Is the *se*-calculus strongly normalising?*

Fairouz Kamareddine[1], and Alejandro Ríos[2]

[1] Computing and Electrical Engineering, Heriot-Watt Univ., Riccarton, Edinburgh EH14 4AS, Scotland, fairouz@cee.hw.ac.uk

[2] Department of Computer Science, University of Buenos Aires, Pabellón I - Ciudad Universitaria (1428) Buenos Aires, Argentina, rios@dc.uba.ar

Abstract. The $\lambda\sigma$-calculus (cf. [1]) reflects in its choice of operators and rules the calculus of categorical combinators (cf. [3]). The main innovation of the $\lambda\sigma$-calculus is the division of terms in two sorts: sort `term` and sort `substitution`. λs_e departs from this style of explicit substitutions in two ways. First, it keeps the classical and unique sort `term` of the λ-calculus. Second, it does not use some of the categorical operators, especially those which are not present in the classical λ-calculus. The λs_e introduces two new operators which reflect the substitution and updating that are present in the meta-language of the λ-calculus, and so it can be said to be closer to the λ-calculus from an intuitive point of view, rather than a categorical one.
The λs_e-calculus, like the $\lambda\sigma$-calculus, simulates β-reduction, is confluent (on open terms[1]) [9] and does not preserve PSN [6]. However, although strong normalisation (SN) of the σ-calculus (the substitution calculus associated with the $\lambda\sigma$-calculus) has been established, it is still unkown whether strong normalisation of the s_e-calculus (the substitution calculus associated with the λs_e-calculus) holds. Only weak normalisation of the s_e-calculus is known so far. This note, is a discussion of the status of strong normalisation of the s_e-calculus. Basically we show that the set of rules s_e is the union of two disjoint sets of rules A and B which are both SN but this does not lead us anywhere as commutation does not hold and hence modularity cannot be used to obtain SN of s_e. In addition, the distribution elimination [13] and recursive path ordering methods are not applicable and we remain unsure whether s_e is actually SN or not.

Strong normalisation of subcalculi of s_e

The last 15 years have seen an explosion in explicit substitution calculi (see [10] for a survey). As far as we know, almost all of them satisfy the property that the underlying calculus of substitutions terminate. For the λs_e-calculus [9], this property remains unsolved. This paper is to pose this problem in the hope that it can generate interest as a termination problem which at least for curiosity, needs to be settled. The answer can go either way. On the one hand, although the $\lambda\sigma$-calculus does not have PSN, the σ-calculus itself is SN. On the other hand, could the loss of PSN in the λs_e-calculus be due to the non-SN of the s_e-calculus? Are there termination techniques that we still have not explored and that could help us settle this problem? We would like to find out.

Let us summarize first the main problems that we face when trying to establish SN for s_e.

Problem 1: Unable to use recursive path ordering By taking a quick look at the s_e-rules (see Definition 22), it becomes obvious that the unfriendly rules, with respect to SN, are *σ-σ-transition* and to a lesser extent *φ-σ-transition*. These rules prevent us from establishing an order on the set of operators in order to solve the normalisation problem with a recursive path ordering.

Problem 2: Unable to use Zantema's distribution elimination lemma The s_e-rules "look like" associative rules but unfortunately they are not; e.g. in *σ-σ-transition* one could think of the σ^j-operator distributing over the σ^i-operator, but it is not a "true" distribution: σ^j changes to σ^{j+1} when acting on the first term and to σ^{j-i+1} when acting on the second. This prevents us from using Zantema's distribution elimination method [13] to obtain SN.

* This work was carried out under EPSRC grants GR/K25014, GR/L15685 and GR/L36963.

[1] The λs_e-calculus is confluent on the whole set of open terms whereas $\lambda\sigma$ is confluent on the open terms without metavariables of sort `substitution` as is shown in [12].

Problem 3: Unable to use modularity Another technique to show SN is modularity, i.e. establish SN for certain subcalculi and afterwards prove that these subcalculi satisfy a commutation property to conclude SN for the whole calculus. At the end of this note we will come back to this point and show that the necessary commutation results do not hold.

Let us say here that, even if σ-σ-*transition* seems responsible for the difficulties in establishing SN, Zantema succeded in establishing that the σ-σ-*transition* scheme on its own is SN (personal communication cited in [9]). Here we shall go a step further: we shall prove that σ-σ-*tr.*+φ-σ-*tr.* is SN and also that the calculus obtained with the rest of the rules is SN as well.

In this note we shall frequently use the following nomenclature:

Definition 1 *We define the following sets of rules:*
$*\varphi = \{\sigma\text{-}\varphi\text{-}tr.1,\ \sigma\text{-}\varphi\text{-}tr.2,\ \varphi\text{-}\varphi\text{-}tr.1,\ \varphi\text{-}\varphi\text{-}tr.2\}$,
$*\sigma = \{\sigma\text{-}\sigma\text{-}tr.,\ \varphi\text{-}\sigma\text{-}tr.\}$,
$*\varphi^{-} = \{\sigma\text{-}\varphi\text{-}tr.1,\ \varphi\text{-}\varphi\text{-}tr.2\}$, $*\varphi^{--} = \{\sigma\text{-}\varphi\text{-}tr.2,\ \varphi\text{-}\varphi\text{-}tr.1\}$.

