Extending a λ-calculus with Explicit Substitution which Preserves Strong Normalisation into a Confluent Calculus on Open Terms[†]

Fairouz Kamareddine and Alejandro Ríos
Department of Computing Science, 17 Lilybank Gardens, University of Glasgow, Glasgow G12 8QQ, Scotland, fax: +44 41 330 4913,
email: fairouz@dcs.gla.ac.uk and rios@dcs.gla.ac.uk

Abstract

The last fifteen years have seen an explosion in work on explicit substitution, most of which is done in the style of the $\lambda\sigma$-calculus. In (Kamareddine & Ríos, 1995a), we extended the λ-calculus with explicit substitutions by turning de Bruijn's meta-operators into object-operators offering a style of explicit substitution that differs from that of $\lambda\sigma$. The resulting calculus, λs, remains as close as possible to the λ-calculus from an intuitive point of view and, while preserving strong normalisation (Kamareddine & Ríos, 1995a), is extended in this paper to a confluent calculus on open terms: the λs_e-caculus. Since the establishment of these results, another calculus, $\lambda\zeta$, came into being in (Muñoz Hurtado, 1996) which preserves strong normalisation and is itself confluent on open terms. However, we believe that λs_e still deserves attention because, while offering a new style to work with explicit substitutions, it is able to simulate one step of classical β-reduction, whereas $\lambda\zeta$ is not.

To prove confluence we introduce a generalisation of the interpretation method (cf. (Hardin, 1989) and (Curien *et al.*, 1992)) to a technique which uses weak normal forms (instead of strong ones). We consider that this extended method is a useful tool to obtain confluence when strong normalisation of the subcalculus of substitutions is not available.

In our case, strong normalisation of the corresponding subcalculus of substitutions s_e, is still a challenging open problem to the rewrite community but its weak normalisation is established here via an effective strategy.

Introduction

Most literature on the λ-calculus considers substitution as an implicit operation. It means that the computations to perform substitution are usually described with operators which do not belong to the language of the λ-calculus. There has however

† We are grateful to the anonymous referees for their comments and suggestions. We are also grateful to Hans Zantema for his interest in the strong normalisation property of our calculus and for his proof of termination of our σ-σ-transition rule. Kamareddine is grateful to Boston University and in particular to Assaf Kfoury and Joe Wells for their hospitality during work on this article. This work was carried out under EPSRC grant GR/K25014.

been an interest in formalising substitution explicitly in order to provide a theoretical framework for the implementation of functional programming languages. Various calculi including new operators to denote substitution have been proposed. Amongst these calculi we mention $C\lambda\xi\phi$ (cf. (de Bruijn, 1978)); the calculi of categorical combinators (cf. (Curien, 1986)); $\lambda\sigma$, $\lambda\sigma_{\Uparrow}$, $\lambda\sigma_{SP}$ (cf. (Abadi *et al.*, 1991), (Curien *et al.*, 1992), (Ríos, 1993)) referred to as the $\lambda\sigma$-family; $\varphi\sigma BLT$ (cf. (Kamareddine & Nederpelt, 1993)); $\lambda\upsilon$ (cf. (Benaissa *et al.*, 1995)), a descendant of the $\lambda\sigma$-family; λs (cf. (Kamareddine & Ríos, 1995a)); $\lambda\mathtt{exp}$ (cf. (Bloo, 1995)) and $\lambda\zeta$ (cf. (Muñoz Hurtado, 1996)).

These calculi (except $\lambda\mathtt{exp}$) are described in a de Bruijn setting where natural numbers play the role of the classical variables. Classical terms are coded as *closed terms* in these calculi and called *pure terms*. A natural question concerning these calculi is the *preservation of strong normalisation*: are strongly normalising terms in the classical λ-calculus still strongly normalising when considered as pure terms of these new calculi? This question is obviously important. However, various calculi of explicit substitutions do not possess this property.

It is possible to consider, besides the classical variables (now numbers), real variables (which correspond to meta-variables in the classical setting). The terms obtained with this extended syntax are called *open terms* and they can be considered as *contexts*, the new variables corresponding to holes. Hence the interest in studying the calculi on open terms, since they allow contexts as first class citizens.

The main interest in introducing the λs-calculus (cf. (Kamareddine & Ríos, 1995a)) was to provide a calculus of explicit substitutions which would both preserve strong normalisation and have a confluent extension on open terms. There are calculi of explicit substitutions which are confluent on open terms: the $\lambda\sigma_{\Uparrow}$-calculus (cf. (Hardin & Lévy, 1989) and (Curien *et al.*, 1992)), but the non-preservation of strong normalisation for $\lambda\sigma_{\Uparrow}$, as well as for the rest of the $\lambda\sigma$-family and for the categorical combinators, has recently been proved (cf. (Melliès, 1995)). There are also calculi which satisfy the preservation property: the $\lambda\upsilon$-calculus (cf. (Benaissa *et al.*, 1995)), but this calculus is not confluent on open terms and the existence of a confluent extension of $\lambda\upsilon$ is still unknown.

We proved in (Kamareddine & Ríos, 1995a) that λs preserves strong normalisation and proposed the extension λs_e in (Kamareddine & Ríos, 1995b), where we proved its local confluence on open terms and the weak normalisation (every term has at least one normal form) of the corresponding subcalculus of substitutions s_e (the calculus obtained from λs_e by removing the rule that starts β-reduction). Confluence of λs_e and strong normalisation (all derivations terminate) of s_e were left open.

This paper establishes the confluence of λs_e making λs a calculus which preserves strong normalisation and admits a confluent extension on open terms. Preservation of strong normalisation of λs_e and strong normalisation of s_e remain open. As far as we know, at the time of writing this paper, no other calculus which had these two properties existed. Since then, the $\lambda\zeta$-calculus (cf. (Muñoz Hurtado, 1996)) came into being which preserves strong normalisation, is itself confluent on open terms and possesses a strongly normalising subcalculus of substitutions. The $\lambda\zeta$-

calculus is obtained by a clever introduction of two new applications that allows the passage of substitutions within the classical application only if the latter has a head variable. This is done to cut the branch of the critical pair which is responsible of the non-confluence of $\lambda \upsilon$ on open terms. Unfortunately, $\lambda\zeta$ is not able to simulate one step of clasical β-reduction as shown in (Muñoz Hurtado, 1996), it simulates only a "big step" beta reduction. Furthermore, this lack of the simulation property is an uncommon feature among calculi of explicit substitutions.

As the strong normalisation of s_e remains open, the *interpretation method* (cf. (Hardin, 1989), (Curien *et al.*, 1992)), which is usually used to prove the confluence of a λ-calculus with explicit substitutions is not applicable to λs_e. In Section 1 we propose a generalisation of the interpretation method which enables us to prove the confluence of λs_e with just weak normal forms. The method is general enough to be applied to any reduction systems satisfying the hypotheses (not necessarily a calculus of explicit substitutions) and therefore we consider it a new tool to prove confluence.

Section 2 is devoted to the syntax and rules of the calculi we are going to deal with: the λ-calculus à la de Bruijn, the λs-calculus and its extension the λs_e-calculus together with a summary of the results obtained so far (cf. (Kamareddine & Ríos, 1995a) and (Kamareddine & Ríos, 1995b)) for these calculi. At the end of the section we provide motivation for the new rules of λs_e and finally we compare λs_e with $\lambda\sigma$, $\lambda\upsilon$, $\lambda\sigma_{\Uparrow}$ and $\lambda\zeta$.

In Section 3 we recall the description of the s_e-normal forms, define a strategy for computing them and establish the weak normalisation of s_e. We also prove that s_e-normal forms are preserved by s_e-reductions and that the s_e-calculus is confluent on open terms.

In Section 4 we introduce the calculus of the interpretation, whose only rule we call β', and prove that the *σ-generation* rule (the rule that starts β-reduction) can be simulated on the corresponding weak normal forms by β'.

In Section 5 we prove the confluence of β' à la Tait-Martin-Löf in order to apply the generalised interpretation method to show the confluence of the λs_e-calculus. We also show that the λs_e-calculus is correct/sound with respect to the λ-calculus in that, all λs_e-derivations beginning and ending with pure terms can also be obtained in the λ-calculus.

We conclude by stating the problems which remain still open and we include a result by Hans Zantema showing the termination of the rule of λs_e which enables the transition of a substitution operator over another one.

This article is an abridged version of (Kamareddine & Ríos, 1996) where the proofs are presented in more detail.

1 The Generalised Interpretation Method

We begin by introducing the notation we use and some essential definitions and properties.

Definition 1

Let A be a set and R a binary relation on A. We denote the fact $(a,b) \in R$ by $a \to_R b$ or $a \to b$ when the context is clear enough. We call *reduction* this relation and *reduction system*, the pair (A, R). R^* or $\twoheadrightarrow_R$ or just $\twoheadrightarrow$ denote the reflexive and transitive closure of R. R^+ or $\twoheadrightarrow_R^+$ or just $\twoheadrightarrow^+$ denote the transitive closure of R. When $a \twoheadrightarrow b$ we say there exists a *derivation* from a to b.

Definition 2
Let R be a reduction on A. For R, we define local confluence (WCR), confluence (CR) and strong confluence (SCR) respectively as follows:

1. WCR: $\forall a,b,c \in A\ \exists d \in A\ : (a \to b \wedge a \to c) \Rightarrow (b \twoheadrightarrow d \wedge c \twoheadrightarrow d)$.
2. CR: $\forall a,b,c \in A\ \exists d \in A\ : (a \twoheadrightarrow b \wedge a \twoheadrightarrow c) \Rightarrow (b \twoheadrightarrow d \wedge c \twoheadrightarrow d)$.
3. SCR: $\forall a,b,c \in A\ \exists d \in A\ : (a \to b \wedge a \to c) \Rightarrow (b \to d \wedge c \to d)$.

Definition 3
Let R be a reduction on A. We say that $a \in A$ is an R-*normal form* (R-nf for short) if there exists no $b \in A$ such that $a \to b$ and we say that b *has a normal form* if there exists a normal form a such that $b \twoheadrightarrow a$. R is *weakly normalising* (WN) if every $a \in A$ has an R-normal form. R is *strongly normalising* (SN) if there is no infinite sequence $(a_i)_{i \geq 0}$ in A such that $a_i \to a_{i+1}$ for all $i \geq 0$.

Note that confluence of R guarantees unicity of R-normal forms. In that case, the R-normal form of a, if it exists, is denoted by $R(a)$. Strong normalisation implies weak normalisation and therefore the existence of normal forms.

At some point we shall need the following lemmas (cf. (Barendregt, 1984)).

Lemma 1
Let R be a reduction, if R is SCR then R^* is also SCR.

Lemma 2 (Newman)
Every strongly normalising, locally confluent reduction is confluent.

