/u]]] & \mapsto_{\mathbf{u}} & B[[t]][-/u] & B \text{ does not bind } u \\ \\ t[-/s][-/v] & \sim_{\mathbf{cs}} & t[-/v][-/s] & \\ \lambda y.(t[-/s]) & \sim_{\sigma_1} & (\lambda y.t)[-/s] & \text{if } y \notin \mathbf{fv}(s) \\ t[-/s] \ v & \sim_{\sigma_2} & (t \ v)[-/s] & \end{array}$$

Figure 6: The $\lambda\mathbf{void}/\mathbf{o}$ -reduction system

Notice that the w -rule $t[x/u] \rightarrow t$ with $x \notin \mathbf{fv}(t)$ of λj is a particular case of the h -rule. Remark also that the unboxing rule of $\lambda \mathbf{void}/o$ moves \mathbf{void} jumps outside terms, which was forbidden in the equation \mathbf{box}_2 of $\lambda j/o\mathbf{box}$. However, this does not break PSN because there is no *boxing* rule in $\lambda \mathbf{void}/o$. Indeed, Guerrini's counterexample uses both boxing and unboxing.

We write $t \rightarrow_{\lambda \mathbf{void}/o} t'$ iff $t \equiv_o t_1 \rightarrow_{\lambda \mathbf{void}} t'_1 \equiv_o t'$, where $\rightarrow_{\lambda \mathbf{void}}$ is the reduction relation generated by the previous rewriting rules $\{\beta, \mathbf{dB}, \mathbf{h}, \mathbf{u}\}$ and \equiv_o is the equivalence relation defined in Figure 2 but restricted here to the $\lambda \mathbf{void}$ -syntax. As before, $\rightarrow_{\mathcal{R}}$ denotes the contextual closure of $\mapsto_{\mathcal{R}}$, for $\mathcal{R} \subseteq \{\beta, \mathbf{dB}, \mathbf{h}, \mathbf{u}\}$.

We now show some properties of the new memory reduction system which are used in Section 6.2 to show PSN.

Lemma 6.1. *Let $u, v, s \in \mathcal{T}_v$. If $u < s$ and $x \notin \mathbf{fv}(u)$, then $u < s\{x/v\}$.*

Proof. By induction on s . □

Lemma 6.2. *Let $t_0, t_1, u_0, u_1 \in \mathcal{T}_v$.*

- *If $t_0 \rightarrow_{\mathbf{h}, \mathbf{u}/o}^* t_1$ then $t_0\{x/u_0\} \rightarrow_{\mathbf{h}, \mathbf{u}/o}^* t_1\{x/u_0\}$.*
- *If $u_0 \rightarrow_{\mathbf{h}, \mathbf{u}/o}^* u_1$ then $t\{x/u_0\} \rightarrow_{\mathbf{h}, \mathbf{u}/o}^* t\{x/u_1\}$.*

Proof. Straightforward. □

Lemma 6.3. *Let $t, v, u, s_i \in \mathcal{T}_v$. Let $y \notin \mathbf{fv}(t)$ and $x \in \mathbf{fv}(v)$. Then $t[-/v\{x/u\}] \rightarrow_{\mathbf{h}} t[-/\overline{s}][-/u]$, where $s_i < v$ and $\mathbf{fv}(s_i) \subseteq \mathbf{fv}(v)$ and $x \notin \mathbf{fv}(s_i)$.*

Proof. Straightforward. □

Lemma 6.4. *Let $t, u, v \in \mathcal{T}_v$. If $y \in \mathbf{fv}(t)$ then $t\{y/v[-/u]\} \rightarrow_{\mathbf{h}, \mathbf{u}/o}^* t\{y/v\}[-/u]$.*

Proof. By induction on t . □

Lemma 6.5. *Let $t_0, t_1, v \in \mathcal{T}_v$. If $t_0 \rightarrow_{\mathbf{h}}^+ t_1$, $x \in \mathbf{fv}(t_0)$ and $x \notin \mathbf{fv}(t_1)$, then $t_0\{x/v\} \rightarrow_{\mathbf{h}, \mathbf{u}/o}^+ t_1[-/v]$.*

Proof. By induction on the number of steps from t_0 to t_1 , and in the base case by induction on the reduction step from t_0 to t_1 .

- $t_0 = u_0[-/u_1] \rightarrow_{\mathbf{h}} u_0[-/v_1] \dots [-/v_m] = t_1$, where $v_i < u_1$ and $\mathbf{fv}(v_i) \subseteq \mathbf{fv}(u_1)$. Then $x \in u_1$ and $x \notin u_0$ and $x \notin v_i$ so that

$$u_0[-/u_1]\{x/v\} = u_0[-/u_1\{x/v\}] \rightarrow_{\mathbf{h}} (L. 6.3) u_0[-/v_1] \dots [-/v_m][-/v]$$

- $t_0 = \lambda y.u_0 \rightarrow \lambda y.u_1 = t_1$, where $u_0 \rightarrow u_1$. Then,

$$(\lambda y.u_0)\{x/v\} = \lambda y.u_0\{x/v\} \rightarrow_{\mathbf{h}, \mathbf{u}/o}^+ (i.h.) \lambda y.u_1[-/v] \equiv_{\sigma_1} (\lambda y.u_1)[-/v]$$

- $t_0 = u_0 v_0 \rightarrow u_1 v_0 = t_1$, where $u_0 \rightarrow u_1$. Then,

$$(u_0 v_0)\{x/v\} = u_0\{x/v\} v_0 \rightarrow_{\mathbf{h}, \mathbf{u}/o}^+ (i.h.) u_1[-/v] v_0 \equiv_{\sigma_2} (u_1 v_0)[-/v]$$

- $t_0 = u_0[-/v_0] \rightarrow u_1[-/v_0] = t_1$, where $u_0 \rightarrow u_1$. Then:

$$u_0[-/v_0]\{x/v\} = u_0\{x/v\}[-/v_0] \rightarrow_{\mathbf{h}, \mathbf{u}/o}^+ (i.h.) u_1[-/v][-/v_0] \equiv_{\mathbf{cs}} u_1[-/v_0][-/v]$$

- All the remaining cases are straightforward. □

Corollary 6.6. *Let $t_0, t_1, v \in \mathcal{T}_v$. If $t_0 \rightarrow_{\mathbf{h}, \mathbf{u}/\circ}^+ t_1$, $x \in \mathbf{fv}(t_0)$ and $x \notin \mathbf{fv}(t_1)$, then $t_0\{x/v\} \rightarrow_{\mathbf{h}, \mathbf{u}/\circ}^+ t_1[-/v]$.*

Proof. By induction on the number of \mathbf{h} -steps in the reduction $t_0 \rightarrow_{\mathbf{h}, \mathbf{u}/\circ}^+ t_1$. Notice first that $\rightarrow_{\mathbf{u}}$ and \equiv_{\circ} do not loose free variables.

- If there is only one \mathbf{h} -step, then the reduction is of the form $t \rightarrow_{\mathbf{u}/\circ}^* u_0 \rightarrow_{\mathbf{h}} u_1 \rightarrow_{\mathbf{u}/\circ}^* t'$. We have $t\{x/v\} \rightarrow_{\mathbf{u}/\circ}^* u_0\{x/v\} \rightarrow_{\mathbf{h}, \mathbf{u}/\circ}^+ (L. 6.5) u_1[-/v] \rightarrow_{\mathbf{u}/\circ}^* t'[-/v]$.
- If there are $n > 0$ \mathbf{h} -steps, then the reduction is of the form $t_0 \rightarrow_{\mathbf{u}/\circ}^* u_0 \rightarrow_{\mathbf{h}} u_1 \rightarrow_{\mathbf{h}, \mathbf{u}/\circ}^+ t_1$, with $n - 1 < n$ \mathbf{h} -steps from u_1 to t_1 we consider two cases.
 - If $x \in \mathbf{fv}(u_0) \ \& \ x \in \mathbf{fv}(u_1)$, then x is lost in the subsequence $u_1 \rightarrow_{\mathbf{h}, \mathbf{u}/\circ}^+ t_1$. We thus have $t_0\{x/v\} \rightarrow_{\mathbf{u}/\circ}^* u_0\{x/v\} \rightarrow_{\mathbf{h}} u_1\{x/v\} \rightarrow_{\mathbf{h}, \mathbf{u}/\circ}^+ (i.h.) t_1[-/v]$.
 - If $x \in \mathbf{fv}(u_0) \ \& \ x \notin \mathbf{fv}(u_1)$, then $t_0\{x/v\} \rightarrow_{\mathbf{u}/\circ}^* u_0\{x/v\} \rightarrow_{\mathbf{h}, \mathbf{u}/\circ}^+ u_1[-/v] (L. 6.5) \rightarrow_{\mathbf{h}, \mathbf{u}/\circ}^+ t_1[-/v]$. □

6.2. Preservation of β -Strong Normalization for $\lambda\mathbf{void}/\circ$. The proof of PSN for $\lambda\mathbf{void}/\circ$ we are going to develop in this section is based on the **IE** property (*cf.* Section 4) and follows the main lines of that of Theorem 4.3. Indeed, given $u \in \mathcal{SN}_{\lambda\mathbf{void}/\circ}$ and $t\{x/u\} \bar{v}_n^1 \in \mathcal{SN}_{\lambda\mathbf{void}}$ we show that $M = t[-/u] \bar{v}_n^1 \in \mathcal{SN}_{\lambda\mathbf{void}}$ by using a measure on terms which decreases for every one-step $\lambda\mathbf{void}/\circ$ -reduct of M . However, PSN for $\lambda\mathbf{void}/\circ$ is much more involved: first because of the nature of the reduction rules $\{\mathbf{h}, \mathbf{u}\}$, second because of the equivalence \equiv_{\circ} .

A first remark is that jumps in $\lambda\mathbf{void}/\circ$ are all void so in particular they cannot be duplicated. As a consequence, there is no need at first sight to generalise the **IE** property to terms of the form $t[x_i/u_i]_m^1 \bar{v}_n^1$ as we did before (Theorem 4.3). However, there are now new ways of getting jumps *on the surface* of the term. Indeed, if $t = \lambda y.t'[-/v]$ and $y \notin \mathbf{fv}(v)$ one has

$$M = t[-/u] \bar{v}_n^1 \equiv_{\circ} (\lambda y.t'[-/v])[-/u] \bar{v}_n^1$$

Things are even more complicated since jumps can also be moved between the arguments v_1, \dots, v_n as in:

$$M \equiv_{\circ} ((\lambda y.t'[-/v]) v_1)[-/u] \bar{v}_n^2$$

The opposite phenomenon can happen too, *i.e.* the jump $[-/u]$ can enter inside t , for instance:

$$M \equiv_{\circ} \lambda y.(t'[-/v])[-/u] \bar{v}_n^1$$

The main point is that the measure we shall use to develop the proof of the **IE** property needs to be stable by the equivalence \equiv_{\circ} , *i.e.* if $M \equiv_{\circ} M'$, then M and M' must have the same measure.

In order to handle this phenomenon we are going to split M in two parts: the multiset $\mathbb{S}\mathbb{J}(M)$ of **jumps** of M which *are* or *can* get to the **surface**, and the **trunk** $\mathbb{T}(M)$, *i.e.* the term obtained from M by removing all the jumps in $\mathbb{S}\mathbb{J}(M)$. This *splitting* of the term is then used to generalise the statement of the **IE** as follows:

If $\mathbb{T}(M) \in \mathcal{SN}_{\lambda\mathbf{void}/\circ}$ and $u \in \mathcal{SN}_{\lambda\mathbf{void}/\circ}$ for every $[-/u] \in \mathbb{S}\mathbb{J}(M)$ then $M \in \mathcal{SN}_{\lambda\mathbf{void}/\circ}$.