Note that $s_e = (s + *\varphi) + *\sigma$. We shall prove in this note that both calculi generated by the set of rules $s + *\varphi$ (Theorem 4) and $*\sigma$ (Theorem 11) are SN. Unfortunately, these calculi do not possess the property of commutation needed to ensure that their union s_e is SN (see Example 14).

It is not difficult to prove that $s + *\varphi$ is SN by giving a weight that decreases through reduction. We begin by defining two weight functions we will need for the final weight:

Definition 2 *Let $P : \Lambda s_{op} \to I\!N$ and $W : \Lambda s_{op} \to I\!N$ be defined inductively by:*

$P(X) = P(\mathtt{n}) = 2$	$W(X) = W(\mathtt{n}) = 1$
$P(a\,b) = P(a) + P(b)$	$W(a\,b) = W(a) + W(b) + 1$
$P(\lambda a) = P(a)$	$W(\lambda a) = W(a) + 1$
$P(a\,\sigma^j b) = j * P(a) * P(b)$	$W(a\,\sigma^j b) = 2 * W(a) * (W(b) + 1)$
$P(\varphi^i_k a) = (k+1) * (P(a) + 1)$	$W(\varphi^i_k a) = 2 * W(a)$

Lemma 3 *For $a, b \in \Lambda s_{op}$ the following hold:*
1. *If $a \to_{s+*\varphi} b$ then $W(a) \geq W(b)$.*
2. *If $a \to_{s+*\varphi^-} b$ then $W(a) > W(b)$.*
3. *If $a \to_{*\varphi^{--}} b$ then $P(a) > P(b)$.*

PROOF: By induction on a: if the reduction is internal, the IH applies; otherwise, the theorem must be checked for each rule. ⊠

An immediate consequence of the previous lemma is:

Theorem 4 *The $s + *\varphi$-calculus is SN.*

PROOF: The previous lemma ensures that the ordinal $(W(a), P(a))$ decreases with the lexicographical order for each $s + *\varphi$-reduction. ⊠

Now, to prove SN for $*\sigma$ we are going to use the isomorphism presented in the appendix and the technique that Zantema used to prove SN for the calculus whose only rule is σ-σ-*transition* (cf. [9]). Following this isomorphism, the schemes σ-σ-*tr.* and φ-σ-*tr.* of λs_e both translate into the same scheme of $\lambda\omega_e$, namely σ-/-*transition* of Definition 28.

Zantema uses the following lemma (cf. [11]):

Lemma 5 *Any reduction relation $\to$ on a set T satisfying 1,2, and 3 is strongly normalising:*
1. *$\to$ is weakly normalising.*
2. *$\to$ is locally confluent.*
3. *$\to$ is increasing, i.e., $\exists$ a function $f : T \longrightarrow I\!N$ where $a \to b \Rightarrow f(a) < f(b)$.*

We use the previous lemma to prove that the calculus whose only rule is σ-/-*transition*, let us call it σ-/-calculus, is strongly normalising. For the σ-/ $-$ *calculus*, 2 follows from a simple critical pair analysis and 3 can be easily established by choosing $f(a)$ to be the size of a. To show weak normalisation of the σ-/ $-$ *calculus* the technique used by Zantema (cf. [9]) can be adapted here:

Definition 6 *We say that $c \in \Lambda\omega^t$ is an* external normal form *if $c = a[s_1]_{i_1} \cdots [s_n]_{i_n}$ where $a \neq c[d/]_k$ and if $s_k = b_k/$ then $i_k > i_{k+1}$. We denote the set of external normal forms ENF.*

Lemma 7 *Let $c = a[s_1]_{i_1} \cdots [s_n]_{i_n} \in ENF$ and let $i_n \leq i_{n+1}$ and $s_n = b_n/$ then there exists a σ-/-derivation $c \to^+ a[t_1]_{j_1} \cdots [t_{n+1}]_{j_{n+1}} \in ENF$ such that $j_{n+1} = i_n$ and for every r with $1 \leq r \leq n+1$ we have either $t_r = s_k$ for some $k \leq n+1$ or $t_r = (a_p[s_{n+1}])/$ for some $s_p = a_p/$ with $1 \leq p \leq n$.*

PROOF: By induction on n. ⊠

Lemma 8 *Let $c = a[s_1]_{i_1} \cdots [s_n]_{i_n}$ such that $a \neq c[d/]_k$. There exists a σ-/-derivation $c \twoheadrightarrow a[t_1]_{j_1} \cdots [t_n]_{j_n} \in ENF$ such that for every r with $1 \leq r \leq n+1$ we have either $t_r = s_k$ for some $k \leq n$ or $t_r = (a_{p_{r\,1}}[s_{p_{r\,2}}]_{k_2} \cdots [s_{p_{r\,n}}]_{k_n})/$ with $1 \leq p_{r\,1} \leq \cdots \leq p_{r\,n} \leq n$ and with some $s_p = a_{p_{r\,1}}/$ $(1 \leq p \leq n)$.*

PROOF: By induction on n, using the previous lemma. ⊠

Lemma 9 *The σ-/-calculus is weakly normalising.*

PROOF: Suppose there is a term c not having a normal form for which every term smaller (in size) than c admits a normal form. Let $c = a[s_1]_{i_1} \cdots [s_n]_{i_n}$ such that $a \neq c[d/]_k$. Applying Lemma 8, we get $c \to \to a[t_1]_{j_1} \cdots [t_n]_{j_n} \in ENF$. Note that a, t_1, $\cdots$ t_n are all smaller than c and hence admit a normal form. Now replacing each of them by its normal form in $a[t_1]_{j_1} \cdots [t_n]_{j_n}$ we have a normal form for c which is a contradiction. ⊠