We state now the interpretation method we wish to generalise. This method was first identified in (Hardin, 1989), where it was used for the categorical combinators. In (Curien *et al.*, 1992), it is used to prove the confluence of the weak $\lambda\sigma$-calculus, of the $\lambda\sigma$-calculus on closed terms and the non-confluence of the $\lambda\sigma_{SP}$-calculus on open terms. In (Ríos, 1993), it was used to prove the confluence of the $\lambda\sigma_{SP}$-calculus on semi-closed terms. Finally, in (Benaissa *et al.*, 1995) and (Kamareddine & Ríos, 1995a), it was used to prove the confluence of $\lambda\upsilon$ and λs respectively.

Lemma 3 (Interpretation method)
Let $R = R_1 \cup R_2$ where R_1 is a confluent and SN reduction on A and R_2 an arbitrary reduction. If there exists a reduction R' on the set of R_1-normal forms satisfying $R' \subseteq R^*$ and $(a \to_{R_2} b \ \Rightarrow\ R_1(a) \twoheadrightarrow_{R'} R_1(b))$, then R' is confluent iff R is confluent.

Lemma 4 (Generalised interpretation method (GIM))
Let $B \subseteq A$, R and R' reduction relations on A and B respectively and $f : A \to B$ such that:

$$\begin{tikzcd}
 & b \arrow[rr, two heads, "R"] & & f(b) \arrow[dr, dotted, two heads, "R'"] & \\
a \arrow[ur, two heads, "R"] \arrow[rr, two heads, "R"] \arrow[dr, two heads, "R"'] & & f(a) \arrow[ur, two heads, "R'"] \arrow[dr, two heads, "R'"'] & & d \\
 & c \arrow[rr, two heads, "R"] & & f(c) \arrow[ur, dotted, two heads, "R'"'] &
\end{tikzcd}$$

Fig. 1. Generalised interpretation method

1. $R' \subseteq R^*$
2. $\forall a \in A : a \twoheadrightarrow_R f(a)$
3. $\forall a, b \in A : a \rightarrow_R b \;\Rightarrow\; f(a) \twoheadrightarrow_{R'} f(b)$

then, if R' is confluent, R is also confluent. Moreover, if we also have:
4. $\forall b \in B : b \twoheadrightarrow_{R'} f(b)$
then R' is confluent iff R is confluent.

Proof

($\Rightarrow$) Assume $a \twoheadrightarrow_R b$ and $a \twoheadrightarrow_R c$. By 2, $x \twoheadrightarrow_R f(x)$ for $x \in \{a, b, c\}$. By 3, $f(a) \twoheadrightarrow_{R'} f(b)$ and $f(a) \twoheadrightarrow_{R'} f(c)$. Now, confluence of R' gives d such that $f(b) \twoheadrightarrow_{R'} d$ and $f(c) \twoheadrightarrow_{R'} d$. 1 implies $f(b) \twoheadrightarrow_R d$ and $f(c) \twoheadrightarrow_R d$. Therefore, $b \twoheadrightarrow_R d$ and $c \twoheadrightarrow_R d$ (see Figure 1).

($\Leftarrow$) Let a, b, $c \in B$, $a \twoheadrightarrow_{R'} b$ and $a \twoheadrightarrow_{R'} c$. By 1, $a \twoheadrightarrow_R b$ and $a \twoheadrightarrow_R c$. By confluence of R, there exists d such that $b \twoheadrightarrow_R d$ and $b \twoheadrightarrow_R d$. By 4, $x \twoheadrightarrow_{R'} f(x)$ for $x \in \{b, c, d\}$. By 3, $f(b) \twoheadrightarrow_{R'} f(d)$ and $f(c) \twoheadrightarrow_{R'} f(d)$ and we are done.

$\square$

A particular case of the GIM that is useful for calculi of explicit substitutions is the following:

Corollary 1 (GIM for explicit substitutions (GIMES))
Let $R = R_1 \cup R_2$ where R_1 and R_2 are arbitrary reductions on A. Let B be the set of R_1-normal forms and let $f : A \rightarrow B$ be a function (strategy) such that $f(a)$ is an R_1-normal form of a. If there exists a reduction R' on the set of R_1-normal forms satisfying

1. $R' \subseteq R^*$
2. $\forall a, b : a \rightarrow_{R_1} b \;\Rightarrow\; f(a) \twoheadrightarrow_{R'} f(b)$
3. $\forall a, b : a \rightarrow_{R_2} b \;\Rightarrow\; f(a) \twoheadrightarrow_{R'} f(b)$

then R' is confluent iff R is confluent.

Proof

Note that 1,2,3 of GIMES imply 1,2,3,4 of GIM. In particular, 2,3 of GIMES imply 3 of GIM, 2 of GIM holds as $f(a)$ is an R_1-nf of a and hence $a \twoheadrightarrow_R f(a)$. Moreover, 4 of GIM holds because for $b \in B$, b is in R_1-nf and hence $b = f(b)$. $\square$

In the context of the GIM lemma and its corollary GIMES, the function f is called the *interpretation function*; B, the *set of the interpretation* and $(B, \rightarrow_{R'})$, the *calculus of the interpretation.*

We use a particular case of the GIMES where 2 is strengthened to: $a \rightarrow_{R_1} b \;\Rightarrow\; f(a) = f(b)$ (syntactic identity). Having established that GIM $\Longrightarrow$ GIMES, we comment now that even the GIMES generalises the interpretation method. In fact, when R_1 is confluent and SN, R_1-normal forms exist and are unique. Hence there is only one f $(f(a) = R_1(a))$ such that $f(a)$ is a normal form of a. Remark that in this case hypothesis 2. of the GIMES is obviously satisfied.

2 The calculi

2.1 The classical λ-calculus in de Bruijn notation

We assume the reader familiar with de Bruijn notation (cf. (de Bruijn, 1972), (Kamareddine & Ríos, 1996)). We define Λ, the *set of terms with de Bruijn indices*, as follows:

$$\Lambda ::= \mathbb{N} \mid (\Lambda\Lambda) \mid (\lambda\Lambda)$$

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. We say that a reduction $\rightarrow$ is *compatible on* Λ when for all a, b, $c \in \Lambda$, we have $a \rightarrow b$ implies $a\,c \rightarrow b\,c$, $c\,a \rightarrow c\,b$ and $\lambda a \rightarrow \lambda b$.

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 4
The *updating functions* $U^i_k : \Lambda \rightarrow \Lambda$ for $k \geq 0$ and $i \geq 1$ are defined inductively:

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

$$U^i_k(\lambda a) = \lambda(U^i_{k+1}(a))$$

Now we define the family of meta-substitution functions:

Definition 5
The *meta-substitution 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:

$$(a_1 a_2)\{\!\{\mathtt{j} \leftarrow b\}\!\} = (a_1\{\!\{\mathtt{j} \leftarrow b\}\!\})\,(a_2\{\!\{\mathtt{j} \leftarrow b\}\!\}) \qquad \mathtt{n}\{\!\{\mathtt{j} \leftarrow b\}\!\} = \begin{cases} \mathtt{n-1} & \text{if } n > j \\ U^j_0(b) & \text{if } n = j \\ \mathtt{n} & \text{if } n < j\,. \end{cases}$$

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

The following lemmas establish the properties of meta-substitution and updating (cf. (Kamareddine & Ríos, 1996)).

Lemma 5
For $k < n < k+i$ we have: $U_k^{i-1}(a) = U_k^i(a)\{\!\{\mathtt{n} \leftarrow b\}\!\}$.

Lemma 6
For $l \leq k < l+j$ we have: $U_k^i(U_l^j(a)) = U_l^{j+i-1}(a)$.

Lemma 7
For $k+i \leq n$ we have: $U_k^i(a)\{\!\{\mathtt{n} \leftarrow b\}\!\} = U_k^i(a\{\!\{\mathtt{n}-\mathtt{i}+\mathtt{1} \leftarrow b\}\!\})$.

Lemma 8 (Meta-substitution Lemma)
For $i \leq n$ we have:

$$a\{\!\{\mathtt{i} \leftarrow b\}\!\}\{\!\{\mathtt{n} \leftarrow c\}\!\} = a\{\!\{\mathtt{n}+\mathtt{1} \leftarrow c\}\!\}\{\!\{\mathtt{i} \leftarrow b\{\!\{\mathtt{n}-\mathtt{i}+\mathtt{1} \leftarrow c\}\!\}\}\!\}$$

Lemma 9
For $l+j \leq k+1$ we have: $U_k^i(U_l^j(a)) = U_l^j(U_{k+1-j}^i(a))$.

Lemma 10 (Distribution Lemma)
For $n \leq k+1$ we have:

$$U_k^i(a\{\!\{\mathtt{n} \leftarrow b\}\!\}) = U_{k+1}^i(a)\{\!\{\mathtt{n} \leftarrow U_{k-n+1}^i(b)\}\!\} .$$

Definition 6
β-reduction is the least compatible reduction on Λ generated by:

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

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

Theorem 1
The λ-calculus à la de Bruijn is confluent.

2.2 The λs-calculus

The subjacent idea in the mechanism of λs is the explicit handling of the meta-operators given in Definitions 4 and 5. 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 7
The *set of terms*, noted Λs , *of the λs-calculus* is given as follows:

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

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	$\mathtt{n}\, \sigma^j b$	$\longrightarrow$	$\begin{cases} \mathtt{n-1} & \text{if } n > j \\ \varphi^j_0\, b & \text{if } n = j \\ \mathtt{n} & \text{if } n < j \end{cases}$
φ-λ-transition	$\varphi^i_k(\lambda a)$	$\longrightarrow$	$\lambda(\varphi^i_{k+1}\, a)$
φ-app-transition	$\varphi^i_k(a_1\, a_2)$	$\longrightarrow$	$(\varphi^i_k\, a_1)\, (\varphi^i_k\, a_2)$
φ-destruction	$\varphi^i_k\, \mathtt{n}$	$\longrightarrow$	$\begin{cases} \mathtt{n+i-1} & \text{if } n > k \\ \mathtt{n} & \text{if } n \le k \end{cases}$

Fig. 2. The λs-calculus

We take a, b, c to range over Λs. A term containing neither σ's nor φ's is called a *pure term*. Λ denotes the set of pure terms.

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

We include, besides the rule mimicking the β-rule (*σ-generation*), a set of rules which are the equations in Definitions 4 and 5 orientated from left to right.

Definition 8
The *λs-calculus* is the reduction system $(\Lambda s, \to_{\lambda s})$, where $\to_{\lambda s}$ is the least compatible reduction on Λs generated by the rules given in Figure 2. We use λs to denote this set of rules. The *subcalculus of substitutions associated with the λs-calculus* is the reduction system generated by the set of rules $s = \lambda s - \{\sigma\text{-}generation\}$ and we call it the *s-calculus*.

The *σ-generation* rule starts β-reduction by generating a substitution operator at the first level (σ^1). The *σ-app* and *σ-λ* rules allow this operator to travel throughout the term until its arrival to the variables. If a variable should be affected by the substitution, the *σ-destruction* rules (case $j = n$) carry out the substitution of the variable by the updated term, thus introducing the updating operators. Finally, the other rules compute the updating.

We state now the main properties of the λs-calculus (cf. (Kamareddine & Ríos, 1995a) and (Kamareddine & Ríos, 1995b)).