An intuition behind the scheme of this proof is that the term $\mathbb{T}(M)$ and the jumps in $\mathbb{S}\mathbb{J}(M)$ are dynamically independent, in the sense that any reduction of M can be seen as an interleaving of a reduction (eventually empty) of $\mathbb{T}(M)$ and reductions (eventually empty) of elements of $\mathbb{S}\mathbb{J}(M)$. Indeed, the void jumps in $\mathbb{S}\mathbb{J}(M)$ *cannot be affected* by a reduction of $\mathbb{T}(M)$, since none of their free variables is bound in M , and *cannot affect* a reduction of $\mathbb{T}(M)$ since they are void. The unboxing

rule slightly complicates things, but morally that is why the new generalised form of the **IE** property holds.

The attentive reader may wonder why we cannot handle the equivalence \equiv_{\circ} by using a strong bisimulation argument, as in the case of $\lambda j/o$ (cf. Theorem 5.3). Unfortunately, \equiv_{\circ} is not a strong bisimulation for λvoid as the following example shows:

$$\begin{array}{ccc} x[-/t[-/x] v] & \rightarrow_{\mathbf{h}} & x[-/t[-/x]] \\ \equiv_{\circ} & & \equiv_{\circ} \\ x[-/(t v)[-/x]] & \rightarrow_{\mathbf{h}} & ? \end{array}$$

Before starting with the technical details of the proof let us add two more important remarks. First, we have just used $\mathbb{T}(M)$ and $\mathbb{S}\mathbb{J}(M)$ for didactic purposes, the actual definitions are parametrised with respect to a set of variables (those which can be captured in the context containing M). Moreover, in order to simplify the proofs we will not work directly with $\mathbb{S}\mathbb{J}(M)$: we are going to define a parametrised predicate $\text{SNS}\mathbb{J}_{\Gamma}(t)$, which is true when all the jumps in $\mathbb{S}\mathbb{J}(M)$ are in $\mathcal{SN}_{\lambda\text{void}/o}$, and a parametrised measure $\text{MS}\mathbb{J}_{\Gamma}(x)$, built out from the elements of $\mathbb{S}\mathbb{J}(M)$. Second, the unboxing rule makes some inductive reasonings non-trivial, so we isolate them in an intermediate lemma (Lemma. 6.11).

Given $t \in \mathcal{T}_{\mathbf{v}}$ and a set of variables Γ s.t. $\mathbf{fv}(t) \subseteq \Gamma$, the trunk $\mathbb{T}_{\Gamma}(t)$ is given by the following inductive definition.

$$\begin{array}{ll} \mathbb{T}_{\Gamma}(x) & := x \\ \mathbb{T}_{\Gamma}(tu) & := \mathbb{T}_{\Gamma}(t)u \\ \mathbb{T}_{\Gamma}(\lambda x.t) & := \lambda x.\mathbb{T}_{\Gamma \cup \{x\}}(t) \\ \mathbb{T}_{\Gamma}(t[-/u]) & := \mathbb{T}_{\Gamma}(t) \quad \text{if } \mathbf{fv}(u) \cap \Gamma = \emptyset \\ \mathbb{T}_{\Gamma}(t[-/u]) & := \mathbb{T}_{\Gamma}(t)[-/u] \quad \text{otherwise} \end{array}$$

Next, we define a predicate on $\mathcal{T}_{\mathbf{v}}$ which is true when all surface jumps contain terminating terms:

$$\begin{array}{ll} \text{SNS}\mathbb{J}_{\Gamma}(x) & := \text{true} \\ \text{SNS}\mathbb{J}_{\Gamma}(tu) & := \text{SNS}\mathbb{J}_{\Gamma}(t) \\ \text{SNS}\mathbb{J}_{\Gamma}(\lambda x.t) & := \text{SNS}\mathbb{J}_{\Gamma \cup \{x\}}(t) \\ \text{SNS}\mathbb{J}_{\Gamma}(t[-/u]) & := \text{SNS}\mathbb{J}_{\Gamma}(t) \ \& \ u \in \mathcal{SN}_{\lambda\text{void}/o} \quad \text{if } \mathbf{fv}(u) \cap \Gamma = \emptyset \\ \text{SNS}\mathbb{J}_{\Gamma}(t[-/u]) & := \text{SNS}\mathbb{J}_{\Gamma}(t) \quad \text{otherwise} \end{array}$$

Remark that $t \in \mathcal{SN}_{\lambda\text{void}/o}$ implies in particular $\text{SNS}\mathbb{J}_{\Gamma}(t)$ for any set Γ .

For any term $t \in \mathcal{T}_{\mathbf{v}}$ s.t. $\text{SNS}\mathbb{J}_{\Gamma}(t)$ we define the following multiset measure:

$$\begin{array}{ll} \text{MS}\mathbb{J}_{\Gamma}(x) & := \emptyset \\ \text{MS}\mathbb{J}_{\Gamma}(tu) & := \text{MS}\mathbb{J}_{\Gamma}(t) \sqcup \text{MS}\mathbb{J}_{\Gamma}(u) \\ \text{MS}\mathbb{J}_{\Gamma}(\lambda x.t) & := \text{MS}\mathbb{J}_{\Gamma \cup \{x\}}(t) \\ \text{MS}\mathbb{J}_{\Gamma}(t[-/u]) & := \text{MS}\mathbb{J}_{\Gamma}(t) \sqcup \langle \eta_{\lambda\text{void}/o}(u), |u| \rangle \quad \text{if } \mathbf{fv}(u) \cap \Gamma = \emptyset \\ \text{MS}\mathbb{J}_{\Gamma}(t[-/u]) & := \text{MS}\mathbb{J}_{\Gamma}(t) \sqcup \text{MS}\mathbb{J}_{\Gamma}(u) \quad \text{otherwise} \end{array}$$

Now, we can reformulate a generalised statement for the **IE** property on **Void** jumps as follows:

(VIE) For all $t \in \mathcal{T}_{\mathbf{v}}$, if $\mathbb{T}_{\emptyset}(t) \in \mathcal{SN}_{\lambda\text{void}/o}$ and $\text{SNS}\mathbb{J}_{\emptyset}(t)$, then $t \in \mathcal{SN}_{\lambda\text{void}/o}$.

Some lemmas about basic properties of $\mathbb{T}_{\Gamma}(t)$, $\text{SNS}\mathbb{J}_{\Gamma}(t)$ and $\text{MS}\mathbb{J}_{\Gamma}(t)$ follow.

Lemma 6.7. *Let $t \in \mathcal{T}_{\mathbf{v}}$ and $x \notin \mathbf{fv}(t)$. Then $\mathbb{T}_{\Gamma \cup \{x\}}(t) = \mathbb{T}_{\Gamma}(t)$ and $\text{SNS}\mathbb{J}_{\Gamma \cup \{x\}}(t)$ iff $\text{SNS}\mathbb{J}_{\Gamma}(t)$.*

Proof. Straightforward. □

Lemma 6.8. *Let $t, u \in \mathcal{T}_{\mathbf{v}}$ s.t. $\mathbf{fv}(t) \subseteq \Gamma$. Then,*

- (1) $t \rightarrow_{\mathbf{h}}^* \mathbb{T}_{\Gamma}(t)$.
- (2) If $x \notin \Gamma$ then $\mathbb{T}_{\Gamma \cup \{x\}}(t)\{x/u\} \rightarrow_{\mathbf{h}}^* \mathbb{T}_{\Gamma}(t\{x/u\})$.

Proof.

- (1) Straightforward by induction on t
- (2) By induction on t .

- $t = x$: Then $\mathbb{T}_{\Gamma \cup \{x\}}(t)\{x/u\} = u$ and $\mathbb{T}_{\Gamma}(t\{x/u\}) = \mathbb{T}_{\Gamma}(u)$. We conclude using Point 1.
- $t = y$: Then $\mathbb{T}_{\Gamma \cup \{x\}}(t)\{x/u\} = y = \mathbb{T}_{\Gamma}(t\{x/u\})$.
- The cases $t = \lambda y.v$ and $t = uv$ are straightforward using the i.h.
- $t = v[-/w]$: Let us analyse one particular case in detail, the other ones being similar can be proved by application of the definitions and the i.h. Let suppose $\Gamma \cap \mathbf{fv}(w) = \emptyset$, $x \in \mathbf{fv}(w)$ and $\Gamma \cap \mathbf{fv}(u) = \emptyset$. Then $\mathbb{T}_{\Gamma \cup \{x\}}(t)\{x/u\} = \mathbb{T}_{\Gamma \cup \{x\}}(v)[-/w]\{x/u\} = \mathbb{T}_{\Gamma \cup \{x\}}(v)\{x/u\}[-/w\{x/u\}]$ and $\mathbb{T}_{\Gamma}(t\{x/u\}) = \mathbb{T}_{\Gamma}(v\{x/u\}[-/w\{x/u\}]) = \mathbb{T}_{\Gamma}(v\{x/u\})$. The i.h. gives $\mathbb{T}_{\Gamma \cup \{x\}}(v)\{x/u\} \rightarrow_{\mathbf{h}}^* \mathbb{T}_{\Gamma}(v\{x/u\})$ and so we conclude with

$$\mathbb{T}_{\Gamma \cup \{x\}}(v)\{x/u\}[-/w\{x/u\}] \rightarrow_{\mathbf{h}}^* \mathbb{T}_{\Gamma}(v\{x/u\})[-/w\{x/u\}] \rightarrow_{\mathbf{h}} \mathbb{T}_{\Gamma}(v\{x/u\})$$

□

Lemma 6.9. *Let $t, u \in \mathcal{T}_{\mathbf{v}}$ and $x \notin \Gamma$. If $\text{SNSJ}_{\Gamma \cup \{x\}}(t)$, $\text{SNSJ}_{\Gamma}(u)$ and $\mathbb{T}_{\Gamma \cup \{x\}}(t)\{x/u\} \in \mathcal{SN}_{\lambda \text{void}/o}$ then $\text{SNSJ}_{\Gamma}(t\{x/u\})$.*

Proof. By induction on t using Lemma 6.7. □

Lemma 6.10. *Let $t_0 \in \mathcal{T}_{\mathbf{v}}$ s.t. $\mathbb{T}_{\Gamma}(t_0) \in \mathcal{SN}_{\lambda \text{void}/o}$ and $\text{SNSJ}_{\Gamma}(t_0)$. If $t_0 \equiv_{\circ} t_1$ then $\text{SNSJ}_{\Gamma}(t_1)$ and $\mathbb{T}_{\Gamma}(t_0) \equiv_{\circ} \mathbb{T}_{\Gamma}(t_1)$ and $\text{MSJ}_{\Gamma}(t_0) = \text{MSJ}_{\Gamma}(t_1)$. Thus in particular $\eta_{\lambda \text{void}/o}(\mathbb{T}_{\Gamma}(t_0)) = \eta_{\lambda \text{void}/o}(\mathbb{T}_{\Gamma}(t_1))$.*

Proof. By induction on $t_0 \equiv_{\circ} t_1$. □

The next lemma deals with the unboxing rule, which requires a complex induction.

Lemma 6.11. *Let $t_0 \in \mathcal{T}_{\mathbf{v}}$ s.t. $\mathbb{T}_{\Gamma}(t_0) \in \mathcal{SN}_{\lambda \text{void}/o}$ and $\text{SNSJ}_{\Gamma}(t_0)$. If $t_0 = B[s[-/u]] \rightarrow_{\mathbf{u}} B[s][-/u] = t_1$, where B does not bind u , then $\text{SNSJ}_{\Gamma}(t_1)$ and one of the following three cases hold:*

- If $\Gamma \cap \mathbf{fv}(u) = \emptyset$ then
 - (1) Either $\mathbb{T}_{\Gamma}(t_0) = \mathbb{T}_{\Gamma}(t_1)$ and $\text{MSJ}_{\Gamma}(t_0) > \text{MSJ}_{\Gamma}(t_1)$,
 - (2) Or $\mathbb{T}_{\Gamma}(t_0) \rightarrow_{\mathbf{h}} \mathbb{T}_{\Gamma}(t_1)$;
- If $\Gamma \cap \mathbf{fv}(u) \neq \emptyset$ then $\mathbb{T}_{\Gamma}(t_0) \rightarrow_{\mathbf{u}, \mathbf{h}/o}^+ \mathbb{T}_{\Gamma}(t_1)$.

Proof. By induction on $B[\cdot]$.