Therefore we can finally apply Lemma 5 to conclude:

Theorem 10 *The σ-/-calculus is strongly normalising on $\Lambda\omega^t$.*

Now, using the isomorphism, since, as we mentioned before, both rule schemes in $*\sigma$ translate into the single σ-/ rule scheme, we have:

Theorem 11 *The $*\sigma$-calculus is strongly normalising.*

Now that $s + *\varphi$ and $*\sigma$ have been proved SN the question arises whether the whole system can be proved SN using a modularity result. The answer is negative for the classical modularity theorem of Bachmair-Dershowitz, which we recall here:

Definition 12 *A rewrite relation R commutes over S if whenever $a \to_S b \to_R c$, there is an alternative derivation $a \to_R d \to_{R \cup S} c$.*

Theorem 13 (Bachmair-Dershowitz-85) *Let R commute over S. The combined system $R \cup S$ is SN iff R and S both are SN.*

The following example shows that no commutation is possible between $s + *\varphi$ and $*\sigma$ and therefore the Bachmair-Dershowitz's Theorem cannot be applied to get SN for s_e.

Example 14 *Now, here is an example which shows that $*\sigma$ does not commute over $s + *\varphi$: Let $k + i \leq j$, $h \leq j - i + 1$ and $h > k + 1$. Let us consider the following derivation:*

$$(\varphi^i_k(a\,\sigma^h b))\,\sigma^j c \to_{*\varphi} \varphi^i_k((a\,\sigma^h b)\,\sigma^{j-i+1} c) \to_{\sigma-\sigma-tr} \varphi^i_k((a\,\sigma^{j-i+2} c)\,\sigma^h (b\,\sigma^{j-i-h+2} c))$$

But it is easy to see that $(\varphi^i_k(a\,\sigma^h b))\,\sigma^j c$ does not contain any $*\sigma$-redex.

On the other hand, $s + *\varphi$ does not commute over $*\sigma$ either: Let $i \leq j$ and let us consider the following derivation:

$$((\lambda a)\,\sigma^i b)\,\sigma^j c) \to_{\sigma-\sigma-tr} ((\lambda a)\,\sigma^{j+1} c)\,\sigma^i (b\,\sigma^{j-i+1} c) \to_s (\lambda(a\,\sigma^{j+2} c))\,\sigma^i (b\,\sigma^{j-i+1} c)$$

But reducing the only s-redex in $((\lambda a)\,\sigma^i b)\,\sigma^j c)$ we get $(\lambda(a\,\sigma^{i+1} b))\,\sigma^j c$ which also has a unique s-redex. Reducing it we get $\lambda((a\,\sigma^{i+1} b)\,\sigma^{j+1} c)$ and now there is only the σ-σ-transition redex, whose reduction gives us $\lambda((a\,\sigma^{j+2} c)\sigma^{i+1}(b\,\sigma^{j-i+1} c))$ which has no further redexes. Therefore, $(\lambda(a\,\sigma^{j+2} c))\,\sigma^i (b\,\sigma^{j-i+1} c)$ cannot be reached from $((\lambda a)\,\sigma^i b)\,\sigma^i c)$ with an s_e-derivation beginning with an s-step.

References

1. M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit Substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.
2. Z. Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. λv, a calculus of explicit substitutions which preserves strong normalisation. *Functional Programming*, 6(5), 1996.
3. P.-L. Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming.* Pitman, 1986. Revised edition : Birkhäuser (1993).
4. P.-L. Curien, T. Hardin, and J.-J. Lévy. Confluence properties of weak and strong calculi of explicit substitutions. Technical Report RR 1617, INRIA, Rocquencourt, 1992.
5. N. de Bruijn. Lambda-Calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser Theorem. *Indag. Mat.*, 34(5):381–392, 1972.
6. B. Guillaume. *Un calcul des substitutions avec etiquettes.* PhD thesis, Université de Savoie, Chambéry, France, 1999.
7. T. Hardin and A. Laville. Proof of Termination of the Rewriting System SUBST on CCL. *Theoretical Computer Science*, 46:305–312, 1986.
8. F. Kamareddine and A. Ríos. A λ-calculus à la de Bruijn with explicit substitutions. Proceedings of PLILP'95. *LNCS*, 982:45–62, 1995.
9. F. Kamareddine and A. Ríos. Extending a λ-calculus with explicit substitution which preserves strong normalisation into a confluent calculus on open terms. *Journal of Functional Programming*, 7(4):395–420, 1997.
10. F. Kamareddine and Alejandro Ríos. Relating the $\lambda\sigma$- and λs-styles of explicit substitutions. *Logic and Computation*, 10(3):349–380, 2000.
11. J.-W. Klop. Term rewriting systems. *Handbook of Logic in Computer Science*, II, 1992.
12. A. Ríos. *Contribution à l'étude des λ-calculs avec substitutions explicites.* PhD thesis, Université de Paris 7, 1993.
13. H. Zantema. Termination of term rewriting: interpretation and type elimination. *J. Symbolic Computation*, 17(1):23–50, 1994.