Theorem 2 (SN and confluence of s)
The s-calculus is strongly normalising and confluent on Λs. Hence, every term a has a unique s-normal form denoted $s(a)$.

Lemma 11

The set of s-normal forms is exactly Λ.

Lemma 12
For all $a,\, b \in \Lambda s$ we have:

$$s(a\,b) = s(a)s(b)\ , \quad s(\lambda a) = \lambda(s(a))\ , \quad s(\varphi^i_k a) = U^i_k(s(a))\ ,$$
$$s(a\,\sigma^j b) = s(a)\{\!\{\mathtt{j} \leftarrow s(b)\}\!\}\ .$$

Lemma 13
Let $a,\, b \in \Lambda s$, if $a \to_{\sigma-gen} b$ then $s(a) \twoheadrightarrow_\beta s(b)$.

Corollary 2
Let $a,\, b \in \Lambda s$, if $a \twoheadrightarrow_{\lambda s} b$ then $s(a) \twoheadrightarrow_\beta s(b)$.

Corollary 3 (Soundness)
Let $a,\, b \in \Lambda$, if $a \twoheadrightarrow_{\lambda s} b$ then $a \twoheadrightarrow_\beta b$.

Lemma 14 (Simulation of β-reduction)
Let $a,\, b \in \Lambda$, if $a \to_\beta b$ then $a \twoheadrightarrow_{\lambda s} b$.

Theorem 3 (Confluence of λs)
The λs-calculus is confluent on Λs.

Theorem 4 (Preservation of SN)
Pure terms which are strongly normalising in the λ-calculus are also strongly normalising in the λs-calculus.

Theorem 5 (SN of typed terms)
Every well typed term is strongly normalising in the simply typed λs-calculus.

2.3 The λs_e-calculus

We introduce the open terms and the rules added to λs to obtain the λs_e-calculus.

Definition 9
The set of *open terms*, noted Λs_{op} is given as follows:

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

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, *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. But as $((\lambda X)\sigma^1\mathtt{1})(Y\sigma^1\mathtt{1}) \to$ $\to (X\sigma^2\mathtt{1})\sigma^1(Y\sigma^1\mathtt{1})$, the solution to the problem seems at hand if one has in mind the properties of meta-substitutions and updating functions of the λ-calculus in the Bruijn notation (cf. Lemmas 5 - 10). These properties are equalities which

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	$i \le j$
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	$k+i \le j$
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	$j \le k+1$
φ-φ-transition 1	$\varphi^i_k\,(\varphi^j_l\,a)$	$\longrightarrow$	$\varphi^j_l\,(\varphi^i_{k+1-j}\,a)$	if	$l+j \le k$
φ-φ-transition 2	$\varphi^i_k\,(\varphi^j_l\,a)$	$\longrightarrow$	$\varphi^{j+i-1}_l\,a$	if	$l \le k < l+j$

Fig. 3. The new rules of the λs_e-calculus

can be given a suitable orientation and the new rules, thus obtained, added to λs give origin to a rewriting system which happens to be locally confluent (cf. (Kamareddine & Ríos, 1995b)). For instance, the rule corresponding to the Meta-substitution Lemma (Lemma 8) is the *σ-σ-transition* rule given in Figure 3. The addition of this rule solves the critical pair in our counterexample, since now we have $(X\sigma^1 Y)\sigma^1 \mathtt{1} \to (X\sigma^2 \mathtt{1})\sigma^1(Y\sigma^1 \mathtt{1})$.

Definition 10
The set of rules λs_e is obtained by adding the rules in Figure 3 to the rules of the λs-calculus (Figure 2). 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 *subcalculus 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.*

We call the rules whose name start with σ, σ-rules. We define similarly the φ-rules.

Remark that when transcribing Lemmas 5 - 10 as rewriting rules, instead of keeping the condition $l + j \le k + 1$ for rule *φ-φ-transition 1*, we restricted it to $l + j \le k$. The reason for this is that for the extreme case $i = 1$, $j = 1$ and $l + j = k + 1$ we would have:

$$\varphi^i_k(\varphi^j_l(a)) \to \varphi^j_l(\varphi^i_{k+1-j}(a)) \to \varphi^i_{k+1-j}(\varphi^j_{l+1-i}(a) = \varphi^i_k(\varphi^j_l(a))\,,$$

giving an infinite loop which would destroy strong normalisation. Furthermore, for $l + j = k + 1$ we have *φ-φ-transition 2* that allows us to reduce $\varphi^i_k(\varphi^j_l(a))$. Note also that for $a, b \in \Lambda s_{op}$:

1. $(\varphi^i_k a)\,\sigma^j b$ has a redex at the root iff $j > k$.
2. $\varphi^i_k(\varphi^j_l a)$ has a redex at the root iff $k \ge l$.

Finally, local confluence for λs_e is obtained by analysis of critical pairs (cf. (Kamareddine & Ríos, 1995b)):

Theorem 6 (Local confluence)

The s_e- and λs_e-calculi are locally confluent on Λs_{op}.

We give now further motivation for the rules of λs_e. Motivation behind the rules of Figure 2 was given in (Kamareddine & Ríos, 1995a) and motivation for explicit substitution rules that belong to the same family can be found in (Kamareddine & Nederpelt, 1993). Hence, we concentrate on the rules of Figure 3.

We gave already some motivation for the *σ-σ-transition* rule where we said that such a rule helps to re-establish confluence. The other rules were also introduced as a necessity to close critical pairs. Remark now the following symetries: there are

- two "simplification" rules: *σ-φ-tr.1* and *φ-φ-tr.2*;
- two "distribution" rules: *σ-σ-tr.* and *φ-σ-tr.*;
- two "commutation" rules: *σ-φ-tr.2* and *φ-φ-tr.1*.

The intuitive interpretation of φ^i_k, as for U^i_k, is *the updating of the free variables greater than k with an increment of $i-1$*. In this informal context one must be careful: if a de Bruijn number corresponds to a free variable, the "real" number of such a variable may not be its value. For instance, in $\mathtt{1}\,\lambda\mathtt{2}$, the index $\mathtt{2}$ corresponds to the "real" free variable $\mathtt{1}$. One may check this fact by translating $\mathtt{1}\,\lambda\mathtt{2}$ to classical notation: the result is $x\,\lambda y.x$ where x is the first variable in the free variable list. Remark that $\varphi^i_1(\mathtt{1}\,\lambda\mathtt{2}) \twoheadrightarrow_s \mathtt{1}\,\lambda\mathtt{2}$ whereas $\varphi^4_0(\mathtt{1}\,\lambda\mathtt{2}) \twoheadrightarrow_s \mathtt{4}\,\lambda\mathtt{5}$.

The intuitive interpretation of $a\,\sigma^j b$, like $a\{\!\{\mathtt{j} \leftarrow b\}\!\}$, is the *substitution of the free variables (whose "real" number is j) by the updating (φ^j_0) of b in a*. In the same way that the occurences of the "real" variable $\mathtt{j}$ in λa are the occurrences of the "real" variable $\mathtt{j+1}$ in a, it is easy to check (for the meta-substitutions) that the occurrences of the "real" variable $\mathtt{j}$ in $a\,\sigma^i b$ ($i \leq j$ and $\mathtt{i}$ free in a) are the occurrences of $\mathtt{j+1}$ in a and the occurrences of $\mathtt{j-i+1}$ in b.

Now we explain each type of rule:

- The intuitive interpretations given above of φ^i_k and $a\,\sigma^j b$ explain the distribution rules: the σ^j operator in the LHS of *σ-σ-tr.* must become, on the RHS, σ^{j+1} when acting on a and σ^{j-i+1} when acting on b. In the same way, the transition of φ^i_k into φ^i_{k+1} and φ^i_{k+1-j} is explained for the rule *φ-σ-tr.*.
- The simplification rules are also easy to grasp:
  To understand the rule *φ-φ-transition 2*, let us consider $n > k$. Since $n > l$ and $l + j > k$ implies $n + j - 1 > k$, we get $\varphi^i_k(\varphi^j_l\mathtt{n}) \rightarrow_s \varphi^i_k(\mathtt{n}+\mathtt{j}-\mathtt{1}) \rightarrow_s \mathtt{n}+\mathtt{j}+\mathtt{i}-\mathtt{2}$. Now this double process of updating can be achieved by a single updating: $\varphi^{i+j-1}_k\mathtt{n} \rightarrow_s \mathtt{n}+\mathtt{j}+\mathtt{i}-\mathtt{2}$, hence our *$\varphi$-$\varphi$-transition 2* rule.
  The rule *σ-φ-tr.1* may be explained as a void substitution (the variable to be replaced does not occur free). In fact, it is also easy to check (for the meta-updatings) that the occurrences of the "real" variable $\mathtt{j}$ in $\varphi^i_k a$ are the occurrences of $\mathtt{j-i+1}$ in a when $j - i + 1 > k$. Hence, if $j < k + i$, the variable $\mathtt{j}$ cannot occur free in $\varphi^i_k a$ and therefore the substitution in the LHS of the rule is void. Furthermore the dissapearance of the σ^j operator is the reason why the upper index of the φ operator is decreased by 1.
- Finally, both commutation rules postpone an updating: *σ-φ-tr.2* postpones the updating φ^i_k, whereas *φ-φ-tr.1* postpones the updating φ^j_l. The transition

of σ^j into σ^{j-i+1} can be explained by the fact that the occurrences of $\mathtt{j}$ in $\varphi^i_k a$ are the occurrences of $\mathtt{j\text{-}i\text{+}1}$ in a. Analogously, the transition of φ^i_k into φ^i_{k+1-j} can be understood.

We believe that further intuition, from the point of view of normalisation, can be gained in the next section where we describe the s_e-normal forms. We define there the *skeletons* as certain structures of φ and σ operators. The rules can be viewed as acting on skeletons to "order" them (what we call *normal skeletons* should be seen as completely "ordered" structures). This point of view helps to understand the interaction between the indices of the σ operators and the lower indices of the φ operators.

¿From a computational point of view these new rules offer the possibility of interaction between σ- and φ-operators, whereas in λs the interaction of these operators was restricted to de Bruijn numbers, applications and abstractions. This restriction is also present in $\lambda\upsilon$ and enables the preservation of strong normalisation, whereas this property does not hold in $\lambda\sigma$, where interaction of substitutions is available through the composition operator. We believe that the interaction we propose in λs_e is more controlled than the interaction allowed in $\lambda\sigma$, because of the restriction on indices and therefore this stratified interaction would not be harmful from the point of view of preservation. However, the preservation of strong normalisation of λs_e is still an open problem.

We remark that Lemmas 5 - 10 were all the knowledge required about meta-substitutions and meta-updatings to prove confluence of λs (cf. (Kamareddine & Ríos, 1995a)). This knowledge must become available within the calculus if we expect to obtain nice confluence properties. Therefore the new rules about σ- and φ-operators internalise the knowledge in the meta-level about the meta-operators they represent.