- Base cases:

– $B[\cdot] = vC[\cdot]$: then $t_0 = vC[s[-/u]] \rightarrow_{\mathbf{u}} (vC[s])[-/u] = t_1$. Hence $\mathbb{T}_{\Gamma}(t_0) = \mathbb{T}_{\Gamma}(v)C[s[-/u]]$ and $\text{SNSJ}_{\Gamma}(t_0)$ iff $\text{SNSJ}_{\Gamma}(v)$. There are two cases:

- (1) $\Gamma \cap \mathbf{fv}(u) = \emptyset$: then $\mathbb{T}_{\Gamma}(t_0) \rightarrow_{\mathbf{h}} \mathbb{T}_{\Gamma}(v)C[s] = \mathbb{T}_{\Gamma}(t_1)$. To show $\text{SNSJ}_{\Gamma}(t_1)$ we need $\overline{\text{SNSJ}_{\Gamma}(v)}$ & $\bar{u} \in \mathcal{SN}_{\lambda \text{void}/o}$. The first point is equivalent to $\text{SNSJ}_{\Gamma}(t_0)$, which holds by hypothesis, the second holds since u is a subterm of $\mathbb{T}_{\Gamma}(t_0) \in \mathcal{SN}_{\lambda \text{void}/o}$.
- (2) $\Gamma \cap \mathbf{fv}(u) \neq \emptyset$: then $\mathbb{T}_{\Gamma}(t_0) \rightarrow_{\mathbf{u}} (\mathbb{T}_{\Gamma}(v)C[s])[-/u] = \mathbb{T}_{\Gamma}(t_1)$. To show $\text{SNSJ}_{\Gamma}(t_1)$ we need $\overline{\text{SNSJ}_{\Gamma}(v)}$, which holds by hypothesis.

– $B[\cdot] = v[-/C[\cdot]]$: there are four cases:

- (1) $\Gamma \cap \mathbf{fv}(u) = \emptyset$ & $\Gamma \cap \mathbf{fv}(C[s]) = \emptyset$: then $\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(v) = \mathbb{T}_\Gamma(t_1)$. Also, $\text{SNSJ}_\Gamma(t_0)$ implies so that $\text{SNSJ}_\Gamma(v) \& C[s[-/u]] \in \mathcal{SN}_{\lambda_{\text{void/o}}}$. To show $\text{SNSJ}_\Gamma(t_1)$ we need $C[s] \in \mathcal{SN}_{\lambda_{\text{void/o}}}$ & $u \in \mathcal{SN}_{\lambda_{\text{void/o}}}$, which clearly follows from $C[s[-/u]] \in \mathcal{SN}_{\lambda_{\text{void/o}}}$. We still need to show that $\text{MSJ}_\Gamma(t_0) > \text{MSJ}_\Gamma(t_1)$ which holds because $\text{MSJ}_\Gamma(t_1)$ is just $\text{MSJ}_\Gamma(t_0)$ where the pair $\langle \eta_{\lambda_{\text{void}}}(C[s[-/u]]), |C[s[-/u]]| \rangle \in \text{MSJ}_\Gamma(t_0)$ is replaced by the strictly smaller multiset $[\langle \eta_{\lambda_{\text{void}}}(C[s]), |C[s[-/u]]| \rangle, \langle \eta_{\lambda_{\text{void}}}(u), |u| \rangle]$.
- (2) $\Gamma \cap \mathbf{fv}(u) = \emptyset$ & $\Gamma \cap \mathbf{fv}(C[s]) \neq \emptyset$: then

$$\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(v)[-/C[s[-/u]]] \rightarrow_{\mathbf{h}} \mathbb{T}_\Gamma(v)[-/C[s]] = \mathbb{T}_\Gamma(t_1)$$

Also, $\text{SNSJ}_\Gamma(t_0)$ implies $\text{SNSJ}_\Gamma(v)$. To show $\text{SNSJ}_\Gamma(t_1)$ we need $\text{SNSJ}_\Gamma(v) \& u \in \mathcal{SN}_{\lambda_{\text{void/o}}}$, which then holds by hypothesis and because u is a subterm of $\mathbb{T}_\Gamma(t_0) \in \mathcal{SN}_{\lambda_{\text{void/o}}}$.

- (3) $\Gamma \cap \mathbf{fv}(u) \neq \emptyset$ & $\Gamma \cap \mathbf{fv}(C[s]) = \emptyset$: then

$$\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(v)[-/C[s[-/u]]] \rightarrow_{\mathbf{u}} \mathbb{T}_\Gamma(v)[-/C[s]][-/u] \rightarrow_{\mathbf{h}} \mathbb{T}_\Gamma(v)[-/u] = \mathbb{T}_\Gamma(t_1)$$

Also, $\text{SNSJ}_\Gamma(t_0)$ implies $\text{SNSJ}_\Gamma(v)$. To show $\text{SNSJ}_\Gamma(t_1)$ we need $\text{SNSJ}_\Gamma(v) \& C[s] \in \mathcal{SN}_{\lambda_{\text{void/o}}}$, which holds by the hypothesis and the fact that $C[s]$ is a subterm of a \mathbf{h} -reduct of $\mathbb{T}_\Gamma(t_0) \in \mathcal{SN}_{\lambda_{\text{void/o}}}$.

- (4) $\Gamma \cap \mathbf{fv}(u) \neq \emptyset$ & $\Gamma \cap \mathbf{fv}(C[s]) \neq \emptyset$: then

$$\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(v)[-/C[s[-/u]]] \rightarrow_{\mathbf{u}} \mathbb{T}_\Gamma(v)[-/C[s]][-/u] = \mathbb{T}_\Gamma(t_1)$$

Also $\text{SNSJ}_\Gamma(t_0)$ implies $\text{SNSJ}_\Gamma(v)$, which is equivalent to $\text{SNSJ}_\Gamma(t_1)$.

• Inductive cases:

- $B[\cdot] = B'[\cdot][-/v]$: We have $t_0 = B'[s[-/u]][-/v] \rightarrow_{\mathbf{u}} B'[s][-/v][-/u] = t_1$. Also $B'[s[-/u]] \rightarrow_{\mathbf{u}} B'[s][-/u]$ and the hypothesis $\mathbb{T}_\Gamma(t_0) \in \mathcal{SN}_{\lambda_{\text{void/o}}}$ and $\text{SNSJ}_\Gamma(t_0)$ imply in particular $\mathbb{T}_\Gamma(B'[s[-/u]]) \in \mathcal{SN}_{\lambda_{\text{void/o}}}$ and $\text{SNSJ}_\Gamma(B'[s[-/u]])$. The i.h. gives $\text{SNSJ}_\Gamma(B'[s][-/u])$ and we distinguish several cases:

- (1) $\Gamma \cap \mathbf{fv}(u) = \emptyset$ & $\Gamma \cap \mathbf{fv}(v) = \emptyset$: The hypothesis $\text{SNSJ}_\Gamma(t_0)$ implies in particular $v \in \mathcal{SN}_{\lambda_{\text{void/o}}}$ and the i.h. $\text{SNSJ}_\Gamma(B'[s][-/u])$ gives $\text{SNSJ}_\Gamma(B'[s]) \& u \in \mathcal{SN}_{\lambda_{\text{void/o}}}$, so we conclude also $\text{SNSJ}_\Gamma(t_1)$. We now consider two cases:

- (a) If $u_0 = \mathbb{T}_\Gamma(B'[s[-/u]]) = \mathbb{T}_\Gamma(B'[s][-/u])$ and $\text{MSJ}_\Gamma(B'[s[-/u]]) > \text{MSJ}_\Gamma(B'[s]) \sqcup \langle \eta_{\lambda_{\text{void/o}}}(u), |u| \rangle$, then $\mathbb{T}_\Gamma(t_0) = u_0 = \mathbb{T}_\Gamma(t_1)$ and $\text{MSJ}_\Gamma(t_0) = \text{MSJ}_\Gamma(B'[s[-/u]]) \sqcup \langle \eta_{\lambda_{\text{void/o}}}(v), |v| \rangle > \text{MSJ}_\Gamma(B'[s]) \sqcup \langle \eta_{\lambda_{\text{void/o}}}(u), |u| \rangle \sqcup \langle \eta_{\lambda_{\text{void/o}}}(v), |v| \rangle = \text{MSJ}_\Gamma(t_1)$.
- (b) If $u_0 = \mathbb{T}_\Gamma(B'[s[-/u]]) \rightarrow_{\mathbf{h}} \mathbb{T}_\Gamma(B'[s][-/u]) = u_1$, then $\mathbb{T}_\Gamma(t_0) = u_0 \rightarrow_{\mathbf{h}} u_1 = \mathbb{T}_\Gamma(t_1)$.

- (2) $\Gamma \cap \mathbf{fv}(u) = \emptyset$ & $\Gamma \cap \mathbf{fv}(v) \neq \emptyset$: The i.h. $\text{SNSJ}_\Gamma(B'[s][-/u])$ gives $\text{SNSJ}_\Gamma(B'[s]) \& u \in \mathcal{SN}_{\lambda_{\text{void/o}}}$, so we conclude also $\text{SNSJ}_\Gamma(t_1)$. We now consider two cases:

- (a) If $u_0 = \mathbb{T}_\Gamma(B'[s[-/u]]) = \mathbb{T}_\Gamma(B'[s][-/u])$ and $\text{MSJ}_\Gamma(B'[s[-/u]]) > \text{MSJ}_\Gamma(B'[s]) \sqcup \langle \eta_{\lambda_{\text{void/o}}}(u), |u| \rangle$, then $\mathbb{T}_\Gamma(t_0) = u_0[-/v] = \mathbb{T}_\Gamma(B'[s])[-/v] = \mathbb{T}_\Gamma(t_1)$ and $\text{MSJ}_\Gamma(t_0) = \text{MSJ}_\Gamma(B'[s[-/u]]) \sqcup \text{MSJ}_\Gamma(v) > \text{MSJ}_\Gamma(B'[s]) \sqcup \langle \eta_{\lambda_{\text{void/o}}}(u), |u| \rangle \sqcup \text{MSJ}_\Gamma(v) = \text{MSJ}_\Gamma(t_1)$.

- (b) If $u_0 = \mathbb{T}_\Gamma(B'[s[-/u]]) \rightarrow_{\mathbf{h}} \mathbb{T}_\Gamma(B'[s][-/u]) = u_1$, then

$$\mathbb{T}_\Gamma(t_0) = u_0[-/v] \rightarrow_{\mathbf{h}} u_1[-/v] = \mathbb{T}_\Gamma(B'[s])[-/v] = \mathbb{T}_\Gamma(t_1)$$

- (3) $\Gamma \cap \mathbf{fv}(u) \neq \emptyset$: Then the i.h. gives $u_0 = \mathbb{T}_\Gamma(B'[s[-/u]]) \rightarrow_{\mathbf{h}, u/o}^+ \mathbb{T}_\Gamma(B'[s][-/u]) = u_1$.

(a) $\Gamma \cap \mathbf{fv}(v) = \emptyset$: then

$$\mathbb{T}_\Gamma(t_0) = u_0 \rightarrow_{\mathbf{h}, \mathbf{u}/\circ}^+ u_1 = \mathbb{T}_\Gamma(B'[\![s]\!])[-/u] = \mathbb{T}_\Gamma(t_1)$$

Also $\text{SNSJ}_\Gamma(t_0)$ implies $v \in \mathcal{SN}_{\lambda\text{void}/\circ}$ and the i.h. $\text{SNSJ}_\Gamma(B'[\![s]\!])[-/u]$ implies $\text{SNSJ}_\Gamma(B'[\![s]\!])$, we thus conclude $\text{SNSJ}_\Gamma(t_1)$.

(b) $\Gamma \cap \mathbf{fv}(v) \neq \emptyset$: then

$$\mathbb{T}_\Gamma(t_0) = u_0[-/v] \rightarrow_{\mathbf{h}, \mathbf{u}/\circ}^+ u_1[-/v] = \mathbb{T}_\Gamma(B'[\![s]\!])[-/u][-/v] \equiv_{\circ} \mathbb{T}_\Gamma(B'[\![s]\!])[-/v][-/u] = \mathbb{T}_\Gamma(t_1)$$

Also, the i.h. $\text{SNSJ}_\Gamma(B'[\![s]\!])[-/u]$ implies $\text{SNSJ}_\Gamma(B'[\![s]\!])$, we thus conclude $\text{SNSJ}_\Gamma(t_1)$.