A Formal Machinery

A.1 The classical λ-calculus in de Bruijn notation

We assume the reader familiar with de Bruijn notation (cf. [5]). We define Λ, the *set of terms with de Bruijn indices*, by: $\Lambda ::= \mathbb{N} \mid (\Lambda\Lambda) \mid (\lambda\Lambda)$.

We use $a, b, \ldots$ to range over Λ and $m, n, \ldots$ to range over $\mathbb{N}$ (positive natural numbers). Furthermore, we assume the usual conventions about parentheses and avoid them when no confusion occurs. Throughout the whole article, $a = b$ is used to mean that a and b are syntactically identical, and write $\to^+$ and $\twoheadrightarrow$ to denote the transitive and the reflexive transitive closures of a reduction notion $\to$. We say that a reduction $\to$ is *compatible on* Λ when for all a, b, $c \in \Lambda$, we have $a \to b$ implies $a\,c \to b\,c$, $c\,a \to c\,b$ and $\lambda a \to \lambda b$.

In order to define β-reduction à la de Bruijn, we must define the substitution of a variable $\mathtt{n}$ for a term b in a term a. Therefore, we need to update the term b:

Definition 15 *The* updating functions $U_k^i : \Lambda \to \Lambda$ *for* $k \geq 0$ *and* $i \geq 1$ *are defined inductively:*

$$U_k^i(ab) = U_k^i(a)\,U_k^i(b) \qquad U_k^i(\mathtt{n}) = \begin{cases} \mathtt{n}+\mathtt{i}-1 & \text{if } n > k \\ \mathtt{n} & \text{if } n \leq k\,. \end{cases}$$

$$U_k^i(\lambda a) = \lambda(U_{k+1}^i(a))$$

Now we define the family of meta-substitution functions:

Definition 16 *The* meta-substitutions at level j, *for* $j \geq 1$, *of a term* $b \in \Lambda$ *in a term* $a \in \Lambda$, *denoted* $a\{\!\{\mathtt{j} \leftarrow b\}\!\}$, *is defined inductively on* a *as follows:*

$$(a_1 a_2)\{\!\{\mathtt{j} \leftarrow b\}\!\} = (a_1\{\!\{\mathtt{j} \leftarrow b\}\!\})\,(a_2\{\!\{\mathtt{j} \leftarrow b\}\!\}) \qquad \mathtt{n}\{\!\{\mathtt{j} \leftarrow b\}\!\} = \begin{cases} \mathtt{n}-1 & \text{if } n > j \\ U_0^j(b) & \text{if } n = j \\ \mathtt{n} & \text{if } n < j\,. \end{cases}$$

$$(\lambda a)\{\!\{\mathtt{j} \leftarrow b\}\!\} = \lambda(a\{\!\{\mathtt{j}+1 \leftarrow b\}\!\})$$

Definition 17 β-reduction *is the least compatible reduction on* Λ *generated by:*

(β-rule) $\qquad (\lambda a)\,b \to_\beta a\{\!\{\mathtt{1} \leftarrow b\}\!\}$

The λ-calculus (à la de Bruijn), *is the reduction system whose only rewriting rule is* β.

A.2 The λs- and λs_e-calculi

The idea of λs is to handle explicitly the meta-operators given in definitions 15 and 16. Therefore, the syntax of the λs-calculus is obtained by adding two families of operators :

- $\{\sigma^j\}_{j\geq 1}$, which denotes the explicit substitution operators. Each σ^j is an infix operator of arity 2 and $a\,\sigma^j b$ has as intuitive meaning the term a where all free occurrences of the variable corresponding to the de Bruijn index j are to be substituted by the term b.
- $\{\varphi_k^i\}_{k\geq 0\ i\geq 1}$, which denotes the updating functions necessary when working with de Bruijn numbers to fix the variables of the term to be substituted.

Definition 18 *The* set of terms, *noted* Λs , of the λs-calculus *is given as follows:*

$$\Lambda s ::= I\!N \mid \Lambda s \Lambda s \mid \lambda \Lambda s \mid \Lambda s\,\sigma^j \Lambda s \mid \varphi_k^i \Lambda s \quad \textit{where} \quad j, i \geq 1\,, \quad k \geq 0\,.$$

We take a, b, c *to range over* Λs*. A term of the form* $a\,\sigma^j b$ *is called a* closure*. Furthermore, a term containing neither* σ*'s nor* φ*'s is called a* pure term*.* Λ *denotes the set of pure terms.*

A compatible reduction *on* Λs *is a reduction* $\rightarrow$ *such that for all* a, b, $c \in \Lambda s$, *if* $a \rightarrow b$ *then* $a\,c \rightarrow b\,c$, $c\,a \rightarrow c\,b$, $\lambda a \rightarrow \lambda b$, $a\,\sigma^j c \rightarrow b\,\sigma^j c$, $c\,\sigma^j a \rightarrow c\,\sigma^j b$ *and* $\varphi_k^i a \rightarrow \varphi_k^i b$.

We include, besides the rule mimicking the β-rule (σ-*generation*), a set of rules which are the equations in definitions 15 and 16 oriented from left to right.