We end this section by comparing λs and λs_e with $\lambda\sigma$, $\lambda\sigma_{\Uparrow}$, $\lambda\upsilon$ and $\lambda\zeta$. Since the interpretations[†] T and S of λs into $\lambda\sigma$ and $\lambda\upsilon$, respectively, were already presented in (Kamareddine & Ríos, 1995a), we highlight here the translation into $\lambda\sigma_{\Uparrow}$.

Definition 11
The translation R of λs-terms into $\lambda\sigma_{\Uparrow}$-terms is defined inductively by:

$$\begin{array}{lll} R(\mathtt{n}) = \mathtt{n} & R(ab) = R(a)R(b) & R(a\,\sigma^{i+1}b) = R(a)[\Uparrow^i(R(b)\cdot id)] \\ & R(\lambda a) = \lambda R(a) & R(\varphi^i_k a) = R(a)[\Uparrow^k(\uparrow^{i-1})] \end{array}$$

where $\uparrow^0 = id$, $\uparrow^{n+1} = \uparrow \circ \uparrow^n$ and $\Uparrow^0(s) = s$, $\Uparrow^{n+1}(s) = \Uparrow(\Uparrow^n(s))$.

The following theorem summarizes the properties of these translations:

Theorem 7

[†] T and S are defined on numbers, abstractions and applications like R in Definition 11. We just recall here the translations of substitutions and updatings:

$$\begin{array}{ll} T(a\,\sigma^{i+1}b) = T(a)[\mathtt{1}\cdot\mathtt{2}\cdot\ldots\cdot\mathtt{i}\cdot T(b)[\uparrow^i]\cdot\uparrow^i] & S(a\,\sigma^{i+1}b) = S(a)[\Uparrow^i(S(b)/)] \\ T(\varphi^i_k a) = T(a)[\mathtt{1}\cdot\mathtt{2}\cdot\ldots\cdot\mathtt{k}\cdot\uparrow^{k+i-1}] & S(\varphi^i_k a) = S(a)[\Uparrow^k(\uparrow)]^{i-1} \end{array}$$

where $\uparrow^n$ and $\Uparrow^n$ are as in Definition 11 and $a[s]^0 = a$, $a[s]^{n+1} = (a[s])[s]^n$.

For $a, b \in \Lambda s$ we have:

1. If $a \to_s b$ then $T(a) \twoheadrightarrow_{\sigma}^{+} T(b)$
2. If $a \to_{\lambda s} b$ then $S(a) \twoheadrightarrow_{\lambda\upsilon}^{+} S(b)$ and $R(a) \twoheadrightarrow_{\lambda\sigma_{\Uparrow}}^{+} R(b)$.
3. If $a \to_{\lambda s_e} b$ then $T(a) =_{\lambda\sigma} T(b)$, $S(a) =_{\lambda\upsilon} S(b)$ and $R(a) =_{\lambda\sigma_{\Uparrow}} R(b)$.

Proof
By induction on a, using the classical equalities of $\lambda\sigma$, $\lambda\upsilon$ and $\lambda\sigma_{\Uparrow}$. $\square$

Remark that since $\lambda\zeta$ only differs from $\lambda\upsilon$ in the treatment of applications, the "natural" translation of λs_e into $\lambda\zeta$ is also S. But, as expected, $a \to_{\lambda s_e} b$ does not imply $S(a) =_{\lambda\zeta} S(b)$. The reason for this is that $\lambda\zeta$ is unable to prove $(a\,b)[s] = a[s]b[s]$, in fact $(\lambda.\mathtt{11})(\lambda.\mathtt{11})[s] \neq_{\lambda\zeta} (\lambda.\mathtt{11})[s](\lambda.\mathtt{11})[s]$ because substitutions may be introduced into applications only if the application has a head variable. Therefore, no translation of λs_e into $\lambda\zeta$ preserving equalities seems possible.

Finally, we compare the amount of reduction steps needed to perform some β-reductions of pure terms in the different calculi. We just give two examples to show that for certain terms $\lambda\sigma$ and $\lambda\upsilon$ are more efficient than λs whereas there are terms for which λs is the most efficient. For instance, the term $(\lambda.\mathtt{1})a$ reduces in two steps to a in $\lambda\sigma$ and $\lambda\upsilon$ but $2+n$ steps are needed in λs, where n is the length of $\varphi_0^1 a \twoheadrightarrow a$. On the other hand, terms of the form $(\lambda \cdots \lambda.\mathtt{n})a$, with m λ's and $n > m > 1$, can be reduced more efficiently in λs beacuse the single step $\mathtt{n}\sigma^m a \to_s \mathtt{n-1}$ requires $2m-1$ steps in $\lambda\upsilon$ and much more in $\lambda\sigma$. Remark that $\lambda\zeta$ is less efficient than $\lambda\upsilon$ every time the new mechanism of application is started.

3 The weak normal forms

The following theorem classifies s_e-normal forms (cf. (Kamareddine & Ríos, 1995b)).

Theorem 8
A term $a \in \Lambda s_{op}$ is an s_e-normal form iff one of the following holds:

- $a \in \mathbf{V} \cup \mathbb{N}$, i.e. a is a variable or a de Bruijn number.
- $a = b\,c$, where b and c are s_e-normal forms.
- $a = \lambda b$, where b is an s_e-normal form.
- $a = b\,\sigma^j c$, where c is an s_e-nf and b is an s_e-nf of the form X, or $d\,\sigma^i e$ with $j < i$, or $\varphi_k^i d$ with $j \le k$.
- $a = \varphi_k^i b$, where b is an s_e-nf of the form X, or $c\,\sigma^j d$ with $j > k+1$, or $\varphi_l^j c$ with $k < l$.

There is a simple way to describe the s_e-nf's using *item notation* (Kamareddine & Nederpelt, 1995). In this notation one writes $a\,b = (b\,\delta)a$, $\lambda a = (\lambda)a$, $a\,\sigma^i b = (b\,\sigma^i)a$ and $\varphi_k^i a = (\varphi_k^i)a$. $(b\,\delta)$, (λ), $(c\,\sigma^i)$, (φ_k^i) are called *items* (δ-, λ-, σ- and φ-items, respectively) and b and c the *bodies* of the respective items. A sequence of items is called a *segment*. Note that every term in Λs_{op} can be written as $\overline{s}\mathtt{n}$ or $\overline{s}X$ for some segment $\overline{s}$.

A *normal $\sigma\varphi$-segment* $\overline{s}$ is a sequence of σ- and φ-items such that every pair of adjacent items in $\overline{s}$ has one of the following forms:

$$(\varphi^i_k)(\varphi^j_l) \text{ and } k<l \quad (\varphi^i_k)(b\,\sigma^j) \text{ and } k<j-1 \quad (b\,\sigma^i)(c\,\sigma^j) \text{ and } i<j$$
$$(b\,\sigma^j)(\varphi^i_k) \text{ and } j\le k.$$

Two examples of normal $\sigma\varphi$-segments are: $(\varphi^2_3)(\varphi^1_4)(\varphi^6_7)(b\sigma^9)(c\sigma^{11})(\varphi^2_{11})(\varphi^5_{16})$ and $(b\sigma^1)(c\sigma^3)(d\sigma^4)(\varphi^2_5)(\varphi^1_6)(\varphi^4_7)(a\sigma^{10})$.

Finally, in order to make the dependence of a normal $\sigma\varphi$-segment on the bodies of the σ-items explicit, we define the *skeleton* of a $\sigma\varphi$-segment as the pseudo-segment obtained by removing the bodies of the σ-items. We call it pseudo-segment because it is not a segment as defined above. We write $\overline{\sigma\varphi}(a_1,\ldots,a_n)$ to mean the normal $\sigma\varphi$-segment $\overline{s}$ (whose skeleton is $\overline{\sigma\varphi}$) which has n σ-items such that the body of the i-th (begining from the left) of them is a_i. We call such a skeleton a *normal skeleton of arity n*. For example, the following segments:

$$\overline{s'} = (\varphi^2_3)(\varphi^1_4)(\varphi^6_7)(b\sigma^9)(c\sigma^{11})(\varphi^2_{11})(b\sigma^{14})(\varphi^5_{16})$$

$$\overline{s''} = (b\sigma^1)(c\sigma^3)(d\sigma^4)(\varphi^2_5)(\varphi^1_6)(\varphi^4_7)(a\sigma^{10})$$

have the respective skeletons

$$\overline{\sigma\varphi'} = (\varphi^2_3)(\varphi^1_4)(\varphi^6_7)(\sigma^9)(\sigma^{11})(\varphi^2_{11})(\sigma^{14})(\varphi^5_{16})$$

$$\overline{\sigma\varphi''} = (\sigma^1)(\sigma^3)(\sigma^4)(\varphi^2_5)(\varphi^1_6)(\varphi^4_7)(\sigma^{10})\,,$$

and are written: $\overline{s'} = \overline{\sigma\varphi'}(b,c,b)$ and $\overline{s''} = \overline{\sigma\varphi''}(b,c,d,a)$.

We can now give another description of the s_e-nf's, as presented in (Kamareddine & Ríos, 1995b). This different point of view of the structure of the s_e-normal forms will be exploited later.

Theorem 9

The s_e-normal forms can be described by the following syntax:

$$NF ::= \mathbf{V} \mid \mathbb{N} \mid (NF\,\delta)NF \mid (\lambda)NF \mid \overline{\sigma\varphi}(NF,\ldots,NF)\,\mathbf{V}$$

where $\overline{\sigma\varphi}$ are normal skeletons. Terms of the form $\overline{\sigma\varphi}(a_1,\ldots,a_n)X$ are called *$\sigma\varphi$-normal forms* (even if they are not written in item notation).

Now, we define an *innermost* strategy (before reducing a redex all its subterms must have been already normalised) to calculate normal forms. We do it in three steps:

1. We define a function s'_e to evaluate a normal form of $\varphi^i_k d$ for $d \in NF$.
2. We use s'_e to define a function s''_e to evaluate a normal form of $d\,\sigma^j e$ for $d,\, e \in NF$.
3. We use s'_e and s''_e to define a function s^*_e to evaluate an s_e-normal form for $a \in \Lambda s_{op}$.

Definition 12

Let $d \in NF$, we define $s'_e(\varphi^i_k d)$ by induction on d as follows:

$$
\begin{array}{lcl}
s'_e(\varphi^i_k X) & = & \varphi^i_k X \\
s'_e(\varphi^i_k \mathtt{n}) & = & \begin{cases} \mathtt{n+i-1} & \text{if } n > k \\ \mathtt{n} & \text{if } n \leq k \end{cases} \\
s'_e(\varphi^i_k(a\,b)) & = & s'_e(\varphi^i_k a)\, s'_e(\varphi^i_k b) \\
s'_e(\varphi^i_k(\lambda a)) & = & \lambda s'_e(\varphi^i_{k+1} a) \\
s'_e(\varphi^i_k(\varphi^j_l a)) & = & \begin{cases} \varphi^i_k(\varphi^j_l a) & \text{if } k < l \\ \varphi^{j+i-1}_l a & \text{if } l \leq k < l+j \\ \varphi^j_l(s'_e(\varphi^i_{k+1-j} a)) & \text{if } l+j \leq k \end{cases} \\
s'_e(\varphi^i_k(a\,\sigma^j b)) & = & \begin{cases} \varphi^i_k(a\,\sigma^j b) & \text{if } j > k+1 \\ s'_e(\varphi^i_{k+1} a)\,\sigma^j\, s'_e(\varphi^i_{k+1-j} b) & \text{if } j \leq k+1 \end{cases}
\end{array}
$$

Note the analogy of these equalities with the φ-rules.