– The cases $B[\![\cdot]\!] = \lambda y. B'[\![\cdot]\!]$ and $B[\![\cdot]\!] = B'[\![\cdot]\!]w$ are similar to the previous ones. \square

The following lemma states that the measure we use for proving **VIE** for $\lambda\text{void}/\circ$ decreases with every rewriting step.

Lemma 6.12. *Let $t_0 \in \mathcal{T}_v$ s.t. $\mathbb{T}_\Gamma(t_0) \in \mathcal{SN}_{\lambda\text{void}/\circ}$ and $\text{SNSJ}_\Gamma(t_0)$. If $t_0 \rightarrow_{\lambda\text{void}} t_1$ then $\text{SNSJ}_\Gamma(t_1)$ and*

- *Either $\mathbb{T}_\Gamma(t_0) \rightarrow_{\lambda\text{void}/\circ}^+ \mathbb{T}_\Gamma(t_1)$ or*
- *$\mathbb{T}_\Gamma(t_0) = \mathbb{T}_\Gamma(t_1)$ and $\text{MSJ}_\Gamma(t_0) > \text{MSJ}_\Gamma(t_1)$.*

Proof. By induction on $t_0 \rightarrow_{\lambda\text{void}} t_1$.

- Base cases:

– If $t_0 = (\lambda x.s)\mathbf{L} u \rightarrow_{\text{ab}} s[-/u]\mathbf{L} = t_1$, where $x \notin \mathbf{fv}(s)$.

Let $\mathbf{L} := [-/v_1] \dots [-/v_k]$, $Q := \{v_i \mid \Gamma \cap \mathbf{fv}(v_i) \neq \emptyset, i \in \{1, \dots, k\}\}$ and $\overline{Q} := \{v_i \mid \Gamma \cap \mathbf{fv}(v_i) = \emptyset, i \in \{1, \dots, k\}\}$. Define \mathbf{L}_Q the sublist of \mathbf{L} containing only the elements in Q .

We have

- * $\text{SNSJ}_\Gamma(t_0)$ iff $\text{SNSJ}_{\Gamma \cup \{x\}}(s) =_{L. 6.7} \text{SNSJ}_\Gamma(s)$ and $v_j \in \mathcal{SN}_{\lambda\text{void}/\circ}$ for every $v_j \in \overline{Q}$.
- * $\mathbb{T}_\Gamma(t_0) = (\lambda x. \mathbb{T}_{\Gamma \cup \{x\}}(s))\mathbf{L}_Q u =_{L. 6.7} (\lambda x. \mathbb{T}_\Gamma(s))\mathbf{L}_Q u$.

There are two cases:

- (1) $\Gamma \cap \mathbf{fv}(u) \neq \emptyset$. We have $\mathbb{T}_\Gamma(t_1) = \mathbb{T}_\Gamma(s)[-/u]\mathbf{L}_Q$. Then $\mathbb{T}_\Gamma(t_0) \rightarrow_{\text{ab}} \mathbb{T}_\Gamma(t_1)$. Moreover, $\text{SNSJ}_\Gamma(t_1)$ iff $\text{SNSJ}_\Gamma(s)$ and $v_j \in \mathcal{SN}_{\lambda\text{void}/\circ}$ for every $v_j \in \overline{Q}$, which holds by the hypothesis $\text{SNSJ}_\Gamma(t_0)$.
- (2) $\Gamma \cap \mathbf{fv}(u) = \emptyset$. We have $\mathbb{T}_\Gamma(t_1) = \mathbb{T}_\Gamma(s)\mathbf{L}_Q$. Then $\mathbb{T}_\Gamma(t_0) \rightarrow_{\text{ab}} \mathbb{T}_\Gamma(s)[-/u]\mathbf{L}_Q \rightarrow_{\mathbf{h}} \mathbb{T}_\Gamma(s)\mathbf{L}_Q = \mathbb{T}_\Gamma(t_1)$. Moreover, $\text{SNSJ}_\Gamma(t_1)$ iff $\text{SNSJ}_\Gamma(s)$ and $u \in \mathcal{SN}_{\lambda\text{void}/\circ}$ and $v_j \in \mathcal{SN}_{\lambda\text{void}/\circ}$ for every $v_j \in \overline{Q}$. The first and third parts follow from the hypothesis $\text{SNSJ}_\Gamma(t_0)$ while the second one follows from the hypothesis $\mathbb{T}_\Gamma(t_0) \in \mathcal{SN}_{\lambda\text{void}/\circ}$.

– $t_0 = (\lambda x.s)\mathbf{L} u \rightarrow_{\beta} s\{x/u\}\mathbf{L} = t_1$, where $x \in \mathbf{fv}(s)$.

Let \mathbf{L} , Q , \overline{Q} and \mathbf{L}_Q be as in the previous case. We have

- * $\text{SNSJ}_\Gamma(t_0)$ iff $\text{SNSJ}_{\Gamma \cup \{x\}}(s)$ and $v_j \in \mathcal{SN}_{\lambda\text{void}/\circ}$ for every $v_j \in \overline{Q}$.
- * $\mathbb{T}_\Gamma(t_0) = (\lambda x. \mathbb{T}_{\Gamma \cup \{x\}}(s))\mathbf{L}_Q u$.

We have $\mathbb{T}_\Gamma(t_0) \rightarrow_{\beta} \mathbb{T}_{\Gamma \cup \{x\}}(s)\{x/u\}\mathbf{L}_Q \rightarrow_{\mathbf{h}}^* (L. 6.8) \mathbb{T}_\Gamma(s\{x/u\})\mathbf{L}_Q = \mathbb{T}_\Gamma(t_1)$. Thus in particular $\mathbb{T}_{\Gamma \cup \{x\}}(s)\{x/u\} \in \mathcal{SN}_{\lambda\text{void}/\circ}$.

Since u is a subterm of $\mathbb{T}_\Gamma(t_0)$, then $u \in \mathcal{SN}_{\lambda\text{void}/\circ}$ and so $\text{SNSJ}_\Gamma(u)$. Then $\text{SNSJ}_\Gamma(t_1)$ iff $\text{SNSJ}_\Gamma(s\{x/u\})$ and $v_j \in \mathcal{SN}_{\lambda\text{void}/\circ}$ for every $v_j \in \overline{Q}$. The first part holds by Lemma 6.9, the second one from the hypothesis $\text{SNSJ}_\Gamma(t_0)$.

– $t_0 = u[-/v] \rightarrow_{\mathbf{h}} u[-/v_1] \dots [-/v_k] = t_1$, where $k \geq 0$, $v_j < u$ for all j and $\mathbf{fv}(v_j) \subseteq \mathbf{fv}(u)$.

There are two cases:

- (1) $\Gamma \cap \mathbf{fv}(v) = \emptyset$: we have that $\text{SNSJ}_{\Gamma}(t_0)$ implies $\text{SNSJ}_{\Gamma}(t_1)$. Then $\mathbb{T}_{\Gamma}(t_0) = \mathbb{T}_{\Gamma}(u) = \mathbb{T}_{\Gamma}(t_1)$, moreover the pair $\langle \eta_{\lambda\text{void}}(u), |u| \rangle$ of $\text{MSJ}_{\Gamma}(t_0)$ is replaced by the following multiset of $\text{MSJ}_{\Gamma}(t_1)$: $[\langle \eta_{\lambda\text{void}}(v_1), |v_1| \rangle, \dots, \langle \eta_{\lambda\text{void}}(v_k), |v_k| \rangle]$. Since $\eta_{\lambda\text{void}}(v) \geq \eta_{\lambda\text{void}}(v_i)$ and $|v| > |v_i|$ we thus conclude $\text{MSJ}_{\Gamma}(t_0) > \text{MSJ}_{\Gamma}(t_1)$.
- (2) $\Gamma \cap \mathbf{fv}(v) \neq \emptyset$: let Q and \bar{Q} as in de dB-case. Then $\text{SNSJ}_{\Gamma}(t_1)$ iff the terms in \bar{Q} are $\mathcal{SN}_{\lambda\text{void}/\text{o}}$ and $\text{SNSJ}_{\Gamma}(u)$ holds: the former requirement holds because $\mathbb{T}_{\Gamma}(t_0) = \mathbb{T}_{\Gamma}(u)[-/v]$ and so $v \in \mathcal{SN}_{\lambda\text{void}/\text{o}}$, the latter because $\text{SNSJ}_{\Gamma}(t_0)$ iff $\text{SNSJ}_{\Gamma}(u)$. Last, $\mathbb{T}_{\Gamma}(t_1) = \mathbb{T}_{\Gamma}(u)\mathbf{L}_Q$, where \mathbf{L}_Q is the list of substitutions associated to the elements in Q , then

$$\mathbb{T}_{\Gamma}(t_0) = \mathbb{T}_{\Gamma}(u)[-/v] \rightarrow_{\mathbf{h}} \mathbb{T}_{\Gamma}(u)\mathbf{L}_Q = \mathbb{T}_{\Gamma}(t_1)$$

– $t_0 = B[s[-/u]] \rightarrow_{\mathbf{u}} B[s][-/u] = t_1$. This case holds by Lemma 6.11.

• Inductive cases:

– $t_0 = u[-/v] \rightarrow_{\lambda\text{void}} u[-/v'] = t_1$ where $v \rightarrow_{\lambda\text{void}} v'$. We consider three cases.

- (1) $\mathbf{fv}(v) \cap \Gamma = \emptyset$ & $\mathbf{fv}(v') \cap \Gamma = \emptyset$: We have $\mathbb{T}_{\Gamma}(t_0) = \mathbb{T}_{\Gamma}(u) = \mathbb{T}_{\Gamma}(t_1)$. Also $\text{SNSJ}_{\Gamma}(t_0)$ implies $v \in \mathcal{SN}_{\lambda\text{void}/\text{o}}$ so that $v' \in \mathcal{SN}_{\lambda\text{void}/\text{o}}$ and thus $\text{SNSJ}_{\Gamma}(t_1)$. Finally,

$$\text{MSJ}_{\Gamma}(t_0) = \text{MSJ}_{\Gamma}(u) \sqcup \langle \eta_{\lambda\text{void}/\text{o}}(v), |v| \rangle > \text{MSJ}_{\Gamma}(u) \sqcup \langle \eta_{\lambda\text{void}/\text{o}}(v'), |v'| \rangle = \text{MSJ}_{\Gamma}(t_1)$$

- (2) $\mathbf{fv}(v) \cap \Gamma \neq \emptyset$ & $\mathbf{fv}(v') \cap \Gamma \neq \emptyset$: We have $\text{SNSJ}_{\Gamma}(t_0) = \text{SNSJ}_{\Gamma}(u) = \text{SNSJ}_{\Gamma}(t_1)$. Also $\mathbb{T}_{\Gamma}(t_0) = \mathbb{T}_{\Gamma}(u)[-/v]$ and $\mathbb{T}_{\Gamma}(t_1) = \mathbb{T}_{\Gamma}(u)[-/v']$, thus $\mathbb{T}_{\Gamma}(t_0) \rightarrow_{\lambda\text{void}/\text{o}}^+ \mathbb{T}_{\Gamma}(t_1)$.

- (3) $\mathbf{fv}(v) \cap \Gamma \neq \emptyset$ & $\mathbf{fv}(v') \cap \Gamma = \emptyset$: We have that $\mathbb{T}_{\Gamma}(t_0) \in \mathcal{SN}_{\lambda\text{void}/\text{o}}$ implies $v \in \mathcal{SN}_{\lambda\text{void}/\text{o}}$, so that $v' \in \mathcal{SN}_{\lambda\text{void}/\text{o}}$ and $\text{SNSJ}_{\Gamma}(t_1)$. Then $\mathbb{T}_{\Gamma}(t_0) = \mathbb{T}_{\Gamma}(u)[-/v] \rightarrow_{\mathbf{h}} \mathbb{T}_{\Gamma}(u) = \mathbb{T}_{\Gamma}(t_1)$.