Definition 19 *The* λs-calculus *is the reduction system* $(\Lambda s, \rightarrow_{\lambda s})$*, where* $\rightarrow_{\lambda s}$ *is the least compatible reduction on* Λs *generated by the following rules:*

σ-*generation*	$(\lambda a)\,b \longrightarrow a\,\sigma^1\,b$
σ-λ-*transition*	$(\lambda a)\,\sigma^j b \longrightarrow \lambda(a\sigma^{j+1}b)$
σ-*app-transition*	$(a_1\,a_2)\,\sigma^j b \longrightarrow (a_1\,\sigma^j b)\,(a_2\,\sigma^j b)$
σ-*destruction*	$\mathtt{n}\,\sigma^j b \longrightarrow \begin{cases} \mathtt{n-1} & \textit{if} \quad n > j \\ \varphi_0^j\,b & \textit{if} \quad n = j \\ \mathtt{n} & \textit{if} \quad n < j \end{cases}$
φ-λ-*transition*	$\varphi_k^i(\lambda a) \longrightarrow \lambda(\varphi_{k+1}^i\,a)$
φ-*app-transition*	$\varphi_k^i(a_1\,a_2) \longrightarrow (\varphi_k^i\,a_1)\,(\varphi_k^i\,a_2)$
φ-*destruction*	$\varphi_k^i\,\mathtt{n} \longrightarrow \begin{cases} \mathtt{n+i-1} & \textit{if} \quad n > k \\ \mathtt{n} & \textit{if} \quad n \leq k \end{cases}$

We use λs *to denote this set of rules. The* s-calculus*, the* calculus of substitutions associated with the λs-calculus*, is the reduction system generated by the set of rules* $s = \lambda s - \{\sigma\text{-}generation\}$.

Lemma 20 (cf. [8]) *The following holds:*

1. *(SN and CR of* s*) The* s*-calculus is strongly normalising and confluent on* Λs*. Hence, every term* a *has a unique* s*-normal form denoted* $s(a)$*.*
2. *The set of* s*-normal forms is exactly* Λ*.*
3. *For all* $a, b \in \Lambda s$ *we have:*
   $s(a\,b) = s(a)s(b)\,, \quad s(\lambda a) = \lambda(s(a))\,, \quad s(\varphi_k^i a) = U_k^i(s(a))\,, \quad s(a\,\sigma^j b) = s(a)\{\!\{\mathtt{j} \leftarrow s(b)\}\!\}$.
4. *Let* $a, b \in \Lambda s$ *, if* $a \rightarrow_{\sigma-gen} b$ *or* $a \twoheadrightarrow_{\lambda s} b$ *then* $s(a) \twoheadrightarrow_\beta s(b)$ *.*
5. *(Soundness) Let* $a, b \in \Lambda$ *, if* $a \twoheadrightarrow_{\lambda s} b$ *then* $a \twoheadrightarrow_\beta b$ *.*
6. *(Simulation of* β*-reduction) Let* $a, b \in \Lambda$*, if* $a \rightarrow_\beta b$ *then* $a \twoheadrightarrow_{\lambda s} b$ *.*
7. *(CR of* λs*) The* λs*-calculus is confluent on* Λs*.*
8. *(Preservation of SN) Pure terms which are strongly normalising in the* λ*-calculus are also strongly normalising in the* λs*-calculus.*

Open terms were introduced in the λs-calculus as follows (see [9]):

Definition 21 *The set of* open terms, *noted* Λs_{op} *is given as follows:*

$$\Lambda s_{op} ::= \mathbf{V} \mid I\!N \mid \Lambda s_{op}\Lambda s_{op} \mid \lambda\Lambda s_{op} \mid \Lambda s_{op}\,\sigma^j\Lambda s_{op} \mid \varphi^i_k\Lambda s_{op} \quad \textit{where} \quad j, i \geq 1\,, \;\; k \geq 0$$

and where $\mathbf{V}$ *stands for a set of variables, over which* X, Y, ... *range. We take* a, b, c *to range over* Λs_{op}. *Furthermore,* closures, pure terms *and* compatibility *are defined as for* Λs.

Working with open terms one loses confluence as shown by the following counterexample:

$$((\lambda X)Y)\sigma^1\mathtt{1} \to (X\sigma^1 Y)\sigma^1\mathtt{1} \qquad ((\lambda X)Y)\sigma^1\mathtt{1} \to ((\lambda X)\sigma^1\mathtt{1})(Y\sigma^1\mathtt{1})$$

and $(X\sigma^1 Y)\sigma^1\mathtt{1}$ and $((\lambda X)\sigma^1\mathtt{1})(Y\sigma^1\mathtt{1})$ have no common reduct. Moreover, the above example shows that even local confluence is lost. In order to solve this problem, [9] added to the λs-calculus a set of rules that guarantees confluence:

Definition 22 *The set of rules* λs_e *is obtained by adding the rules given below to the set* λs.