Definition 13
Let d, $e \in NF$, we define $s''_e(d\,\sigma^j e)$ by induction on d as follows:

$$
\begin{array}{lcl}
s''_e(X\,\sigma^j b) & = & X\,\sigma^j b \\
s''_e(\mathtt{n}\,\sigma^j b) & = & \begin{cases} \mathtt{n-1} & \text{if } n > j \\ s'_e(\varphi^j_0\, b) & \text{if } n = j \\ \mathtt{n} & \text{if } n < j \end{cases} \\
s''_e((a\,c)\,\sigma^j b) & = & s''_e(a\,\sigma^j b)\, s''_e(c\,\sigma^j b) \\
s''_e((\lambda a)\,\sigma^j b) & = & \lambda s''_e(a\,\sigma^{j+1} b) \\
s''_e((\varphi^i_k a)\,\sigma^j b) & = & \begin{cases} (\varphi^i_k a)\,\sigma^j b & \text{if } j \leq k \\ \varphi^{i-1}_k a & \text{if } k < j < k+i \\ s'_e(\varphi^i_{k+1} a)\,\sigma^{k+1} s'_e(\varphi^i_0 b) & \text{if } j = k+i \\ \varphi^i_k(s''_e(a\,\sigma^{j+1-i} b)) & \text{if } j > k+i \end{cases} \\
s''_e((a\,\sigma^i c)\,\sigma^j b) & = & \begin{cases} (a\,\sigma^i c)\,\sigma^j b & \text{if } i > j \\ s''_e(a\,\sigma^{j+1} b)\,\sigma^i s''_e(c\,\sigma^{j+1-i} b) & \text{if } i \leq j \end{cases}
\end{array}
$$

Remark again the analogy of these equalities with the σ-rules. Only one does not fit the pattern: $s''_e((\varphi^i_k a)\,\sigma^j b) = s'_e(\varphi^i_{k+1} a)\sigma^{k+1} s'_e(\varphi^i_0 b)$ when $j = k+i$. The reason for treating this case separately is that only when $j = k+i$ an application of σ-φ *tr.2* creates a new φ-σ *tr.*-redex:

$$(\varphi^i_k\, a)\,\sigma^j\, b \longrightarrow_{\sigma\text{-}\varphi\text{-}tr.2} \varphi^i_k(a\,\sigma^{k+1}\, b) \longrightarrow_{\varphi\text{-}\sigma\text{-}tr} (\varphi^i_{k+1} a)\,\sigma^{k+1}(\varphi^i_0 b)$$

Definition 14
Let $d \in \Lambda s_{op}$, we define $s^*_e(d)$ by induction on d as follows:

$$
\begin{array}{lll}
s^*_e(X) = X & s^*_e(a\,b) = s^*_e(a)\, s^*_e(b) & s^*_e(\varphi^i_k a) = s'_e(\varphi^i_k s^*_e(a)) \\
s^*_e(\mathtt{n}) = \mathtt{n} & s^*_e(\lambda a) = \lambda s^*_e(a) & s^*_e(a\,\sigma^j b) = s''_e(s^*_e(a)\,\sigma^j\, s^*_e(b))
\end{array}
$$

In order to prove weak normalisation of s_e, we need to show that s'_e and s''_e define normal forms and this requires a powerful inductive hypothesis (see Lemmas 15 and 16) which uses S and N below:

Definition 15
The *set of sorts* is defined as $\mathcal{S} = \{V, B, \delta, \lambda, \sigma, \varphi\}$. The *sort* of a term a, denoted $S(a)$, is defined as: $S(X) = V$, $S(\mathtt{n}) = B$, $S(a\,b) = \delta$, $S(\lambda a) = \lambda$, $S(a\,\sigma^i b) = \sigma$, $S(\varphi^i_k a) = \varphi$. The *number* of a term c of sort σ or φ or V, denoted $N(c)$, is defined as $N(\varphi^i_k a) = k$, $N(a\,\sigma^j b) = j$ and $N(X) = 0$.

S and N really matter in deciding the existence of redexes:

Remark 1
Let $b \in NF$:

1. If $\varphi^i_k a \in NF$, $S(a) = S(b)$ and $N(a) = N(b)$, then $\varphi^j_k b \in NF$ for every $j \geq 1$.
2. If $a\,\sigma^j c \in NF$, $S(a) = S(b)$ and $N(a) = N(b)$, then $b\,\sigma^j c \in NF$.
3. If $\varphi^i_k a \in NF$, $S(a) = S(b)$ and $N(a) = N(b)$, then $b\,\sigma^{k+1} c \in NF$ for $c \in NF$.

Proof
By analysis of the redex at the root (cf. (Kamareddine & Ríos, 1996)). □

Lemma 15
If $a \in NF$ then $s'_e(\varphi^i_k a)$ is an s_e-normal form of $\varphi^i_k a$.
Moreover, if $s'_e(\varphi^i_k a) \neq \varphi^i_k a$ then $S(a) = S(s'_e(\varphi^i_k a))$ and when $S(a) = \sigma$ or $S(a) = \varphi$ we also have $N(a) = N(s'_e(\varphi^i_k a))$.

Proof
By induction on a. We only show the case $a = \varphi^j_l b$ and $l + j \leq k$.
Now, $s'_e(\varphi^i_k a) = s'_e(\varphi^i_k(\varphi^j_l b)) \overset{D.12}{=} \varphi^j_l s'_e(\varphi^i_{k+1-j} b)$.
If $s'_e(\varphi^i_{k+1-j} b) = \varphi^i_{k+1-j} b$, then $s'_e(\varphi^i_k a) = \varphi^j_l(\varphi^i_{k+1-j} b)$ wich is in normal form, beacuse $l < k+1-j$. If $s'_e(\varphi^i_{k+1-j} b) \neq \varphi^i_{k+1-j} b$, then our strong inductive hypothesis ensures $S(b) = S(s'_e(\varphi^i_{k+1-j} b))$. Note that, since $a = \varphi^j_l b \in NF$, b is neither an application nor an abstraction, also b is not a variable (otherwise $s'_e(\varphi^i_{k+1-j} b) = \varphi^i_{k+1-j} b$). Hence b is a $\sigma\varphi$-normal form, and we have $N(b) = N(s'_e(\varphi^i_k b))$. We conclude by Remark 1.1 that $\varphi^j_l s'_e(\varphi^i_k b) \in NF$. □

Lemma 16
If a, $b \in NF$ then $s''_e(a\,\sigma^j b)$ is an s_e-normal form of $a\,\sigma^j b$.
Moreover, if $s''_e(a\,\sigma^j b) \neq a\,\sigma^j b$ and $a \neq \mathtt{j}$ then:

1. If $a \neq \varphi^i_k c$ with $i + k = j$ then $S(a) = S(s''_e(a\,\sigma^j b))$ and when $S(a) = \sigma$ or $S(a) = \varphi$ we have furthermore $N(a) = N(s''_e(a\,\sigma^j b))$.
2. If $a = \varphi^i_k c$ with $i + k = j$ then $S(s''_e(a\,\sigma^j b)) = \sigma$ and $N(s''_e(a\,\sigma^j b)) = k + 1$.

Proof
By induction on a. The proof is similar to the proof of the previous lemma. □

Theorem 10 (Weak normalisation of s_e)

For every term $a \in \Lambda s_{op}$, $s_e^*(a)$ is an s_e-normal form of a. Hence, the s_e-calculus is weakly normalising.

Proof

By induction on a using Lemmas 15 and 16 and the fact that left members of s_e-rules are neither applications nor abstractions. $\square$

We have therefore a strategy to find s_e-normal forms. Furthermore, the strategy is *innermost* indeed: notice that for the out-of-the-pattern case we pointed out after Definition 13, the strategy remains innermost. In fact, for $j = k + i$, we had:

$$(\varphi^i_k\, a)\, \sigma^j\, b \longrightarrow_{\sigma\text{-}\varphi\text{-}tr.2}\ \varphi^i_k(a\, \sigma^{k+1}\, b) \longrightarrow_{\varphi\text{-}\sigma\text{-}tr}\ (\varphi^i_{k+1} a)\, \sigma^{k+1} (\varphi^i_0 b)\,,$$

and if $\varphi^i_k a \in NF$ and $b \in NF$ then, $a\, \sigma^{k+1}\, b \in NF$. Therefore, the only redex in $\varphi^i_k(a\, \sigma^{k+1}\, b)$ is the φ-σ-*transition*-redex at the root.

If we knew that s_e is SN, since we proved local confluence of s_e (cf. Theorem 6), we could apply Newman's Lemma to show the confluence of s_e. In the abscence of this information, we establish the following proposition.

Proposition 1

Let a, $b \in \Lambda s_{op}$, if $a \rightarrow_{s_e} b$ then $s_e^*(a) = s_e^*(b)$.

Proof

Induction on a showing first that $s_e^*(s_e^*(a)) = s_e^*(a)$, $s_e^*(\varphi^i_k a) = s_e^*(\varphi^i_k s_e^*(a))$ and $s_e^*(a\, \sigma^j\, b) = s_e^*(s_e^*(a)\, \sigma^j\, s_e^*(b))$ and that for every rule $L \rightarrow R$ of the s_e-calculus, $s_e^*(L) = s_e^*(R)$. This last statement should be first proved assuming that all the terms involved in the rules are s_e-nfs. This is easy for the s-rules, but for the other rules an enormous amount of elementary calculations is needed. Furthermore, for some rules it is necessary to assume that the fact hold for other rules, hence the importance of the chosen order for the proofs. This order works: φ-φ-*tr.1*, φ-φ-*tr.2*, σ-φ-*tr.1*, σ-φ-*tr.2*, φ-σ-*tr.*, σ-σ-*tr.*. More details in (Kamareddine & Ríos, 1996). $\square$

Theorem 11 (Confluence of s_e)

The s_e-calculus is confluent both on Λs_{op} and on Λs.

Proof

Since all the s_e-rules preserve closed terms, we just prove the theorem for Λs_{op}.

It is easy to show by induction on the length of the derivation and using Proposition 1 that for a, $b \in \Lambda s_{op}$, $a \twoheadrightarrow_{s_e} b$ implies $s_e^*(a) = s_e^*(b)$.