– All the other cases are straightforward. □

Theorem 6.13 (VIE for $\lambda\text{void}/\text{o}$). *Let $t \in \mathcal{T}_v$ s.t. $\mathbb{T}_{\emptyset}(t) \in \mathcal{SN}_{\lambda\text{void}/\text{o}}$ and $\text{SNSJ}_{\emptyset}(t)$, then $t \in \mathcal{SN}_{\lambda\text{void}/\text{o}}$.*

Proof. We proceed by induction on the measure $m(t) = \langle \eta_{\lambda\text{void}/\text{o}}(\mathbb{T}_{\emptyset}(t)), \text{MSJ}_{\emptyset}(t) \rangle$. To show $t \in \mathcal{SN}_{\lambda\text{void}/\text{o}}$ it is sufficient to show $t' \in \mathcal{SN}_{\lambda\text{void}/\text{o}}$ for every $\lambda\text{void}/\text{o}$ -reduct of t . Take any of such reducts t' . Then Lemmas 6.10 and 6.12 guarantee $\mathbb{T}_{\emptyset}(t') \in \mathcal{SN}_{\lambda\text{void}/\text{o}}$ and $\text{SNSJ}_{\emptyset}(t')$. Moreover, \equiv_{o} preserves $m(t)$ and $\rightarrow_{\lambda\text{void}/\text{o}}$ strictly decreases $m(t)$. We thus apply the i.h. to conclude. □

The following is a consequence of the previous theorem: let $t, u, \bar{v}_n^1 \in \lambda$ -terms and $s = t[-/u]\bar{v}_n^1$. If $\mathbb{T}_{\emptyset}(s) = t\bar{v}_n^1 \in \mathcal{SN}_{\lambda\text{void}}$ and $\text{SNSJ}_{\emptyset}(s)$ holds, i.e. $u \in \mathcal{SN}_{\lambda\text{void}}$, then $s = t[-/u]\bar{v}_n^1 \in \mathcal{SN}_{\lambda\text{void}}$. Hence:

Corollary 6.14 (IE for $\lambda\text{void}/\text{o}$). *The $\lambda\text{void}/\text{o}$ -calculus enjoys the IE property.*

Lemma 6.15 (Adequacy of IE). *If $\lambda\text{void}/\text{o}$ verifies IE, then $\lambda\text{void}/\text{o}$ satisfies PSN.*

Proof. By Theorem 4.1 it is sufficient to show **F0**, **F1** and **F2**. The first two properties are straightforward. To show **F2** assume $v \in \mathcal{SN}_{\lambda\text{void}}$ and $u\{x/v\} \bar{t}_n^1 \in \mathcal{SN}_{\lambda\text{void}}$, both are λ -terms. Then in particular $u, v, \bar{t}_n^1 \in \mathcal{SN}_{\lambda\text{void}}$. We show that $t = (\lambda x.u) v \bar{t}_n^1 \in \mathcal{SN}_{\lambda\text{void}}$ by induction on $\eta_{\lambda\text{void}}(u) + \eta_{\lambda\text{void}}(v) + \sum_i \eta_{\lambda\text{void}}(t_i)$. For that, it is sufficient to show that every λvoid -reduct of t is in $\mathcal{SN}_{\lambda\text{void}}$. If the λvoid -reduct of t is internal we conclude by the i.h. If $t \rightarrow_{\beta} u\{x/v\} \bar{t}_n^1 = t'$ with $x \in \mathbf{fv}(u)$, then

$t' \in \mathcal{SN}_{\lambda\text{void}}$ by hypothesis. If $t \rightarrow_{\text{dB}} u[-/v]t_n^1 = t'$, then $t' \in \mathcal{SN}_{\lambda\text{void}}$ by the **IE** property. There is no other possible λvoid -reduct of t which is a λ -term and has no jumps. \square

Corollary 6.16 (PSN for $\lambda\text{void}/\text{o}$). *The $\lambda\text{void}/\text{o}$ -calculus enjoys PSN, i.e. if $t \in \mathcal{T}_\lambda \cap \mathcal{SN}_\beta$, then $t \in \mathcal{SN}_{\lambda\text{void}/\text{o}}$.*

Proof. Corollary 6.14 and Lemma 6.15 imply that the requirements **F0**, **F1** and **F2** hold for $\lambda\text{void}/\text{o}$, then we conclude by Theorem 4.1. \square

6.3. Projecting $\lambda\text{j}/\text{obox}$ into $\lambda\text{void}/\text{o}$. In order to relate the $\lambda\text{j}/\text{obox}$ and the $\lambda\text{void}/\text{o}$ calculi we define a projection function from λj -terms to λvoid -terms:

$$\begin{aligned} \mathbf{wj}(x) &:= x \\ \mathbf{wj}(\lambda x.t) &:= \lambda x.\mathbf{wj}(t) \\ \mathbf{wj}(tu) &:= \mathbf{wj}(t)\mathbf{wj}(u) \\ \mathbf{wj}(t[x/u]) &:= \begin{cases} \mathbf{wj}(t)\{x/\mathbf{wj}(u)\} & \text{if } x \in \text{fv}(t) \\ \mathbf{wj}(t)[-/\mathbf{wj}(u)] & \text{if } x \notin \text{fv}(t) \end{cases} \end{aligned}$$

Notice that $\text{fv}(t) = \text{fv}(\mathbf{wj}(t))$. Also, $\mathbf{wj}(t) = t$ if $t \in \mathcal{T}_\lambda$.

We now state some basic static properties of \mathbf{wj} .

Lemma 6.17. *Let $t, u \in \mathcal{T}$. Then,*

- (1) $\mathbf{wj}(t\{x/u\}) = \mathbf{wj}(t)\{x/\mathbf{wj}(u)\}$.
- (2) $u < t$ and $\text{fv}(u) \subseteq \text{fv}(t)$ imply $\mathbf{wj}(u) < \mathbf{wj}(t)$.

Proof. By induction on t . \square

Lemma 6.18 (Projection). *Let $t_0 \in \mathcal{T}$. Then,*

- (1) $t_0 \rightarrow_{\text{dB}} t_1$ implies $\mathbf{wj}(t_0) \rightarrow_{\beta, \text{dB}}^+ \mathbf{wj}(t_1)$.
- (2) $t_0 \rightarrow_{\mathbf{w}} t_1$ implies $\mathbf{wj}(t_0) \rightarrow_{\mathbf{h}, \text{u}/\text{o}}^+ \mathbf{wj}(t_1)$.
- (3) $t_0 \rightarrow_{\text{d}, \text{c}} t_1$ implies $\mathbf{wj}(t_0) = \mathbf{wj}(t_1)$.
- (4) $t_0 \equiv_{\circ} t_1$ implies $\mathbf{wj}(t_0) \equiv_{\circ} \mathbf{wj}(t_1)$.
- (5) $t_0 \equiv_{\text{box}_1, \text{box}_2} t_1$ implies $\mathbf{wj}(t_0) = \mathbf{wj}(t_1)$.

Proof.

- $t_0 = (\lambda x.t)\mathbf{L} u \rightarrow_{\text{dB}} t[x/u]\mathbf{L} = t_1$.

Let $\mathbf{M} = [-/\mathbf{wj}(v_i)]_m^1$ (resp. ρ) be the sequence of jumps (resp. the meta-level substitution) resulting from the projection of t_0 , i.e. $\mathbf{wj}(t_0) = (\lambda x.\mathbf{wj}(t))\mathbf{M} \rho \mathbf{wj}(u)$.

If $x \in \text{fv}(t)$, then

$$\begin{aligned} \mathbf{wj}(t_0) &= (\lambda x.\mathbf{wj}(t)\rho)[-/\mathbf{wj}(v_i)\rho]_m^1 \mathbf{wj}(u) \rightarrow_{\beta} \\ &= \mathbf{wj}(t)\rho\{x/\mathbf{wj}(u)\}[-/\mathbf{wj}(v_i)\rho]_m^1 = \\ &= \mathbf{wj}(t)\{x/\mathbf{wj}(u)\}\rho[-/\mathbf{wj}(v_i)\rho]_m^1 = \mathbf{wj}(t_1) \end{aligned}$$

If $x \notin \text{fv}(t)$, then

$$\begin{aligned} \mathbf{wj}(t_0) &= (\lambda x.\mathbf{wj}(t)\rho)[-/\mathbf{wj}(v_i)\rho]_m^1 \mathbf{wj}(u) \rightarrow_{\text{dB}} \\ &= \mathbf{wj}(t)\rho[-/\mathbf{wj}(u)][-/\mathbf{wj}(v_i)\rho]_m^1 = \\ &= \mathbf{wj}(t)[-/\mathbf{wj}(u)][-/\mathbf{wj}(v_i)\rho]_m^1 \rho = \mathbf{wj}(t_1) \end{aligned}$$

- $t_0 = t[x/u] \rightarrow_{\mathbf{w}} t = t_1$ where $|t|_x = 0$. Then $\mathbf{wj}(t[x/u]) = \mathbf{wj}(t)[-/\mathbf{wj}(u)] \rightarrow_{\mathbf{h}} \mathbf{wj}(t)$.

- $t_0 = t[x/u] \rightarrow_d t\{x/u\} = t_1$ where $|t|_x = 1$. Then $\mathbf{wj}(t[x/u]) = \mathbf{wj}(t)\{x/\mathbf{wj}(u)\} =_{L. 6.17:1} \mathbf{wj}(t\{x/u\})$.
- $t_0 = t[x/u] \rightarrow_c t_{[y]_x}[x/u][y/u] = t_1$ where $|t|_x \geq 2$. Then $\mathbf{wj}(t[x/u]) = \mathbf{wj}(t)\{x/\mathbf{wj}(u)\} = \mathbf{wj}(t_{[y]_x})\{y/\mathbf{wj}(u)\}\{x/\mathbf{wj}(u)\} =_{L. 6.17:1} \mathbf{wj}(t_{[y]_x}[x/u][y/u])$
- $t_0 = t[x/u][y/v] \equiv_{cs} t[y/v][x/u] = t_1$ where $y \notin \mathbf{fv}(u)$ $x \notin \mathbf{fv}(v)$. There are two cases:
If $x \in \mathbf{fv}(t)$ or $y \in \mathbf{fv}(t)$, then we obtain $\mathbf{wj}(t_0) = \mathbf{wj}(t_1)$.
If $x \notin \mathbf{fv}(t)$ and $y \notin \mathbf{fv}(t)$, then $\mathbf{wj}(t_0) = \mathbf{wj}(t)[-/\mathbf{wj}(u)][-/\mathbf{wj}(v)] \equiv_{cs} \mathbf{wj}(t)[-/\mathbf{wj}(v)][-/\mathbf{wj}(u)] = \mathbf{wj}(t_1)$
- $t_0 = (\lambda y.t)[x/u] \equiv_{\sigma_1} \lambda y.t[x/u] = t_1$ where $y \notin \mathbf{fv}(u)$. There are two cases:
If $x \in \mathbf{fv}(\lambda y.t)$, then $\mathbf{wj}(t_0) = (\lambda y.\mathbf{wj}(t))\{x/\mathbf{wj}(u)\} = \lambda y.\mathbf{wj}(t)\{x/\mathbf{wj}(u)\} = \mathbf{wj}(t_1)$.
If $x \notin \mathbf{fv}(\lambda y.t)$, then $\mathbf{wj}(t_0) = (\lambda y.\mathbf{wj}(t))[-/\mathbf{wj}(u)] \equiv_{\sigma} \lambda y.\mathbf{wj}(t)[-/\mathbf{wj}(u)] = \mathbf{wj}(t_1)$.
- $t_0 = (tv)[x/u] \equiv_{\sigma_2} t[x/u]v = t_1$ where $x \notin \mathbf{fv}(v)$.
There are two cases:
If $x \in \mathbf{fv}(t)$, then $\mathbf{wj}(t_0) = (\mathbf{wj}(t) \mathbf{wj}(v))\{x/\mathbf{wj}(u)\} = \mathbf{wj}(t)\{x/\mathbf{wj}(u)\} \mathbf{wj}(v) = \mathbf{wj}(t_1)$.
If $x \notin \mathbf{fv}(t)$, then $\mathbf{wj}(t_0) = (\mathbf{wj}(t) \mathbf{wj}(v))[-/\mathbf{wj}(u)] \equiv_{\sigma} \mathbf{wj}(t)[-/\mathbf{wj}(u)] \mathbf{wj}(v) = \mathbf{wj}(t_1)$.
- $t_0 \equiv_{\text{box}_1, \text{box}_2} t_1$. Then trivially $\mathbf{wj}(t_0) = \mathbf{wj}(t_1)$.