σ-σ-*transition*	$(a\,\sigma^i b)\,\sigma^j\,c \longrightarrow (a\,\sigma^{j+1}\,c)\;\sigma^i\,(b\,\sigma^{j-i+1}\,c)$	*if* $i \leq j$
σ-φ-*transition 1*	$(\varphi^i_k\,a)\,\sigma^j\,b \longrightarrow \varphi^{i-1}_k\,a$	*if* $k < j < k+i$
σ-φ-*transition 2*	$(\varphi^i_k\,a)\,\sigma^j\,b \longrightarrow \varphi^i_k\,(a\,\sigma^{j-i+1}\,b)$	*if* $k+i \leq j$
φ-σ-*transition*	$\varphi^i_k(a\,\sigma^j\,b) \longrightarrow (\varphi^i_{k+1}\,a)\,\sigma^j\,(\varphi^i_{k+1-j}\,b)$	*if* $j \leq k+1$
φ-φ-*transition 1*	$\varphi^i_k\,(\varphi^j_l\,a) \longrightarrow \varphi^j_l\,(\varphi^i_{k+1-j}\,a)$	*if* $l+j \leq k$
φ-φ-*transition 2*	$\varphi^i_k\,(\varphi^j_l\,a) \longrightarrow \varphi^{j+i-1}_l\,a$	*if* $l \leq k < l+j$

The λs_e-calculus *is the reduction system* $(\Lambda s_{op}, \to_{\lambda s_e})$ *where* $\to_{\lambda s_e}$ *is the least compatible reduction on* Λs_{op} *generated by the set of rules* λs_e. *The* calculus of substitutions associated with the λs_e-calculus *is the rewriting system generated by the set of rules* $s_e = \lambda s_e - \{\sigma\text{-generation}\}$ *and we call it* s_e-calculus.

Lemma 23 (See [9]) *The following holds:*

1. *(WN and CR of* s_e*) The* s_e*-calculus is weakly normalising and confluent.*
2. *(Simulation of* β*-reduction) Let* $a, b \in \Lambda$, *if* $a \to_\beta b$ *then* $a \twoheadrightarrow_{\lambda s_e} b$.
3. *(CR of* λs_e*) The* λs_e*-calculus is confluent on open terms.*
4. *(Soundness) Let* $a, b \in \Lambda$, *if* $a \twoheadrightarrow_{\lambda s_e} b$ *then* $a \twoheadrightarrow_\beta b$.

A.3 The $\lambda\sigma$-calculus

The $\lambda\sigma$-calculus [1] is a formalism which enables explicit substitution. Its syntax is two-sorted: the sort `term` of *terms* and the sort `substitution` of *explicit substitutions*. These can be interpreted as a sequence of terms and the result of executing a substitution in a term can be interpreted as the term obtained by replacing the occurrences of the n-th index of de Bruijn in the term by the n-th term of the sequence. This intuitive interpretation is developped and illustrated with many examples in [1]. Here are the syntax and the rules of the $\lambda\sigma$-calculus:

Definition 24 *The syntax of the* $\lambda\sigma$-calculus *is given by:*

Terms $\quad \Lambda\sigma^t ::= \mathtt{1} \mid \Lambda\sigma^t\Lambda\sigma^t \mid \lambda\Lambda\sigma^t \mid \Lambda\sigma^t[\Lambda\sigma^s]$
Substitutions $\quad \Lambda\sigma^s ::= id \mid \uparrow \mid \Lambda\sigma^t \cdot \Lambda\sigma^s \mid \Lambda\sigma^s \circ \Lambda\sigma^s$

The set, denoted $\lambda\sigma$, *of rules of the* $\lambda\sigma$-calculus *is the following:*

(Beta)	$(\lambda a)\, b \longrightarrow a\,[b \cdot id]$
(VarId)	$\mathtt{1}\,[id] \longrightarrow \mathtt{1}$
(VarCons)	$\mathtt{1}\,[a \cdot s] \longrightarrow a$
(App)	$(a\, b)[s] \longrightarrow (a\,[s])\,(b\,[s])$
(Abs)	$(\lambda a)[s] \longrightarrow \lambda(a\,[\mathtt{1} \cdot (s \circ \uparrow)])$
(Clos)	$(a\,[s])[t] \longrightarrow a\,[s \circ t]$
(IdL)	$id \circ s \longrightarrow s$
(ShiftId)	$\uparrow \circ id \longrightarrow \uparrow$
(ShiftCons)	$\uparrow \circ (a \cdot s) \longrightarrow s$
(Map)	$(a \cdot s) \circ t \longrightarrow a\,[t] \cdot (s \circ t)$
(Ass)	$(s_1 \circ s_2) \circ s_3 \longrightarrow s_1 \circ (s_2 \circ s_3)$

The set of rules of the σ-calculus *is* $\lambda\sigma - \{(Beta)\}$ *. We use* $a, b, c, \ldots$ *to range over* $\Lambda\sigma^t$ *and* $s, t, \ldots$ *to range over* $\Lambda\sigma^s$*. For every substitution* s *we define the* iteration of the composition of s *inductively as* $s^1 = s$ *and* $s^{n+1} = s \circ s^n$*. We use the convention* $s^0 = id$ *. Note that the only de Bruijn index used is* $\mathtt{1}$ *, but we can code* $\mathtt{n}$ *by the term* $\mathtt{1}[\uparrow^{n-1}]$ *. By so doing, we have* $\Lambda \subset \Lambda\sigma^t$ *.*

Lemma 25 *(cf. [1, 7])*

1. *The* σ*-calculus is strongly normalising (SN) and confluent (CR).*
2. *The* $\lambda\sigma$*-calculus is confluent.*

It is well known that the $\lambda\sigma$-calculus is not confluent on open terms, furthermore it is not even locally confluent. To obtain local confluence four rules must be added, and the calculus thus obtained is called the $\lambda\sigma_{SP}$-calculus.