Let us suppose $a \twoheadrightarrow_{s_e} b$ and $a \twoheadrightarrow_{s_e} c$, hence $s_e^*(a) = s_e^*(b)$ and $s_e^*(a) = s_e^*(c)$. The theorem is therefore settled since $b \twoheadrightarrow_{s_e} s_e^*(b)$ and $c \twoheadrightarrow_{s_e} s_e^*(c)$. $\square$

Hence, for every term $a \in \Lambda s_{op}$ there exists (Theorem 10) a unique s_e-normal form that we denote $s_e(a)$. Hence, $s_e(a) = s_e^*(a)$ for every $a \in \Lambda s_{op}$, $s_e(\varphi^i_k b) = s'_e(\varphi^i_k b)$ for every $b \in NF$ and every $i \geq 1$, $k \geq 0$ and $s_e(c\, \sigma^j\, d) = s''_e(c\, \sigma^j\, d)$ for every c, $d \in NF$ and $j \geq 1$.

4 The calculus of the interpretation

Our aim is to apply the GIMES (Corollary 1) to obtain the confluence of the λs_e-calculus. Our interpretation function will be s_e. Coming back to the notation of Corollary 1, we intend to apply the GIMES with: $f = s_e$, $R = \rightarrow_{\lambda s_e}$, $R_1 = \rightarrow_{s_e}$ and $R_2 = \rightarrow_{\sigma-gen}$. In the previous section we proved Proposition 1 which evidently implies that Condition 2 of the GIMES is satisfied. In this section we are going to introduce the calculus of the interpretation. The set of the interpretation is, of course, NF. Therefore, we shall define R' on NF and prove that Conditions 1 and 3 of the GIMES are also satisfied. We postpone the confluence of R' until the next section.

Definition 16 (The interpretation reduction β')
For a, $b \in NF$, $a \rightarrow_{\beta'} b$ iff there exists $d \in \Lambda s_{op}$ such that $a \rightarrow_{\sigma-gen} d$ and $b = s_e(d)$.

We take $\rightarrow_{\beta'}$ as R'. Condition 1 of the GIMES is immediate:

Proposition 2
Let a, $b \in NF$, if $a \rightarrow_{\beta'} b$ then $a \twoheadrightarrow_{\lambda s_e} b$.

The following lemmas are needed to prove Proposition 3 which is Condition 3 of the GIMES.

Lemma 17
Let a, b, $c \in NF$.

1. If $a \twoheadrightarrow_{\beta'} b$ and $c \twoheadrightarrow_{\beta'} d$ then $a\, c \twoheadrightarrow_{\beta'} b\, d$.
2. If $a \twoheadrightarrow_{\beta'} b$, then $\lambda a \twoheadrightarrow_{\beta'} \lambda b$.

Proof
1. Prove first that if $a \rightarrow_{\beta'} b$, then $a\, c \rightarrow_{\beta'} b\, c$ and $c\, a \rightarrow_{\beta'} c\, b$. Then use a double induction. 2. Prove first that if $a \rightarrow_{\beta'} b$, then $\lambda a \rightarrow_{\beta'} \lambda b$. □

Lemma 18
If a is a $\sigma\varphi$-normal form and $a \rightarrow_{\sigma-gen} d$ then $S(a) = S(s_e(d))$ and $N(a) = N(s_e(d))$ (cf. Definition 15).

Proof
By induction on a using Remark 1 (cf. (Kamareddine & Ríos, 1996) for details). □

Lemma 19
For a, b, c, $e \in NF$ the following hold:

1. If $a \rightarrow_{\beta'} b$ and $\varphi^i_k a \in NF$ then $\varphi^i_k b \in NF$ and $\varphi^i_k a \rightarrow_{\beta'} \varphi^i_k b$.
2. If $a \rightarrow_{\beta'} b$ and $a\, \sigma^j c \in NF$ then $b\, \sigma^j c \in NF$ and $a\, \sigma^j c \rightarrow_{\beta'} b\, \sigma^j c$.
3. If $a \rightarrow_{\beta'} b$ and $c\, \sigma^j a \in NF$ then $c\, \sigma^j b \in NF$ and $c\, \sigma^j a \rightarrow_{\beta'} c\, \sigma^j b$.
4. If $a \twoheadrightarrow_{\beta'} b$ and $\varphi^i_k a \in NF$ then $\varphi^i_k b \in NF$ and $\varphi^i_k a \twoheadrightarrow_{\beta'} \varphi^i_k b$.
5. If $a \twoheadrightarrow_{\beta'} b$, $c \twoheadrightarrow_{\beta'} e$ and $a\, \sigma^j c \in NF$ then $b\, \sigma^j e \in NF$ and $a\, \sigma^j c \twoheadrightarrow_{\beta'} b\, \sigma^j e$.

Proof
Using Lemma 18 and Remark 1 (cf. (Kamareddine & Ríos, 1996) for details). □

Lemma 20
Let $a, b \in NF$ and $d \in \Lambda s_{op}$. The following holds:

1. If $a \to_{\sigma-gen} d$ then $s_e(\varphi^i_k a) \to_{\beta'} s_e(\varphi^i_k d)$.
2. If $a \to_{\sigma-gen} d$ then $s_e(a\,\sigma^j b) \to_{\beta'} s_e(d\,\sigma^j b)$.
3. If $b \to_{\sigma-gen} d$ then $s_e(a\,\sigma^j b) \twoheadrightarrow_{\beta'} s_e(a\,\sigma^j d)$.

Proof
By induction on a

1. Use Lemma 19.1, 2 and 3.
2. The previous item is required.
3. Use Lemma 17 and Lemma 19.3, 4 and 5, as well as the first item.

See (Kamareddine & Ríos, 1996) for more technical details. □

Lemma 21
For a, b, $c \in NF$ the following hold:

1. If $a \twoheadrightarrow_{\beta'} b$ then $s_e(\varphi^i_k a) \twoheadrightarrow_{\beta'} s_e(\varphi^i_k b)$.
2. If $a \twoheadrightarrow_{\beta'} b$ then $s_e(a\,\sigma^j c) \twoheadrightarrow_{\beta'} s_e(b\,\sigma^j c)$.
3. If $b \twoheadrightarrow_{\beta'} c$ then $s_e(a\,\sigma^j b) \twoheadrightarrow_{\beta'} s_e(a\,\sigma^j c)$.

Proof
Straightforward using Lemma 20. □

The following proposition states that Condition 3 of the GIMES is satisfied.

Proposition 3
Let a, $b \in \Lambda s_{op}$, if $a \to_{\sigma-gen} b$ then $s_e(a) \twoheadrightarrow_{\beta'} s_e(b)$.

Proof
By induction on a.

$a = c\,d$ **:** If the reduction is internal ($c \to_{\sigma-gen} c'$ or $d \to_{\sigma-gen} d'$), use IH and Lemma 17.1. If the reduction is at the root ($c = \lambda c'$ and $b = c'\,\sigma^1 d$) we have: $s_e((\lambda c')d) = (\lambda s_e(c'))s_e(d) \to_{\beta'} s_e(s_e(c')\,\sigma^1 s_e(d)) = s_e(c'\,\sigma^1 d)$.
$a = \lambda c$ **:** Use the IH and Lemma 17.2.
$a = \varphi^i_k c$ **:** Use the IH and Lemma 21.1.
$a = c\,\sigma^j d$ **:** Use IH and Lemma 21.2 or 21.3, if the reduction is within c or d respectively.

□

5 Confluence results

In this section we prove confluence for the calculus of the interpretation $(NF, \to_{\beta'})$ in order to obtain the confluence of the λs_e-calculus via the GIMES. The confluence of $(NF, \to_{\beta'})$ is obtained via a parallelisation à la Tait-Martin-Löf (cf. (Barendregt, 1984)) defined as follows:

Definition 17

Let a, b, c, d, $a_1, \ldots, a_n \in NF$. The reduction $\Rightarrow$ is defined on NF by:

$$(REFL) \qquad a \Rightarrow a \qquad\qquad (SPHI) \quad \frac{a_h \Rightarrow b_h \qquad 1 \le h \le n}{\overline{\sigma\varphi}(a_1, \ldots, a_n)X \Rightarrow \overline{\sigma\varphi}(b_1, \ldots, b_n)X}$$

$$(ABST) \quad \frac{a \Rightarrow b}{\lambda a \Rightarrow \lambda b} \qquad\qquad (BETA) \quad \frac{a \Rightarrow c \quad\; b \Rightarrow d}{(\lambda a)\, b \Rightarrow s_e(c\,\sigma^1 d)}$$

$$(APPL) \quad \frac{a \Rightarrow c \quad\; b \Rightarrow d}{a\, b \Rightarrow c\, d}$$

We remark that *SPHI* is a rule scheme where $\overline{\sigma\varphi}$ ranges over normal skeletons.

To prove that the transitive closures of $\rightarrow_{\beta'}$ and $\Rightarrow$ coincide, we establish the following two lemmas.

Lemma 22
Let a, $b \in NF$, if $a \Rightarrow b$ then $a \twoheadrightarrow_{\beta'} b$.

Proof
By induction on the length of the deduction $a \Rightarrow b$. We only treat two cases according to the last rule applied in this deduction:

SPHI: $a = \overline{\sigma\varphi}(a_1, \ldots, a_n)X$, $b = \overline{\sigma\varphi}(b_1, \ldots, b_n)X$ and $a_h \Rightarrow b_h$ for $1 \le h \le n$.
By IH, $a_h \twoheadrightarrow_{\beta'} b_h$, and we use the following (proved by induction on the length of $\overline{\sigma\varphi}$ and using Lemma 19.4 and 5):
Fact: For every normal skeleton $\overline{\sigma\varphi}$ of arity n and for every a_h, $b_h \in NF$ $(1 \le h \le n)$, if $a_h \twoheadrightarrow_{\beta'} b_h$ for $1 \le h \le n$, then $\overline{\sigma\varphi}(a_1, \ldots, a_n)X \twoheadrightarrow_{\beta'} \overline{\sigma\varphi}(b_1, \ldots, b_n)X$.

BETA: $a = (\lambda a')b'$, $b = s_e(c'\sigma^1 d')$, $a' \Rightarrow c'$ and $b' \Rightarrow d'$.
By IH, $a' \twoheadrightarrow_{\beta'} c'$ and $b' \twoheadrightarrow_{\beta'} d'$ and therefore $(\lambda a')b' \rightarrow_{\beta'} s_e(a'\sigma^1 b') \overset{L\,21.2}{\twoheadrightarrow_{\beta'}} s_e(c'\sigma^1 b') \overset{L\,21.3}{\twoheadrightarrow_{\beta'}} s_e(c'\sigma^1 d')$.

□

Remark 2
For $a_1, \ldots, a_n \in \Lambda s_{op}$ and $\overline{\sigma\varphi}$ the skeleton of a normal $\sigma\varphi$-segment, we have $s_e(\overline{\sigma\varphi}(a_1, \ldots, a_n)X) = \overline{\sigma\varphi}(s_e(a_1), \ldots, s_e(a_n))X$.

Proof
Because $\overline{\sigma\varphi}(a_1, \ldots, a_n)X \twoheadrightarrow_{s_e} \overline{\sigma\varphi}(s_e(a_1), \ldots, s_e(a_n))X$ which is a (unique) s_e-nf.
□

Lemma 23
Let $a \in NF$ and $d \in \Lambda s_{op}$, if $a \rightarrow_{\sigma-gen} d$ then $a \Rightarrow s_e(d)$.