The inductive cases:

- $t_0 = u[x/v] \rightarrow$ (resp. \equiv) $u'[x/v] = t_1$, where $u \rightarrow$ (resp. \equiv) u' . If $x \in \mathbf{fv}(u)$ & $x \in \mathbf{fv}(u')$ or $x \notin \mathbf{fv}(u)$ & $x \notin \mathbf{fv}(u')$ then the property is straightforward by the i.h. So let us suppose $x \in \mathbf{fv}(u)$ and $x \notin \mathbf{fv}(u')$ (so that the reduction step is necessarily a \mathbf{w} -step). We have $\mathbf{wj}(u) \rightarrow_{\mathbf{h}, \mathbf{u}/\circ}^+ (i.h.) \mathbf{wj}(u')$. Then $\mathbf{wj}(t_0) = \mathbf{wj}(u)\{x/\mathbf{wj}(v)\} \rightarrow_{\mathbf{h}, \mathbf{u}/\circ}^+ \mathbf{wj}(u')[-/\mathbf{wj}(v)] = \mathbf{wj}(t_1)$ holds by Corollary 6.6.
- All the other cases are straightforward. □

Here are some interesting examples:

t	\rightarrow	t'	$\mathbf{wj}(t)$	\rightarrow^*	$\mathbf{wj}(t')$
$f[y/x][x/u]$	$\rightarrow_{\mathbf{w}}$	$f[x/u]$	$f[-/u]$	$=$	$f[-/u]$
$f[y/xz][x/u][z/v]$	$\rightarrow_{\mathbf{w}}$	$f[x/u][z/v]$	$f[-/uv]$	$\rightarrow_{\mathbf{h}}$	$f[-/u][-/v]$
$f[y/xx][x/u]$	$\rightarrow_{\mathbf{w}}$	$f[x/u]$	$f[-/uu]$	$\rightarrow_{\mathbf{h}}^+$	$f[-/u]$
$(f[w/f[y/xz]]g)[x/u][z/v]$	$\rightarrow_{\mathbf{w}}$	$(f[w/f]g)[x/u][z/v]$	$(f[-/f[-/uv]]g)$	$\rightarrow_{\mathbf{h}, \mathbf{u}}$	$(f[-/f]g)[-/u][-/v]$

The previous property allows us to conclude with one of the main results of this paper.

Theorem 6.19 (PSN for $\lambda\mathbf{j}/\text{obox}$). *The $\lambda\mathbf{j}/\text{obox}$ -calculus enjoys PSN, i.e. if $t \in \mathcal{T}_\lambda \cap \mathcal{SN}_\beta$, then $t \in \mathcal{SN}_{\lambda\mathbf{j}/\text{obox}}$.*

Proof. We apply Theorem 2.3, where $\mathcal{A} = \lambda\text{void}$, $\mathcal{A}_1 = \{\mathbf{d}, \mathbf{c}\}$, $\mathcal{A}_2 = \{\mathbf{dB}, \mathbf{w}\}$, \mathbf{E} is \equiv_{obox} , \mathbf{F} is \equiv_{\circ} and $t \mathcal{R} \mathbf{wj}(t)$. Property **(P0)** holds by Lemma 6.18:4-5, Property **(P1)** holds by Lemma 6.18:3, Property **(P2)** holds by Lemma 6.18:1-2 and Property **(P3)** holds by Lemma 5.6. Now, take $t \in \mathcal{T}_\lambda \cap \mathcal{SN}_\beta$ so that Corollary 6.16 gives $t \in \mathcal{SN}_{\lambda\text{void}/\circ}$. Since $\mathbf{wj}(t) = t$, then $t \in \mathcal{SN}_{\lambda\mathbf{j}/\text{obox}}$ by application of the Theorem. □

7. CONSEQUENCES OF THE MAIN RESULT

In this section we show how the strong result obtained in Section 6.3 can be used to prove PSN for different variants of the $\lambda\mathbf{j}/\mathbf{obox}$ -calculus.

7.1. Adding $\{\mathbf{h}, \mathbf{u}\}$ to $\lambda\mathbf{j}/\mathbf{obox}$. First of all, we can show that the $\{\mathbf{h}, \mathbf{u}\}$ rules of $\lambda\mathbf{void}/\mathbf{o}$ can be added to $\lambda\mathbf{j}/\mathbf{obox}$ without breaking PSN. The main point of this extension is to show that it is safe to consider unboxing (for void jumps) together with the box equations (for non-void jumps). For that, we first extend the rules \mathbf{h} and \mathbf{u} to act on the whole set \mathcal{T} and not only on \mathcal{T}_v (but they still concern void substitutions only). Formally,

$$\begin{array}{ll} t[x/u] & \mapsto_{\mathbf{h}} t[x_1/u_1] \dots [x_n/u_n] \quad x \notin \mathbf{fv}(t) \ \& \ n \geq 0 \ \& \ \forall i (x_i \text{ fresh} \ \& \ u_i < u \ \& \ \mathbf{fv}(u_i) \subseteq \mathbf{fv}(u)) \\ B[[t[x/u]]] & \mapsto_{\mathbf{u}} B[[t]][x/u] \quad B \text{ does not bind } u \ \& \ x \notin \mathbf{fv}(t) \end{array}$$

Indeed, the \mathbf{wj} function maps $\{\mathbf{h}, \mathbf{u}\}$ -reduction steps of $\{\lambda\mathbf{j}, \mathbf{h}, \mathbf{u}\}/\mathbf{obox}$ into $\{\mathbf{h}, \mathbf{u}\}$ -reduction steps of $\lambda\mathbf{void}/\mathbf{o}$, as the next lemma shows.

Lemma 7.1 (Extended Projection). *Let $t_0 \in \mathcal{T}$. Then,*

- (1) $t_0 \rightarrow_{\mathbf{h}} t_1$ implies $\mathbf{wj}(t_0) \rightarrow_{\mathbf{h}, \mathbf{u}/\mathbf{o}}^+ \mathbf{wj}(t_1)$.
- (2) $t_0 \rightarrow_{\mathbf{u}} t_1$ implies $\mathbf{wj}(t_0) \rightarrow_{\mathbf{h}, \mathbf{u}/\mathbf{o}}^* \mathbf{wj}(t_1)$.

Proof. By induction on the reduction relations.

- $t_0 = t[x/u] \rightarrow_{\mathbf{h}} t[x_1/u_0] \dots [x_n/u_n] = t_1$ where $\forall i (x_i \text{ fresh} \ \& \ u_i < u \ \& \ \mathbf{fv}(u_i) \subseteq \mathbf{fv}(u))$. Then $\mathbf{wj}(t[x/u]) = \mathbf{wj}(t)[-/\mathbf{wj}(u)] \rightarrow_{\mathbf{h}} \mathbf{wj}(t)[-/\mathbf{wj}(u_0)] \dots [-/\mathbf{wj}(u_n)] = \mathbf{wj}(t_1)$. We have $\mathbf{fv}(\mathbf{wj}(u_i)) = \mathbf{fv}(u_i) \subseteq \mathbf{fv}(u) = \mathbf{fv}(\mathbf{wj}(u))$. Then, the inequalities $\mathbf{wj}(u_i) < \mathbf{wj}(u)$ hold by Lemma 6.17:2.
- $t_0 = B[[t[x/u]]] \rightarrow_{\mathbf{u}} B[[t]][x/u] = t_1$ where V does not bind u and $x \notin \mathbf{fv}(t)$. We show a stronger property, namely:

If $t_0 = C[[t[x/u]]] \rightarrow C[[t]][x/u] = t_1$ where C does not bind u and $x \notin \mathbf{fv}(t)$, then $\mathbf{wj}(t_0) \rightarrow_{\mathbf{h}, \mathbf{u}/\mathbf{o}}^* \mathbf{wj}(t_1)$. Then the property we want show is just a particular case of the stronger property. By α -conversion we assume w.l.g. that x is not even free in $C[[t]]$.

We reason by induction on C .

- $t_0 = [[t[x/u]]] \rightarrow_{\mathbf{u}} [[t]][x/u] = t_1$. Then $t_0 = t_1$ so that $\mathbf{wj}(t_0) = \mathbf{wj}(t_1)$.
- $t_0 = C'[[t[x/u]]]v \rightarrow_{\mathbf{u}} (C'[[t]v])[x/u] = t_1$. Then we conclude by using the i.h. and the equivalence \equiv_{σ_2} .
- $t_0 = vC'[[t[x/u]]] \rightarrow_{\mathbf{u}} (vC'[[t]])[x/u] = t_1$. Then we conclude by using the i.h. and the reduction $\rightarrow_{\mathbf{u}}$.
- $t_0 = \lambda y.C'[[t[x/u]]] \rightarrow_{\mathbf{u}} (\lambda y.C'[[t]])[x/u] = t_1$. Then we conclude by using the i.h. and the equivalence \equiv_{σ_1} .
- $t_0 = v[y/C'[[t[x/u]]]] \rightarrow_{\mathbf{u}} v[y/C'[[t]]][x/u] = t_1$. We reason by cases.
If $y \notin \mathbf{fv}(v)$, then

$$\begin{aligned} \mathbf{wj}(t_0) &= \mathbf{wj}(v[y/C'[[t[x/u]]]]) && = \\ & \mathbf{wj}(v)[-/\mathbf{wj}(C'[[t[x/u]]])] && \rightarrow_{\mathbf{h}, \mathbf{u}/\mathbf{o}}^* (i.h.) \\ & \mathbf{wj}(v)[-/\mathbf{wj}(C'[[t]])[-/\mathbf{wj}(u)]] && \rightarrow_{\mathbf{u}} \\ & \mathbf{wj}(v)[-/\mathbf{wj}(C'[[t]])[-/\mathbf{wj}(u)]] && = \mathbf{wj}(t_1) \end{aligned}$$

If $y \in \text{fv}(v)$, then

$$\begin{aligned}
\text{wj}(t_0) &= \text{wj}(v[y/C'[t[x/u]]]) && = \\
&\text{wj}(v)\{y/\text{wj}(C'[t[x/u]])\} && \rightarrow_{\text{h,u/o}}^* \text{ (i.h. \& L. 6.2)} \\
&\text{wj}(v)\{y/\text{wj}(C'[t])[-/\text{wj}(u)]\} && \rightarrow_{\text{h,u/o}}^* \text{ (L. 6.4)} \\
&\text{wj}(v)\{y/\text{wj}(C'[t])[-/\text{wj}(u)]\} && = \\
&\text{wj}(v[y/C'[t]][-/\text{wj}(u)]) && = \text{wj}(t_1)
\end{aligned}$$

– $t_0 = C'[t[x/u]][y/v] \rightarrow_u C'[t][y/v][x/u] = t_1$. Note that $y \notin \text{fv}(u)$, otherwise the rule cannot be applied. We reason by cases.

If $y \notin \text{fv}(C'[t])$, then

$$\begin{aligned}
\text{wj}(t_0) &= \text{wj}(C'[t[x/u]][y/v]) && = \\
&\text{wj}(C'[t[x/u]][-/\text{wj}(v)]) && \rightarrow_{\text{h,u/o}}^* \text{ (i.h.)} \\
&\text{wj}(C'[t])[-/\text{wj}(u)] [-/\text{wj}(v)] && \equiv_{\text{CS}} \\
&\text{wj}(C'[t])[-/\text{wj}(v)] [-/\text{wj}(u)] && = \text{wj}(t_1)
\end{aligned}$$

If $y \in \text{fv}(C'[t])$, then

$$\begin{aligned}
\text{wj}(t_0) &= \text{wj}(C'[t[x/u]][y/v]) && = \\
&\text{wj}(C'[t[x/u]])\{y/\text{wj}(v)\} && \rightarrow_{\text{h,u/o}}^* \text{ (i.h.)} \\
&\text{wj}(C'[t])[-/\text{wj}(u)]\{y/\text{wj}(v)\} && = \\
&\text{wj}(C'[t])\{y/\text{wj}(v)\}[-/\text{wj}(u)] && = \text{wj}(t_1)
\end{aligned}$$

then we conclude by using the i.h. and the equivalence \equiv_{CS} if $y \notin \text{fv}(C'[t])$.