Definition 26 *The* $\lambda\sigma_{SP}$*-calculus is obtained by adding to* $\lambda\sigma$ *the rules given below and by deleting the rules (VarId) and (ShiftId), since both of them are instances of the new rules.*

(Id)	$a[id] \longrightarrow a$
(IdR)	$s \circ id \longrightarrow s$
(VarShift)	$\mathtt{1} \cdot \uparrow \longrightarrow id$
(SCons)	$\mathtt{1}[s] \cdot (\uparrow \circ s) \longrightarrow s$

Even the $\lambda\sigma_{SP}$-calculus is not confluent on open terms (terms which admit metavariables of both sorts), as shown in [4], but it is confluent when the set of open terms is restricted to those which admit metavariables of sort `term` only [12].

A.4 The $\lambda\omega$- and $\lambda\omega_e$-calculi

In order to express λs-terms in the $\lambda\sigma$-style, [10] split the closure operator of $\lambda\sigma$ (denoted in a semi-infix notation as $-[-]$) in a family of closures operators that were denoted also with a semi-infix notation as $-[-]_i$, where i ranges on the set of natural numbers. [10] also admitted as basic operators the iterations of $\uparrow$ and therefore had a countable set of basic substitutions $\uparrow^n$, where n ranges on the set of natural numbers. By doing so, the updating operators of λs become available as $-[\uparrow^n]_i$. Finally, [10] introduced a *slash* operator of sort $\mathtt{term} \to \mathtt{substitution}$ which transforms a term a into a substitution $a/$. This operator may be considered as *consing with id* (in the $\lambda\sigma$-jargon) and was first introduced and exploited in the λv-calculus (cf. [2]). Here is the formalisation of this syntax and the rewriting rules of $\lambda\omega$:

Definition 27 *The set of terms of the $\lambda\omega$-calculus, noted $\Lambda\omega$, is defined as $\Lambda\omega^t \cup \Lambda\omega^s$, where $\Lambda\omega^t$ and $\Lambda\omega^s$ are mutually defined as follows:*

$$\begin{array}{ll} \textbf{Terms} & \Lambda\omega^t ::= \mathbb{N} \mid \Lambda\omega^t\Lambda\omega^t \mid \lambda\Lambda\omega^t \mid \Lambda\omega^t[\Lambda\omega^s]_j \\ \textbf{Substitutions} & \Lambda\omega^s ::= \uparrow^i \mid \Lambda\omega^t/ \end{array}$$

*where $j \geq 1$ and $i \geq 0$. The set, denoted $\lambda\omega$, of rules of the $\lambda\omega$-*calculus *is given as follows:*

$$\begin{array}{ll} \sigma\text{-generation} & (\lambda a)\, b \longrightarrow a\,[b/]_1 \\ \sigma\text{-app-transition} & (a\,b)[s]_j \longrightarrow (a\,[s]_j)\,(b\,[s]_j) \\ \sigma\text{-}\lambda\text{-transition} & (\lambda a)[s]_j \longrightarrow \lambda(a[s]_{j+1}) \\ \sigma\text{-}/\text{-destruction} & \mathtt{n}[a/]_j \longrightarrow \begin{cases} \mathtt{n-1} & \text{if } n > j \\ a[\uparrow^{j-1}]_1 & \text{if } n = j \\ \mathtt{n} & \text{if } n < j \end{cases} \\ \sigma\text{-}\uparrow\text{-destruction} & \mathtt{n}[\uparrow^i]_j \longrightarrow \begin{cases} \mathtt{n+i} & \text{if } n \geq j \\ \mathtt{n} & \text{if } n < j \end{cases} \end{array}$$

The set of rules of the ω-calculus *is $\lambda\omega - \{\sigma - generation\}$. We use $a, b, c, \ldots$ to range over $\Lambda\omega^t$ and $s, t, \ldots$ to range over $\Lambda\omega^s$.*

Definition 28 *The set of* open terms*, noted $\Lambda\omega_{op}$ is defined as $\Lambda\omega^t_{op} \cup \Lambda\omega^s_{op}$, where $\Lambda\omega^t_{op}$ and $\Lambda\omega^s_{op}$ are mutually defined as follows:*

$$\begin{array}{ll} \textbf{Open Terms} & \Lambda\omega^t_{op} ::= \mathbf{V} \mid \mathbb{N} \mid \Lambda\omega^t_{op}\Lambda\omega^t_{op} \mid \lambda\Lambda\omega^t_{op} \mid \Lambda\omega^t_{op}[\Lambda\omega^s_{op}]_j \\ \textbf{Substitutions} & \Lambda\omega^s_{op} ::= \uparrow^i \mid \Lambda\omega^t_{op}/ \end{array}$$

where $j \geq 1$ and $i \geq 0$, and where $\mathbf{V}$ stands for a set of variables, over which $X, Y, \ldots$ range. We take a, b, c to range over $\Lambda\omega^t_{op}$ and $s, t, \ldots$ over $\Lambda\omega^s_{op}$. Furthermore, closures, pure terms *and* compatibility *are defined as for $\Lambda\omega$. The set, denoted $\lambda\omega_e$, of rules of the $\lambda\omega_e$-*calculus *is obtained by adding to the set of rules $\lambda\omega$ the new following rules:*

$$\begin{array}{ll} \sigma\text{-}/\text{-transition} & a\,[b/]_k[s]_j \longrightarrow a\,[s]_{j+1}[b[s]_{j-k+1}/]_k \quad \text{if } k \leq j \\ /\text{-}\uparrow\text{-transition} & a\,[\uparrow^i]_k[b/]_j \longrightarrow \begin{cases} a[b/]_{j-i}[\uparrow^i]_k & \text{if } k+i \leq j \\ a[\uparrow^{i-1}]_k & \text{if } k \leq j < k+i \end{cases} \\ \uparrow\text{-}\uparrow\text{-transition} & a\,[\uparrow^i]_k[\uparrow^l]_j \longrightarrow \begin{cases} a[\uparrow^l]_{j-i}[\uparrow^i]_k & \text{if } k+i < j \\ a[\uparrow^{i+l}]_k & \text{if } k \leq j \leq k+i \end{cases} \end{array}$$