Proof
By induction on a. As an example, we treat the case $a = \overline{\sigma\varphi}(a_1, \ldots, a_n)X$. The reduction must occur within some a_i, hence $d = \overline{\sigma\varphi}(a_1, \ldots, a'_i, \ldots, a_n)$ such that $a_i \rightarrow_{\sigma-gen} a'_i$. By IH, $a_i \Rightarrow s_e(a'_i)$ and, since $a_h \Rightarrow a_h$, applying rule *SPHI*:

$a \Rightarrow \overline{\sigma\varphi}(a_1, \ldots, s_e(a'_i), \ldots, a_n) \overset{R\,2}{=} s_e(\overline{\sigma\varphi}(a_1, \ldots, a'_i, \ldots, a_n))$ □

Lemma 24
The transitive closures of $\to_{\beta'}$ and $\Rightarrow$ coincide, i.e. $\twoheadrightarrow_{\beta'} = \Rightarrow^*$.

Proof
If $a \to_{\beta'} b$ then $a \to_{\sigma-gen} d$ and $b = s_e(d)$ and, by Lemma 23, $a \Rightarrow b$. Therefore, $\to_{\beta'} \subseteq \Rightarrow$. Now, by Lemma 22, $\Rightarrow \subseteq \twoheadrightarrow_{\beta'}$, hence $\to_{\beta'} \subseteq \Rightarrow \subseteq \twoheadrightarrow_{\beta'}$. Therefore, $\twoheadrightarrow_{\beta'} = \Rightarrow^*$.
□

To prove that $\Rightarrow$ is SCR we must first establish some lemmas (see (Kamareddine & Ríos, 1996) for details).

Lemma 25
For every $i \geq 1$, $k \geq 0$ and normal skeleton $\overline{\sigma\varphi}$ of arity n, there exists a normal skeleton $\overline{\sigma\varphi_1}$, m, $i_1, \ldots, i_m$, $k_1, \ldots, k_m$ such that $0 \leq m \leq n$, $i_h \geq 1$ and $k_h \geq 0$ $(1 \leq h \leq m)$ and such that for every $a_1, \ldots, a_n \in NF$ the following holds:
$s_e(\varphi^i_k\, \overline{\sigma\varphi}(a_1, \ldots, a_n)X) = \overline{\sigma\varphi_1}(s_e(\varphi^{i_1}_{k_1} a_1), \ldots, s_e(\varphi^{i_m}_{k_m} a_m), a_{m+1}, \ldots, a_n)X$

Proof
By induction on the length of the skeleton $\overline{\sigma\varphi}$. □

Lemma 26
For every $j \geq 1$ and normal skeleton $\overline{\sigma\varphi}$ of arity n, there exists a normal skeleton $\overline{\sigma\varphi_2}$, m, $i_1, \ldots, i_m$ such that $0 \leq m \leq n$, $i_h \geq 1$ $(1 \leq h \leq m)$ and one and only one of the following holds:

- There exist $i_0 \geq 1$, p, $i_{m+1}, \ldots, i_p$, $k_{m+1}, \ldots, k_p$ such that $m \leq p \leq n$, $i_h \geq 1$, $k_h \geq 0$ $(m+1 \leq h \leq p)$ and for every $a_1, \ldots, a_n, c \in NF$, the following holds:
  $s_e(\overline{\sigma\varphi}(a_1, \ldots, a_n)X\ \sigma^j\ c) =$
  $\overline{\sigma\varphi_2}(s_e(b_1), \ldots, s_e(b_m), s_e(\varphi^{i_0}_0 c),\ s_e(d_{m+1}), \ldots, s_e(d_p), a_{p+1}, \ldots, a_n)X$
  where $b_l = a_l \sigma^{i_l} c$ and $d_l = \varphi^{i_l}_{k_l} a_l$
- For every $a_1, \ldots, a_n, c \in NF$, the following holds:
  $s_e(\overline{\sigma\varphi}(a_1, \ldots, a_n)X\ \sigma^j\ c) =$
  $\overline{\sigma\varphi_2}(s_e(a_1\sigma^{i_1}c), \ldots, s_e(a_m\sigma^{i_m}c),\ c, a_{m+1}, \ldots, a_n)X$
- For every $a_1, \ldots, a_n, c \in NF$, the following holds:
  $s_e(\overline{\sigma\varphi}(a_1, \ldots, a_n)X\ \sigma^j\ c) =$
  $\overline{\sigma\varphi_2}(s_e(a_1\sigma^{i_1}c), \ldots, s_e(a_m\sigma^{i_m}c), a_{m+1}, \ldots, a_n)X$

Proof
By induction on the length of the skeleton $\overline{\sigma\varphi}$. □

Lemma 27
Let $a, b \in NF$, if $a \Rightarrow b$ then $s_e(\varphi^i_k a) \Rightarrow s_e(\varphi^i_k b)$.

Proof
By induction on the length of the deduction $a \Rightarrow b$. If the last rule is, e.g.:

SPHI: Hence $a = \overline{\sigma\varphi}(a_1, \ldots, a_n)X$, $b = \overline{\sigma\varphi}(b_1, \ldots, b_n)X$ and $a_h \Rightarrow b_h$ for all h. By Lemma 25 we have
$s_e(\varphi^i_k\, \overline{\sigma\varphi}(a_1, \ldots, a_n)X) = \overline{\sigma\varphi_1}(s_e(\varphi^{i_1}_{k_1} a_1), \ldots, s_e(\varphi^{i_m}_{k_m} a_m), a_{m+1}, \ldots, a_n)X.$
By IH, $s_e(\varphi^{i_h}_{k_h} a_h) \Rightarrow s_e(\varphi^{i_h}_{k_h} b_h)$ for $h \leq m$ and, since $a_h \Rightarrow b_h$ for all h, in

particular for $m < h \leq n$, we apply rule *SPHI* to get
$\overline{\sigma\varphi_1}(s_e(\varphi^{i_1}_{k_1} a_1), \ldots, s_e(\varphi^{i_m}_{k_m} a_m), a_{m+1}, \ldots, a_n)X \Rightarrow$
$\overline{\sigma\varphi_1}(s_e(\varphi^{i_1}_{k_1} b_1), \ldots, s_e(\varphi^{i_m}_{k_m} b_m), b_{m+1}, \ldots, b_n)X \overset{L\,25}{=} s_e(\varphi^i_k\, \overline{\sigma\varphi}(b_1, \ldots, b_n)X)$

BETA: Hence $a = (\lambda a_1)a_2$, $b = s_e(b_1\sigma^1 b_2)$, $a_1 \Rightarrow b_1$ and $a_2 \Rightarrow b_2$.
$s_e(\varphi^i_k((\lambda a_1)a_2)) = (\lambda s_e(\varphi^i_{k+1} a_1))s_e(\varphi^i_k a_2) \overset{IH}{\Rightarrow} s_e(s_e(\varphi^i_{k+1} b_1)\sigma^1 s_e(\varphi^i_k b_2)) =$
$s_e((\varphi^i_{k+1} b_1)\sigma^1(\varphi^i_k b_2)) = s_e(\varphi^i_k(b_1\sigma^1 b_2)) = s_e(\varphi^i_k s_e(b_1\sigma^1 b_2))$

□

Corollary 4
Let $a, b \in NF$ such that $\varphi^i_k a, \varphi^i_k b \in NF$, if $a \Rightarrow b$ then $\varphi^i_k a \Rightarrow \varphi^i_k b$.

Lemma 28
Let $a, b, c \in NF$, if $b \Rightarrow c$ then $s_e(a\,\sigma^j b) \Rightarrow s_e(a\,\sigma^j c)$.

Proof
By induction on a (see (Kamareddine & Ríos, 1996) for details). □

Lemma 29
Let $a, b, c, d \in NF$, if $a \Rightarrow b$ and $c \Rightarrow d$ then $s_e(a\,\sigma^j c) \Rightarrow s_e(b\,\sigma^j d)$.

Proof
By induction on the length of the deduction $a \Rightarrow b$. If the last rule is, e.g.:

BETA: Hence $a = (\lambda a_1)a_2$, $b = s_e(b_1\sigma^1 b_2)$, $a_1 \Rightarrow b_1$ and $a_2 \Rightarrow b_2$.
$s_e(((\lambda a_1)a_2)\,\sigma^j c) = (\lambda s_e(a_1\,\sigma^{j+1} c))s_e(a_2\,\sigma^j c) \overset{IH\,\&\,BETA}{\Rightarrow}$
$s_e(s_e(b_1\,\sigma^{j+1} d)\sigma^1 s_e(b_2\sigma^j d)) =$
$s_e((b_1\,\sigma^{j+1} d)\sigma^1(b_2\sigma^j d)) = s_e((b_1\sigma^1 b_2)\,\sigma^j d) = s_e(s_e(b_1\sigma^1 b_2)\,\sigma^j d)$

SPHI: Hence $a = \overline{\sigma\varphi}(a_1, \ldots, a_n)X$, $b = \overline{\sigma\varphi}(b_1, \ldots, b_n)X$, $a_h \Rightarrow b_h$ $(1 \leq h \leq n)$.
Lemma 26 offers 3 possibilities which are treated analogously. We choose the second one:
$s_e(\overline{\sigma\varphi}(a_1, \ldots, a_n)X\ \sigma^j\ c) =$
$\overline{\sigma\varphi_2}(s_e(a_1\sigma^{i_1} c), \ldots, s_e(a_m\sigma^{i_m} c),\ c, a_{m+1}, \ldots, a_n)X$
by IH, $s_e(a_h\sigma^{i_h} c) \Rightarrow s_e(b_h\sigma^{i_h} d)$ and as $c \Rightarrow d$ and $a_h \Rightarrow b_h$ for $m+1 \leq h \leq n$,
SPHI gives: $\overline{\sigma\varphi_2}(s_e(a_1\sigma^{i_1} c), \ldots, s_e(a_m\sigma^{i_m} c),\ c, a_{m+1}, \ldots, a_n)X \Rightarrow$
$\overline{\sigma\varphi_2}(s_e(b_1\sigma^{i_1} d), \ldots, s_e(b_m\sigma^{i_m} d),\ d, b_{m+1}, \ldots, b_n)X$. Finally, by Lemma 26,
$s_e(\overline{\sigma\varphi}(b_1, \ldots, b_n)X\ \sigma^j\ d) =$
$\overline{\sigma\varphi_2}(s_e(b_1\sigma^{i_1} d), \ldots, s_e(b_m\sigma^{i_m} d),\ d, b_{m+1}, \ldots, b_n)X$.
Remark that, to check the first option of Lemma 26, Lemma 27 is needed.

□

Theorem 12
The parallelisation $\Rightarrow$ is strongly confluent.

