The confluence of the λs_e-calculus via a generalized interpretation method *

Fairouz Kamareddine and Alejandro Ríos †

July 5, 1996

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 [KR95a], 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 ([KR95a]), is extended in this paper to a confluent calculus on open terms: the λs_e-caculus. Since the establishment of the results of this paper[1], another calculus, $\lambda\zeta$, came into being in [MH95] 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 generalization of the interpretation method (cf. [Har89] and [CHL92]) to a technique which uses weak normal forms (instead of strong ones). This technique is general enough to apply to many reduction systems and we consider it as a powerful tool to obtain confluence.

Strong normalisation of the corresponding calculus of substitutions s_e, is left as a challenging problem to the rewrite community but its weak normalisation is established 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 been an interest in formalising substitution explicitly; various calculi including new operators to denote substitution have been proposed. Amongst these calculi we mention $C\lambda\xi\phi$ (cf. [dB78]); the calculi of categorical combinators (cf. [Cur86]); $\lambda\sigma$, $\lambda\sigma_{\Uparrow}$, $\lambda\sigma_{SP}$ (cf. [ACCL91], [CHL92], [Río93]) referred to as the $\lambda\sigma$-family; $\varphi\sigma BLT$ (cf. [KN93]); $\lambda\upsilon$ (cf. [BBLRD95]), a descendant of the $\lambda\sigma$-family; λs (cf. [KR95a]); $\lambda\texttt{exp}$ (cf. [Blo95]) and $\lambda\zeta$ (cf. [MH95]).

These calculi (except $\lambda\texttt{exp}$) 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

*This work was carried out under EPSRC grant GR/K25014.

†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

[1]The proof of confluence of the λs_e-calculus presented here was achieved in July 1995

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. [KR95a]) 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. [HL89] and [CHL92]), 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. [Mel95]). There are also calculi which satisfy the preservation property: the $\lambda\upsilon$-calculus (cf. [BBLRD95]), but this calculus is not confluent on open terms. Moreover, in order to get a confluent extension, the introduction of a composition operator for substitutions seems unavoidable, but precisely this operator is the cause of the non-preservation of strong normalisation as shown in [Mel95].

We proved in [KR95a] that λs preserves strong normalisation and proposed the extension λs_e in [KR95b], where we proved its local confluence on open terms and the weak normalisation (every term has at least one normal form) of the corresponding calculus 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. [MH95]) came into being which preserves strong normalisation and is itself strongly normalising and confluent on open terms. 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 [MH95], 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. [Har89], [CHL92]), 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 generalization 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 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. [KR95a] and [KR95b]) 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 $\lambda\sigma$, $\lambda\upsilon$ 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.

In section 6 we 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.

1 The Generalized Interpretation Method

We begin by introducing the notation we shall use throughout this paper concerning rewriting and we recall the definitions of the essential properties of the reduction systems.

Definition 1 *Let A be a set and R a binary relation on A, i.e. a subset of $A \times A$. 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). We denote R^* or $\twoheadrightarrow_R$ or just $\twoheadrightarrow$ or $\to^*$ the reflexive and transitive closure of R. We denote R^+ or just $\to^+$ 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.*

1. *R is* locally confluent *or WCR (*weakly Church-Rosser*) when*

$$\forall a,b,c \in A \ \exists d \in A \ ((a \to b \wedge a \to c) \Rightarrow (b \twoheadrightarrow d \wedge c \twoheadrightarrow d)).$$

2. *R is* confluent *or CR (*Church-Rosser*) when*

$$\forall a,b,c \in A \ \exists d \in A \ ((a \twoheadrightarrow b \wedge a \twoheadrightarrow c) \Rightarrow (b \twoheadrightarrow d \wedge c \twoheadrightarrow d)).$$

3. *R is* strongly confluent *or SCR when*

$$\forall a,b,c \in A \ \exists d \in A \ ((a \to b \wedge a \to c) \Rightarrow (b \to d \wedge c \to d)).$$

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 *or WN if every $a \in A$ has an R-normal form.*

R is strongly normalising *or SN if there is no infinite sequence $(a_i)_{i \geq 0}$ in A such that $a_i \to a_{i+1}$ for all $i \geq 0$.*

Remark 1 *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.