The set of rules of the ω_e-calculus *is $\lambda\omega_e - \{\sigma - generation\}$.*

Remark 29 *Note that the rule schemes $/\text{-}\uparrow$ and $\uparrow\text{-}\uparrow$ can be merged into the single scheme*

$$a\,[\uparrow^i]_k[s]_j \to a[s]_{j-i}[\uparrow^i]_k \qquad \text{for } k+i < j$$

but they must be kept distinct for the case $k + i = j$ if SN is to be preserved. In fact, the $\uparrow\text{-}\uparrow$-scheme, if admited in the case $k + i = j$, may generate an infinite loop by itself (take for instance $i = k = l = 1$ and $j = 2$).

[10] defined two functions, that are inverse of each other, and establish an isomorphism between λs_e and $\lambda\omega_e$. Furthermore, their restriction to ground terms also establishes an isomorphism between λs and $\lambda\omega$. These isomorphisms translate properties of λs and λs_e to $\lambda\omega$ and $\lambda\omega_e$, respectively.

Definition 30 *The functions $T : \Lambda s_{op} \to \Lambda\omega^t_{op}$ and $S : \Lambda\omega^t_{op} \to \Lambda s_{op}$ are defined inductively by:*

$$\begin{array}{ll} T(X) = X & S(X) = X \\ T(\mathtt{n}) = \mathtt{n} & S(\mathtt{n}) = \mathtt{n} \\ T(a\,b) = T(a)T(b) & S(a\,b) = S(a)S(b) \\ T(\lambda a) = \lambda T(a) & S(\lambda a) = \lambda S(a) \\ T(a\,\sigma^j b) = T(a)[T(b)/]_j & S(a\,[b/]_j) = S(a)\,\sigma^j S(b) \\ T(\varphi^i_k a) = T(a)[\uparrow^{i-1}]_{k+1} & S(a\,[\uparrow^i]_k) = \varphi^{i+1}_{k-1}(S(a)) \end{array}$$

We use the same names T and S for the trivial restrictions of these functions to ground terms. The context will be always clear enough in order to avoid ambiguities.

Theorem 31 (cf. [10]) *The following hold:*

1. *Let $a, b \in \Lambda s$. If $a \to_s b$ then $T(a) \to_\omega T(b)$. If $a \to_{\lambda s} b$ then $T(a) \to_{\lambda\omega} T(b)$.*
2. *Let $a, b \in \Lambda s_{op}$. If $a \to_{s_e} b$ then $T(a) \to_{\omega_e} T(b)$. If $a \to_{\lambda s_e} b$ then $T(a) \to_{\lambda\omega_e} T(b)$.*
3. *Let $a, b \in \Lambda\omega^t$. If $a \to_\omega b$ then $S(a) \to_s S(b)$. If $a \to_{\lambda\omega} b$ then $S(a) \to_{\lambda s} S(b)$.*
4. *Let $a, b \in \Lambda\omega^t_{op}$. If $a \to_{\omega_e} b$ then $S(a) \to_{s_e} S(b)$. If $a \to_{\lambda\omega_e} b$ then $S(a) \to_{\lambda s_e} S(b)$.*
5. *If $a \in \Lambda\omega^t$ or $a \in \Lambda\omega^t_{op}$, then we have $T(S(a)) = a$.*
6. *If $a \in \Lambda s$ or $a \in \Lambda s_{op}$, then we have $S(T(a)) = a$.*

Now that the calculi have been proved isomorphic, all the results mentioned in sections 1.2 and 1.3 concerning λs and λs_e translate into corresponding results for the sort `term` to $\lambda\omega$ and $\lambda\omega_e$.

Theorem 32 *The following hold:*

1. *The ω-calculus is SN and confluent on $\Lambda\omega^t$.*
2. *Let $a, b \in \Lambda$. If $a \twoheadrightarrow_{\lambda\omega} b$ then $a \twoheadrightarrow_\beta b$. If $a \to_\beta b$ then $a \twoheadrightarrow_{\lambda\omega} b$.*
3. *The $\lambda\omega$-calculus is confluent on $\Lambda\omega^t$.*
4. *Pure terms which are SN in the λ-calculus are also SN in the $\lambda\omega$-calculus.*
5. *The ω_e-calculus is weakly normalising and confluent.*
6. *The $\lambda\omega_e$-calculus is confluent on open terms.*
7. *Let $a, b \in \Lambda$. If $a \twoheadrightarrow_{\lambda\omega_e} b$ then $a \twoheadrightarrow_\beta b$. If $a \to_\beta b$ then $a \twoheadrightarrow_{\lambda\omega_e} b$.*