Proof
By induction on the length of the deduction $a \Rightarrow b$. We just study two cases for the last rule applied in this deduction:

SPHI: Hence $a = \overline{\sigma\varphi}(a_1, \ldots, a_n)X$, $b = \overline{\sigma\varphi}(b_1, \ldots, b_n)X$ and $a_h \Rrightarrow b_h$ for all h. Remark that $c = \overline{\sigma\varphi}(c_1, \ldots, c_n)X$ and $a_h \Rrightarrow c_h$ for all h, since the last rule to obtain $\overline{\sigma\varphi}(a_1, \ldots, a_n)X \Rrightarrow c$ must be either *SPHI* or *REFL*. By IH there exist d_h such that $b_h \Rrightarrow d_h$ and $c_h \Rrightarrow d_h$ for all h. Take $d = \overline{\sigma\varphi}(d_1, \ldots, d_n)X$.

BETA: Hence $a = (\lambda a_1)a_2$, $b = s_e(b_1\sigma^1 b_2)$, $a_1 \Rrightarrow b_1$ and $a_2 \Rrightarrow b_2$. There are two possibilities for c according to the last rule applied to obtain $a \Rrightarrow c$. We only treat the case where the last rule is *BETA*:
Hence $c = s_e(c_1\sigma^1 c_2)$, with $a_1 \Rrightarrow c_1$ and $a_2 \Rrightarrow c_2$, then by IH there exists d_1, d_2 such that $b_1 \Rrightarrow d_1$, $c_1 \Rrightarrow d_1$, $b_2 \Rrightarrow d_2$ and $c_2 \Rrightarrow d_2$. Take $d = s_e(d_1\sigma^1 d_2)$ and use Lemma 29.

□

Proposition 4
The calculus of the interpretation $(NF, \rightarrow_{\beta'})$ is confluent.

Proof
By Theorem 12, $\Rrightarrow$ is SCR, and by Lemma 1, also $\Rrightarrow^*$ is SCR. Hence, by Lemma 24, $\twoheadrightarrow_{\beta'}$ is SCR, and so $\rightarrow_{\beta'}$ is confluent. □

Theorem 13
The λs_e-calculus is confluent on Λs_{op}.

Proof
All the conditions hold (see our four propositions) and the GIMES (Corollary 1) can be applied as proposed at the beginning of Section 4. □

Corollary 5
The λs_e-calculus is confluent on Λs.

Finally, we show that λs_e is correct with respect to the λ-calculus, i.e. that all λs_e-derivations beginning and ending with pure terms can also be obtained in the λ-calculus.

Theorem 14 (Soundness)
For a, $b \in \Lambda$, if $a \twoheadrightarrow_{\lambda s_e} b$ then $a \twoheadrightarrow_{\beta} b$.

Proof
First we show by induction on a that for all a, $b \in \Lambda$, $a \rightarrow_{\beta'} b$ iff $a \rightarrow_{\beta} b$ and deduce that $a \twoheadrightarrow_{\beta'} b$ iff $a \twoheadrightarrow_{\beta} b$. Then we show by induction on the length of the derivation $a \twoheadrightarrow_{\lambda s_e} b$ using Prpositions 1 and 3 that for all a, $b \in \Lambda s_{op}$, if $a \twoheadrightarrow_{\lambda s_e} b$ then $s_e(a) \twoheadrightarrow_{\beta'} s_e(b)$. Now we are done because a, $b \in \Lambda \subset NF$. □

Conclusion

We think that λs is an interesting alternative to calculi of explicit substitutions in the $\lambda\sigma$-style: it preserves SN (cf. (Kamareddine & Ríos, 1995a)), has a confluent extension on open terms (cf. Theorem 13) and simulates one step β-reduction (cf. Lemma 14). Two important questions are still open:

1. Is the s_e-calculus strongly normalising?
2. Does the λs_e-calculus preserve SN?

If the second question could be decided positively, λs_e would be the answer to the two open problems in (Muñoz Hurtado, 1996), namely, a confluent (on open terms) calculus of explicit substitutions that preserves strong normalisation which

1. reduces substitution redexes before β-redexes.
2. admits interaction of substitutions.

We remark that SN of s_e would also shorten the proof of confluence that we have given here: most of the results of Section 3 become trivial in the presence of SN.

Finally, from a computational point of view, the lack of SN is not a major problem, since the s_e-calculus has been shown weakly normalising and an effective strategy to evaluate normal forms has been proposed.

However, from a theoretical point of view, the strong normalisation of the s_e-calculus is an important feature and seems a very difficult problem which remains still a challenge to the rewriting community. Zantema showed in a private communication, that the σ-σ-transition rule terminates. He considered the infinite Term Rewriting Structure TRS (with this rule), ranging over an infinite signature $\{\,\sigma^i, i > 0\}$. He showed strong normalisation of this TRS (call it S) by showing weak normalisation and using the following lemma (cf. (Klop, 1992)):

Lemma 30
Any reduction relation $\rightarrow$ on a set T satisfying 1,2, and 3 is strongly normalising:

1. $\rightarrow$ is weakly normalising.
2. $\rightarrow$ is WCR.
3. $\rightarrow$ is increasing, i.e., there exists $f : T \longrightarrow \mathbb{N}$ s.t. $a \rightarrow b \Rightarrow f(a) < f(b)$.

For S, 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 S, Zantema establishes first two lemmas:

Lemma 31
Let $b = ((\cdots(a\sigma^{i_1}a_1)\sigma^{i_2}a_2)\sigma^{i_3}a_3)\cdots\sigma^{i_n}a_n$, where a is either a variable or its root is not σ^q, and $i_1 > i_2 > ... > i_{n-1}$, $i_{n-1} \leq i_n$. Then
$b \rightarrow^+ ((\cdots(a\sigma^{j_1}b_1)\sigma^{j_2}b_2)\sigma^{j_3}b_3)\cdots\sigma^{j_n}b_n$, where $j_1 > j_2 > j_3 ... > j_{n-1} > j_n = i_{n-1}$, and for every $r = 1, \cdots, n$ either $b_r = a_p$ for some $p \leq n$ or $b_r = a_p\sigma^k a_n$ for some $p < n$ and some k.

Proof
By induction on n. At the top level, $b \rightarrow ((....\sigma^{i_n+1}a_n)\sigma^{i_{n-1}}(a_{n-1}\sigma^k a_n))$. □

Lemma 32
Let $b = ((..(a\sigma^{i_1}a_1)\sigma^{i_2}a_2)\sigma^{i_3}a_3)\ldots\sigma^{i_n}a_n$, where a is either a variable or its root is not σ^q. Then $b \rightarrow^* ((\cdots(a\sigma^{j_1}b_1)\sigma^{j_2}b_2)\sigma^{j_3}b_3)\cdots\sigma^{j_n}b_n$, where $j_1 > j_2 > j_3 ... > j_{n-1} > j_n$, and for every $r = 1, \cdots, n$ the term b_r can be written as
$b_r = (..(a_{c(r,1)}\sigma a_{c(r,2)})\sigma a_{c(r,2)})...\sigma a_{c(r,n)}$ for $1 \leq c(r,1) < c(r,2) < ... < c(r,n) \leq n$, where σ stands for arbitrary σ^k.

Proof
Induction on n using Lemma 31. □

Lemma 33 (Weak normalisation of S)
S is weakly normalising.

Proof
By induction on the size of the term: assume there is a term b not having a normal form for which every term of size smaller than b admits a normal form. Apply Lemma 32 to this term. Note that a and $b_1, b_2, ...b_n$ are all smaller than b, hence admit a normal form. Replace a and $b_1, b_2, \cdots, b_n$ by their normal forms in the term $((..(a\sigma^{j_1}b_1)\sigma^{j_2}b_2)\sigma^{j_3}b_3)\cdots\sigma^{j_n}b_n$, yielding a normal form of b, contradiction. □

Zantema correctly comments that weak normalisation of this TRS does not follow from weak normalisation of the whole s_e-calculus (cf. Theorem 10). We note moreover that his proof of weak normalisation differs from ours which provides an effective strategy to calculate normal forms. Furthermore, Zantema notes that his proof above is the first one that establishes strong normalisation from weak normalisation. Finally, he remarks that Lemma 30 cannot be used to establish strong normalisation of s_e from its weak normalisation because the full system is easily proved to be non-increasing.

References

Abadi, M., Cardelli, L., Curien, P.-L., & Lévy, J.-J. (1991). Explicit Substitutions. *Journal of functional programming*, **1**(4), 375–416.

Barendregt, H. (1984). *The Lambda Calculus : Its Syntax and Semantics (revised edition)*. North Holland.

Benaissa, Z., Briaud, D., Lescanne, P., & Rouyer-Degli, J. (1995). λv, a calculus of explicit substitutions which preserves strong normalisation. *Personal communication.*

Bloo, R. (1995). *Preservation of Strong Normalisation for Explicit Substitution.* Tech. rept. 95-08. Department of Mathematics and Computing Science, Eindhoven University of Technology.

Curien, P.-L. (1986). *Categorical Combinators, Sequential Algorithms and Functional Programming.* Pitman. Revised edition : Birkhäuser (1993).

Curien, P.-L., Hardin, T., & Lévy, J.-J. (1992). *Confluence properties of weak and strong calculi of explicit substitutions.* Tech. rept. RR 1617. INRIA, Rocquencourt. To appear in the JACM.

de Bruijn, N. (1972). 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.

de Bruijn, N. G. (1978). *A namefree lambda calculus with facilities for internal definition of expressions and segments.* Tech. rept. TH-Report 78-WSK-03. Department of Mathematics, Eindhoven University of Technology.

Hardin, T. (1989). Confluence Results for the Pure Strong Categorical Logic CCL : λ-calculi as Subsystems of CCL. *Theoretical computer science*, **65**(2), 291–342.

Hardin, T., & Lévy, J.-J. (1989). A Confluent Calculus of Substitutions. *France-japan artificial intelligence and computer science symposium*, December.

Kamareddine, F., & Nederpelt, R. P. (1993). On stepwise explicit substitution. *International journal of foundations of computer science*, **4(3)**, 197–240.

Kamareddine, F., & Nederpelt, R. P. (1995). Generalising reduction in the λ-calculus. *Journal of functional programming*, **5(4)**, 637–651.

Kamareddine, F., & Ríos, A. (1995a). A λ-calculus à la de Bruijn with explicit substitutions. Proceedings of PLILP'95. *Lecture notes in computer science*, **982**, 45–62.

Kamareddine, F., & Ríos, A. (1995b). *The λs-calculus: its typed and its extended versions.* Tech. rept. Department of Computing Science, University of Glasgow.

Kamareddine, F., & Ríos, A. (1996). *The confluence of the λs_e-calculus via a generalized interpretation method.* Tech. rept. Glasgow University.

Klop, J.-W. (1992). Term rewriting systems. *Handbook of logic in computer science*, **II**.

Melliès, P.-A. (1995). Typed λ-calculi with explicit substitutions may not terminate in Proceedings of TLCA'95. *Lecture Notes in Computer Science*, **902**.

Muñoz Hurtado, C. A. (1996). Confluence and preservation of strong normalisation in an explicit substitutions calculus. *Proceedings of the IEEE conference on Logics in Computer Science.*

Ríos, A. (1993). *Contribution à l'étude des λ-calculs avec substitutions explicites.* Ph.D. thesis, Université de Paris 7.