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

Proof: See [Bar84], lemma 3.3.2. □

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

Proof: See [Bar84], proposition 3.1.25.
G. Huet gives another proof using the Principle of noetherian induction (cf. [Hue80]). □

We state now the interpretation method we wish to generalize. This method was first identified in [Har89], where it was used for the categorical combinators. In [CHL92], 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 [Río93], it was used to the prove the confluence of the $\lambda\sigma_{SP}$-calculus on semi-closed terms. Finally, in [BBLRD95] it was used to prove the confluence of $\lambda\upsilon$ and we shall use it in this article to prove the confluence of the λs-calculus.

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.

Proof: It is easy (cf. [CHL92], lemma 1.1) and similar to the proof of the generalized interpretation method given below. □

Lemma 4 (Generalized interpretation method (GIM))
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 $f : A \to B$ 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. *$a \to_{R_1} b \;\Rightarrow\; f(a) = f(b)$ (syntactic identity)*
3. *$a \to_{R_2} b \;\Rightarrow\; f(a) \twoheadrightarrow_{R'} f(b)$*

then R' is confluent iff R is confluent.

Proof:

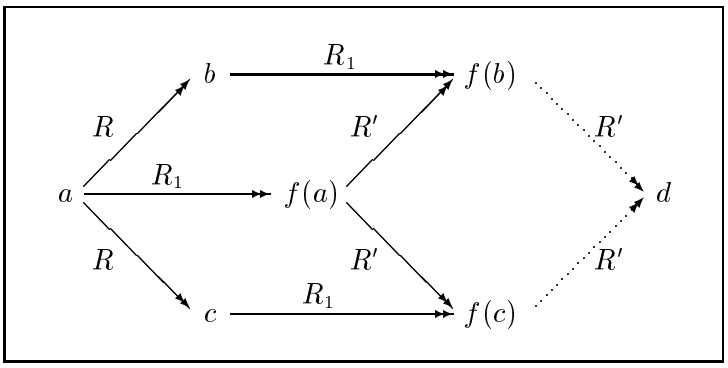

Figure 1: Generalized interpretation method

($\Rightarrow$) This is the implication which is usually useful. The proof is depicted in Figure 1. Let us suppose $a \twoheadrightarrow_R b$ and $a \twoheadrightarrow_R c$. It is easy to show by induction on the length of the derivation and using hypotheses 2 and 3 that $a' \twoheadrightarrow_R b'$ implies $f(a') \twoheadrightarrow_{R'} f(b')$. Hence, $f(a) \twoheadrightarrow_{R'} f(b)$ and $f(a) \twoheadrightarrow_{R'} f(c)$. Now we use the confluence of R' to obtain d such that $f(b) \twoheadrightarrow_{R'} d$ and $f(c) \twoheadrightarrow_{R'} d$. Finally the first hypothesis ensures $f(b) \twoheadrightarrow_R d$ and $f(c) \twoheadrightarrow_R d$. Therefore, $b \twoheadrightarrow_R d$ and $c \twoheadrightarrow_R d$.

($\Leftarrow$) Let a, b, $c \in B$ and let us assume $a \twoheadrightarrow_{R'} b$ and $a \twoheadrightarrow_{R'} c$. The first hypothesis guarantees $a \twoheadrightarrow_R b$ and $a \twoheadrightarrow_R c$ and confluence of R provides d such that $b \twoheadrightarrow_R d$ and $b \twoheadrightarrow_R d$. As before, hypotheses 2 and 3 give $f(b) \twoheadrightarrow_{R'} f(d)$ and $f(c) \twoheadrightarrow_{R'} f(d)$. And the lemma is settled because a, b, c being normal forms, we have $a = f(a)$, $b = f(b)$ and $c = f(c)$. $\Box$

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

We end this section by remarking that the GIM lemma really generalizes the interpretation method. In fact, in the particular case when R_1 is confluent and SN, R_1-normal forms exist and are unique. Hence there is only one f such that $f(a)$ is a normal form of a, namely $f(a) = R_1(a)$. Moreover, in this case the second hypothesis of the GIM lemma is superfluous.

2 The calculi

The results contained in this sections have already been proved in [KR95a] and [KR95b]. Nevertheless, since we wish to make this article self-contained we shall include here abridged proofs of the results concerning confluence. However, we offer here an independent proof of SN of s, whereas in [KR95a] we derived it from the SN of the σ-calculus.

We divide this section in three parts. In the first subsection we recall the classical λ-calculus à la de Bruijn and some of its properties, in particular the ones that give origin to the rules of the λs_e-calculus. In the second subsection we recall the λs-calculus and its properties. In the third one we introduce the λs_e-calculus, give some motivation for its rules and compare λs_e with $\lambda\sigma$ and $\lambda\upsilon$ by showing that the translations given in [KR95a] for λs, when extended to λs_e preserve equivalences. Finally, we discuss briefly the amount of reductions needed to simulate some β-contractions in the three calculi.