- The inductive cases for the abstraction, the application and reduction inside substitution are straightforward.
- $t_0 = u_0[y/u_1] \rightarrow u'_0[y/u_1] = t_1$, where $u_0 \rightarrow_{\text{h}} u'_0$ (resp. $u_0 \rightarrow_u u'_0$).

If $y \in \text{fv}(u_0)$ & $y \in \text{fv}(u'_0)$ or $y \notin \text{fv}(u_0)$ & $y \notin \text{fv}(u'_0)$ then the property is straightforward by the i.h. So let us suppose $y \in \text{fv}(u_0)$ and $y \notin \text{fv}(u'_0)$ (so that the reduction step is necessarily a h-step). We have $\text{wj}(u_0) \rightarrow_{\text{h,u/o}}^+ \text{ (i.h.) } \text{wj}(u'_0)$ so that

$$\begin{aligned}
\text{wj}(t_0) &= \text{wj}(u_0)\{y/\text{wj}(u_1)\} && \rightarrow_{\text{h,u/o}}^+ \text{ (C. 6.6)} \\
&\text{wj}(u'_0)[-/\text{wj}(u_1)] && = \text{wj}(t_1)
\end{aligned}$$

□

Theorem 7.2. *The $\{\lambda_j, \text{h}, \text{u}\}/\text{obox}$ -calculus enjoys PSN, i.e. $t \in \mathcal{T}_\lambda \cap \mathcal{SN}_\beta$, then $t \in \mathcal{SN}_{\{\lambda_j, \text{h}, \text{u}\}/\text{obox}}$.*

Proof. We apply Theorem 2.3, where $\mathcal{A} = \lambda\text{void}$, $\mathcal{A}_1 = \{\text{d}, \text{c}, \text{u}\}$, $\mathcal{A}_2 = \{\text{dB}, \text{w}, \text{h}\}$, E is \equiv_{obox} , F is \equiv_{\circ} and $t \mathcal{R} \text{wj}(t)$. Property **(P0)** holds by Lemma 6.18:4-5, Property **(P1)** holds by Lemmas 6.18:3 and 7.1:2, Property **(P2)** holds by Lemmas 6.18:1-2 and 7.1:1. To show Property **(P3)** we proceed as follows. First of all notice that u/obox is trivially terminating, then show that $\mathcal{A}_1/\text{obox}$ is terminating by showing that $t \rightarrow_{\mathcal{A}_1/\text{obox}} t'$ implies $\langle \text{jm}(t), t \rangle > \langle \text{jm}(t'), t' \rangle$, where the first component of the pair is compared with respect to the multiset order and the second component of the pair is compared with respect to the terminating relation $\rightarrow_{\text{u/obox}}$. Now, take $t \in \mathcal{T}_\lambda \cap \mathcal{SN}_\beta$ so that Corollary 6.16 gives $t \in \mathcal{SN}_{\lambda\text{void/o}}$. Since $\text{wj}(t) = t$, then $t \in \mathcal{SN}_{\{\lambda_j, \text{h}, \text{u}\}/\text{obox}}$ by application of the Theorem. □

$$\begin{array}{lcl}
(\lambda x.t)L u & \mapsto_{\text{dB}} & t[x/u]L \\
t[x/u] & \mapsto_{\text{w}} & t & \text{if } |t|_x = 0 \\
t[x/u] & \mapsto_{\text{d}} & t\{x/u\} & \text{if } |t|_x = 1 \\
t[x/u] & \mapsto_{\text{c}} & t_{[y]_x}[x/u][y/u] & \text{if } |t|_x > 1 \\
\\
(\lambda y.t)[x/u] & \mapsto_{\text{in/CS}_1} & \lambda y.(t[x/u]) \\
(t v)[x/u] & \mapsto_{\text{in/CS}_2} & t[x/u] v & \text{if } x \notin \text{fv}(v) \\
(t v)[x/u] & \mapsto_{\text{in/CS}_3} & t v[x/u] & \text{if } x \notin \text{fv}(t) \ \& \ x \in \text{fv}(v) \\
t[y/v][x/u] & \mapsto_{\text{in/CS}_4} & t[y/v][x/u] & \text{if } x \notin \text{fv}(t) \ \& \ x \in \text{fv}(v) \\
\\
t[x/u][y/v] & \sim_{\text{CS}} & t[y/v][x/u] & \text{if } x \notin \text{fv}(v) \ \& \ y \notin \text{fv}(s)
\end{array}$$

Figure 7: The inner structural λ -calculus λj_{in}

$$\begin{array}{lcl}
(\lambda x.t) u & \rightarrow_{\text{B}} & t[x/u] \\
x[x/u] & \rightarrow_{\text{d}'} & u \\
t[x/u] & \rightarrow_{\text{w}} & t & \text{if } x \notin \text{fv}(t) \\
(t v)[x/u] & \rightarrow_{\text{@}_r} & t v[x/u] & \text{if } x \notin \text{fv}(t) \ \text{and } x \in \text{fv}(v) \\
(t v)[x/u] & \rightarrow_{\text{@}_l} & t[x/u] v & \text{if } x \in \text{fv}(t) \ \text{and } x \notin \text{fv}(v) \\
(t v)[x/u] & \rightarrow_{\text{@}} & t[x/u] v[x/u] & \text{if } x \in \text{fv}(t) \ \text{and } x \in \text{fv}(v) \\
(\lambda y.t)[x/u] & \rightarrow_{\lambda} & \lambda y.t[x/u] \\
t[x/u][y/v] & \rightarrow_{\text{comp}_1} & t[x/u][y/v] & \text{if } y \notin \text{fv}(t) \ \text{and } y \in \text{fv}(u) \\
t[x/u][y/v] & \rightarrow_{\text{comp}_2} & t[y/v][x/u][y/v] & \text{if } y \in \text{fv}(t) \ \text{and } y \in \text{fv}(u) \\
\\
t[x/u][y/v] & \sim_{\text{CS}} & t[y/v][x/u] & \text{if } y \notin \text{fv}(u) \ \text{and } x \notin \text{fv}(v) \\
& & & (\text{and } x \neq y)
\end{array}$$

Figure 8: The λes -calculus

7.2. Orienting the axioms of obox. Another interesting result concerns a more traditional form of explicit substitutions calculus, called here the **inner structural λ -calculus**, and noted λj_{in} , whose rules appear in Figure 7.

Let $\rightarrow_{\text{in/CS}}$ be the context closure of the rules $\mapsto_{\text{in/CS}_{1,2,3,4}}$ modulo \equiv_{CS} .

Lemma 7.3. *The reduction relation $\rightarrow_{\text{in/CS}}$ is strongly normalising.*

Proof. Define $M(t)$ to be the sum of all the sizes of the subterms of t directly affected by jumps. It is easily seen that such a measure strictly decreases by one-step rewriting and is invariant by \equiv_{CS} . \square

Corollary 7.4. *The inner structural λ -calculus λj_{in} enjoys PSN.*

Proof. By application of Theorem 2.3, where the required properties of the projection of λj_{in} into $\lambda j/\text{obox}$ are guaranteed by Lemmas 6.18 and 7.3. \square

The inner structural λ -calculus can be seen as a refinement of Kesner's λes [17], an explicit substitution calculus related to Proof-Nets, whose rules are in Figure 8.

Indeed, only rules $\{\text{@}, \text{comp}_2\}$ are not particular cases of rules of λj_{in} , but they can be decomposed by using duplication followed by propagations as follows:

$$\begin{array}{ccc}
(t v)[x/u] & \rightarrow_{\textcircled{a}} & t[x/u] v[x/u] \\
\downarrow_{\text{c}} & & \uparrow^{\text{in}/\text{CS}_3} \\
(t v\{x/y\})[x/u][y/u] & \rightarrow_{\text{in}/\text{CS}_2} & (t[x/u] v\{x/y\})[y/u] \equiv_{\alpha} (t[x/u] v)[x/u]
\end{array}$$

It is then straightforward to simulate λes inside $\lambda\text{j}_{\text{in}}$, so we get:

Corollary 7.5. *The λes -calculus enjoys PSN.*

The second author shows in [18] that from PSN of λes one can infer PSN of a wide range of calculi, $\lambda\mathbf{x}$, Kesner's λes and λesw [17], Milner's calculus λsub [30], David's and Guillaume's λ_{ws} [6], the calculus with director strings of [36]. Hence PSN for $\lambda\text{j}/\text{obox}$ encompasses most results of PSN in the literature of explicit substitutions.

The interesting feature of $\lambda\text{j}_{\text{in}}$ with respect to λes is that the propagation subsystem $\rightarrow_{\text{in}/\text{CS}}$ is not needed in order to compute a normal form. Propagations are rather (linear) re-arrangements of term constructors which may be used as the basis of some term transformations used for compilation or optimisation.

The strength of a splitting of the whole calculus into a core and propagation system lies in the fact that the latter can be changed without affecting the former. In particular, it is possible to orient the axioms $\{\sigma_1, \sigma_2, \text{box}_1, \text{box}_2\}$ in the opposite direction by getting the *outer* structural λ -calculus $\lambda\text{j}_{\text{out}}$, whose rules are in Figure 9.

Observe that in contrast to the inner calculus the outer box rules act also on void jumps, *i.e.* they are not just an orientation of the box equations, but an extension too. This is possible because - as we showed earlier (Theorem 7.2) - extending $\lambda\text{j}/\text{obox}$ with for void jumps is safe (while we do not know whether it is safe to extend $\lambda\text{j}/\text{obox}$ with boxing for void jumps). Let $\rightarrow_{\text{out}/\text{CS}}$ be the derived context closure of the outer rules $\mapsto_{\text{out}_{1,2,3,4}}$ modulo \equiv_{CS} .

Lemma 7.6. *The reduction relation $\rightarrow_{\text{out}/\text{CS}}$ is strongly normalising.*

Corollary 7.7. *The outer structural λ -calculus $\lambda\text{j}_{\text{out}}$ enjoys PSN.*

Proof. By application of Theorem 2.3, where the required properties of the projection of $\lambda\text{j}_{\text{out}}$ into $\lambda\text{j}/\text{obox}$ are guaranteed by Lemmas 6.18 and 7.6. \square

In fact, it is easily seen that no matter how the axioms $\{\sigma_1, \sigma_2, \text{box}_1, \text{box}_2\}$ are oriented that they get a terminating rewriting system. As for $\lambda\text{j}_{\text{in}}$ and $\lambda\text{j}_{\text{out}}$, PSN can also be proved for the remaining 14 derived calculi, even if it is not clear to what extent they would be interesting.

8. CONCLUSIONS

We have introduced the structural λj -calculus, a concise but expressive λ -calculus with jumps admitting graphical interpretations by means of λj -dags and Pure Proof-Nets. Even if λj has strong linear logic background, the calculus can be understood as a particular reduction system, based on the notion of multiplicity and reduction at a distance, and being independently from any logic or type system. We established different properties for λj such as confluence and PSN. Moreover, full composition holds without any need of structural composition nor commutation of jumps. The λj -calculus admits a graphical operational equivalence \equiv_{\circ} allowing to commute jumps with linear constructs. The relation \equiv_{\circ} can be naturally understood as Regnier's σ -equivalence on λ -terms and turns out to be

$$\begin{array}{lll}
(\lambda x.t)\mathbf{L} u & \mapsto_{\text{dB}} & t[x/u]\mathbf{L} \\
t[x/u] & \mapsto_{\text{w}} & t \quad \text{if } |t|_x = 0 \\
t[x/u] & \mapsto_{\text{d}} & t\{x/u\} \quad \text{if } |t|_x = 1 \\
t[x/u] & \mapsto_{\text{c}} & t_{[y]_x}[x/u][y/u] \quad \text{if } |t|_x > 1 \\
\\
\lambda y.(t[x/u]) & \mapsto_{\text{out}_1} & (\lambda y.t)[x/u] \quad \text{if } y \notin \text{fv}(u) \\
t[x/u] v & \mapsto_{\text{out}_2} & (t v)[x/u] \\
t v[x/u] & \mapsto_{\text{out}_3} & (t v)[x/u] \\
t[y/v][x/u] & \mapsto_{\text{out}_4} & t[y/v][x/u] \\
\\
t[x/u][y/v] & \sim_{\text{CS}} & t[y/v][x/u] \quad \text{if } x \notin \text{fv}(v) \ \& \ y \notin \text{fv}(s)
\end{array}$$

Figure 9: The outer structural λ -calculus $\lambda\mathbf{j}_{\text{out}}$

a strong bisimulation. Moreover, \equiv_{\circ} can be further extended to the substitution equivalence \equiv_{obox} allowing to commute also jumps and non-linear constructs. The resulting calculus enjoys PSN, a non-trivial result from which one derives several known PSN results.

PSN of $\lambda\mathbf{j}$ modulo \equiv_{obox} is shown by means of an auxiliary calculus $\lambda\text{void}/\text{o}$ which can be understood as a *memory* calculus specified by means of *void* substitutions. A memory calculus due to Klop [24] is often used for termination arguments. Its syntax is usually presented as follows:

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

where $x \in \text{fv}(t)$ for every term $\lambda x.t$ and the memory construct $[t, u]$ is used to collect in u the arguments of the erasing β -redexes. The rule associated to this calculus are:

$$\begin{array}{ll}
(\lambda x.t)u & \mapsto_{\beta} t\{x/u\} \\
[t, v] u & \mapsto_{\pi} [tu, v]
\end{array}$$

If one interprets $[t, v]$ as $t[-/v]$ then Klop's calculus can be mapped into $\lambda\text{void}/\text{o}$: β maps to β and π becomes the reduction rule $t[-/v] u \rightarrow (t u)[-/v]$, which is subsumed by the equation \equiv_{σ_2} of $\lambda\text{void}/\text{o}$. Indeed, $\lambda\text{void}/\text{o}$ is more expressive than Klop's calculus. We claim that $\lambda\text{void}/\text{o}$ is interesting on its own and can be used for proving termination results beyond those of this paper.

We do not know whether $\lambda\mathbf{j}/\text{obox}$ extended with unrestricted boxing, in contrast to $\lambda\mathbf{j}/\text{obox}$ extended with $\{\mathbf{h}, \mathbf{u}\}$ presented in Section 7.1, enjoys PSN. The point is delicate, indeed from the literature ([29]) we know that unrestricted boxing together with the following traditional explicit substitution rule (without side condition on x):

$$(t v)[x/u] \rightarrow_{\text{@}} t[x/u] v[x/u]$$

break PSN. Now, the rule $\rightarrow_{\text{@}}$ cannot be simulated in $\lambda\mathbf{j}/\text{obox}$, so it would be interesting to understand if $\lambda\mathbf{j}/\text{obox}$ plus unrestricted boxing enjoys PSN.

An interesting research direction would be to formalise the link between $\lambda\mathbf{j}$, linear logic and abstract machines. Indeed, in contrast to explicit substitution calculi, $\lambda\mathbf{j}$ naturally expresses the notion of linear head reduction [5], which relates in a simpler way to Krivine's Abstract Machine [26] and many of its variations such as the Zinc Abstract Machine [27]. This is because linear head reduction performs the minimal amount of substitutions necessary to find which occurrences of variables will stand in head positions. While this is not a reduction strategy in the usual sense of λ -calculus, it can

be seen as a clever way to implement β -reduction by means of proof-nets technology, which can be reformulated in the λ_j -calculus as a strategy.

The λ_j -calculus has been used [3] to specify XL-developments, a terminating notion of reduction generalising those of development [14] and superdevelopment [25]. It would be certainly interesting to go further into the understanding of such a notion.

It would be also interesting to exploit distance and multiplicities in other different frameworks dealing for example with pattern matching, continuations or differential features. A direction which seems particularly challenging is standardization for λ_j . It would be interesting in particular to obtain a notion of standard reduction which is stable by \equiv_{\circ} -equivalence (or at least \equiv_{CS} , so that the result would pass to λ_j -dags). Indeed, classical notions as leftmost-outermost reduction do not easily generalise to λ_j modulo \equiv_{\circ} where jumps can be swapped and permuted with linear constructors.

Acknowledgements. We would like to thank Stefano Guerrini for stimulating discussions.

REFERENCES

- [1] B. Accattoli. Jumping around the box: graphical and operational studies on Lambda Calculus and Linear Logic. Ph.D. Thesis, Università di Roma La Sapienza, 2011.
- [2] B. Accattoli and S. Guerrini. Jumping Boxes. representing Lambda-Calculus Boxes by Jumps. In E. Grädel and R. Kahle, editors, Proc. of 18th Computer Science Logic (CSL), volume 5771 of Lecture Notes in Computer Science, pages 55–70. Springer-Verlag, Sept. 2009.
- [3] B. Accattoli and D. Kesner. The structural lambda-calculus. In A. Dawar and H. Veith, editors, Proc. of 24th Computer Science Logic (CSL), volume 6247 of Lecture Notes in Computer Science, pages 381–395. Springer-Verlag, Aug. 2010.
- [4] R. Bloo and K. Rose. Preservation of strong normalization in named lambda calculi with explicit substitution and garbage collection. In Computing Science in the Netherlands, pages 62–72. NCSRF, 1995.
- [5] V. Danos and L. Regnier. Reversible, irreversible and optimal lambda-machines. Theoretical Computer Science, 227(1):79–97, 1999.
- [6] R. David and B. Guillaume. A lambda-calculus with explicit weakening and explicit substitution. Mathematical Structures in Computer Science, 11(1):169–206, 2001.
- [7] N. G. de Bruijn. Generalizing Automath by Means of a Lambda-Typed Lambda Calculus. In Mathematical Logic and Theoretical Computer Science, number 106 in Lecture Notes in Pure and Applied Mathematics, pages 71–92. Marcel Dekker, 1987.
- [8] R. Di Cosmo, D. Kesner, and E. Polonovski. Proof nets and explicit substitutions. Mathematical Structures in Computer Science, 13(3):409–450, 2003.
- [9] J. Espírito Santo. A note on preservation of strong normalisation in the λ -calculus. Theoretical Computer Science, 412(12-14):169–183, 2011.
- [10] J.-Y. Girard. Linear logic. Theoretical Computer Science, 50, 1987.
- [11] J.-Y. Girard. Geometry of interaction I: an interpretation of system F. Proc. of the Logic Colloquium, 88:221–260, 1989.
- [12] M. Hasegawa. Models of Sharing Graphs: A Categorical Semantics of let and letrec, volume Distinguished Dissertation Series. Springer-Verlag, 1999.
- [13] H. Herbelin and S. Zimmermann. An operational account of call-by-value minimal and classical lambda-calculus in "natural deduction" form. In P.-L. Curien, editor, Proc. of 9th Typed Lambda Calculus and Applications (TLCA), volume 5608 of Lecture Notes in Computer Science, pages 142–156. Springer-Verlag, July 2009.
- [14] J. R. Hindley. Reductions of residuals are finite. Transactions of the American Mathematical Society, 240:345–361, 1978.
- [15] G. Huet. Résolution d'équations dans les langages d'ordre 1, 2, . . . , ? Thèse de doctorat d'état, Université Paris VII, 1976.
- [16] F. Kamareddine. Postponement, conservation and preservation of strong normalization for generalized reduction. Journal of Logic and Computation, 10(5):721–738, 2000.

- [17] D. Kesner. The theory of calculi with explicit substitutions revisited. In J. Duparc and T. A. Henzinger, editors, Proc. of 16th Computer Science Logic (CSL), volume 4646 of Lecture Notes in Computer Science, pages 238–252. Springer-Verlag, Sept. 2007.
- [18] D. Kesner. A theory of explicit substitutions with safe and full composition. Logical Methods in Computer Science, 5(3:1):1–29, 2009.
- [19] D. Kesner and S. O. Conchúir. Milner’s lambda calculus with partial substitutions, 2008.
- [20] D. Kesner and S. Lengrand. Extending the explicit substitution paradigm. In J. Giesl, editor, 16th International Conference on Rewriting Techniques and Applications (RTA), volume 3467 of Lecture Notes in Computer Science, pages 407–422. Springer-Verlag, Apr. 2005.
- [21] D. Kesner and S. Lengrand. Resource operators for lambda-calculus. Information and Computation, 205(4):419–473, 2007.
- [22] D. Kesner and F. Renaud. The prismoid of resources. In R. Královic and D. Niwinski, editors, Proc. of the 34th Mathematical Foundations in Computer Science, volume 5734 of Lecture Notes in Computer Science, pages 464–476. Springer-Verlag, Aug. 2009.
- [23] A. J. Kfoury and J. B. Wells. New notions of reduction and non-semantic proofs of beta-strong normalization in typed lambda-calculi. In D. Kozen, editor, 10th Annual IEEE Symposium on Logic in Computer Science (LICS), pages 311–321. IEEE Computer Society Press, June 1995.
- [24] J.-W. Klop. Combinatory Reduction Systems, volume 127 of Mathematical Centre Tracts. Mathematisch Centrum, Amsterdam, 1980. PhD Thesis.
- [25] J.-W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: introduction and survey. Theoretical Computer Science, 121(1/2):279–308, 1993.
- [26] J.-L. Krivine. Un interpréteur du lambda-calcul. Available on <http://www.pps.jussieu.fr/~krivine/articles/>.
- [27] X. Leroy. The ZINC experiment: an economical implementation of the ML language. Technical report 117, INRIA, 1990.
- [28] J. Maraist, M. Odersky, D. N. Turner, and P. Wadler. Call-by-name, call-by-value, call-by-need and the linear lambda calculus. Theoretical Computer Science, 228(1-2):175–210, 1999.
- [29] P.-A. Mellès. Typed lambda-calculi with explicit substitutions may not terminate. In Proc. of 2nd Typed Lambda Calculus and Applications (TLCA), volume 902 of Lecture Notes in Computer Science, pages 328–334. Springer-Verlag, Apr. 1995.
- [30] R. Milner. Local Bigraphs and Confluence: Two Conjectures (extended abstract). Electronic Notes in Theoretical Computer Science, 175(3):65–73, 2007.
- [31] R. P. Nederpelt. The fine-structure of lambda calculus. Technical Report CSN 92/07, Eindhoven Univ. of Technology, 1992.
- [32] Y. Ohta and M. Hasegawa. A terminating and confluent linear lambda calculus. In F. Pfenning, editor, Rewriting Techniques and Applications (RTA), volume 4098 of Lecture Notes in Computer Science, pages 166–180. Springer-Verlag, 2006.
- [33] L. Regnier. Une équivalence sur les lambda-termes. Theoretical Computer Science, 2(126):281–292, 1994.
- [34] F. Renaud. Metaconfluence of λ_j : dealing with non-deterministic replacements. Available on <http://www.pps.jussieu.fr/~renaud>.
- [35] H. Schwichtenberg. Termination of permutative conversions in intuitionistic Gentzen calculi. Theoretical Computer Science, 212(1-2):247–260, 99.
- [36] F.-R. Sinot, M. Fernández, and I. Mackie. Efficient reductions with director strings. In R. Nieuwenhuis, editor, 14th International Conference on Rewriting Techniques and Applications (RTA), volume 2706 of Lecture Notes in Computer Science, pages 46–60. Springer-Verlag, June 2003.
- [37] N. Yoshida. Optimal reduction in weak-lambda-calculus with shared environments. In Proc. of Int. Conference on Functional Programming Languages and Computer Architecture, pages 243–252. ACM Press, June 1993.