2.1 The classical λ-calculus in de Bruijn notation

We assume the reader familiar with de Bruijn notation. Let us just say here that de Bruijn indices (or numbers) are used to make the bindings explicit: to find the λ which binds a variable represented by the number $\mathtt{n}$ you must travel upwards in the tree associated with the term and choose the n-th λ you find. For instance, $\lambda x.\lambda y.xy$ is written using de Bruijn indices as $\lambda\lambda(\mathtt{21})$ and $\lambda x.\lambda y.(x(\lambda z.zx))y$ is written as $\lambda(\lambda(\mathtt{2}(\lambda(\mathtt{13}))\mathtt{1}))$. Finally, to translate free variables, you must assume a fixed ordered list of binders and prefix the term to be translated with this list. For instance, if the list (written from left to right) is $\cdots, \lambda z, \lambda y, \lambda x$ then the term $\lambda x.yz$ translates as $\lambda\mathtt{34}$ whereas $\lambda x.zy$ translates as $\lambda\mathtt{43}$.

The interest in introducing de Bruijn indices is that they avoid clashes of variable names and therefore neither α-conversion nor Barendregt's convention are needed. Here is the syntax of the λ-calculus à la de Bruijn.

Definition 4 *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 must identify amongst the numbers of the term a those that correspond to the variable $\mathtt{n}$. Furthermore, we need to update the term b (rename its variables) in order to preserve the correct bindings after the replacement of the variable by b.

For example, translating $(\lambda x\lambda y.zxy)(\lambda x.yx) \rightarrow_\beta \lambda u.z(\lambda x.yx)u$ to de Bruijn notation we get $(\lambda\lambda\mathtt{521})(\lambda\mathtt{31}) \rightarrow_\beta \lambda\mathtt{4}(\lambda\mathtt{41})\mathtt{1}$. But if we simply replace $\mathtt{2}$ in $\lambda\mathtt{521}$ by $\lambda\mathtt{31}$ we get $\lambda\mathtt{5}(\lambda\mathtt{31})\mathtt{1}$, which is not correct. We needed to decrease $\mathtt{5}$ as one λ disappeared and to increment the free variables of $\lambda\mathtt{31}$ as they occur within the scope of one more λ.

For incrementing the free variables we need a family of updating functions:

Definition 5 *The* updating functions $U_k^i : \Lambda \rightarrow \Lambda$ *for $k \geq 0$ and $i \geq 1$ are defined inductively as follows:*

$$U_k^i(ab) = U_k^i(a)\,U_k^i(b)$$

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

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

The intuition behind U_k^i is the following: k tests for free variables and $i - 1$ is the value by which a variable, if free, must be incremented.

Now we define the family of meta-substitution functions:

Definition 6 *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:*

$$(a_1 a_2)\{\!\{\mathtt{j} \leftarrow b\}\!\} \;=\; (a_1\{\!\{\mathtt{j} \leftarrow b\}\!\})\,(a_2\{\!\{\mathtt{j} \leftarrow b\}\!\})$$

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

$$\mathtt{n}\{\!\{\mathtt{j} \leftarrow b\}\!\} \;=\; \begin{cases} \mathtt{n}-\mathtt{1} & \textit{if} \quad n > j \\ U_0^j(b) & \textit{if} \quad n = j \\ \mathtt{n} & \textit{if} \quad n < j\,. \end{cases}$$

Ultimately, the intention is to define $(\lambda a)b \rightarrow_\beta a\{\!\{\mathtt{1} \leftarrow b\}\!\}$ (see definition 7 below). The first two equalities propagate the substitution through applications and abstractions and the last one carries out the substitution of the intended variable (when $n = i$) by the updated term. If the variable is not the intended one it must be decreased by 1 if it is free (case $n > i$) beacuse one λ has disappeared, whereas if it is bound (case $n < i$) it must remain unaltered.

It is easy to check that $(\lambda\mathtt{521})\{\!\{\mathtt{1} \leftarrow (\lambda\mathtt{31})\}\!\} = \lambda\mathtt{4}(\lambda\mathtt{41})\mathtt{1}$. This will mean $(\lambda\lambda\mathtt{521})(\lambda\mathtt{31}) \rightarrow_\beta \lambda\mathtt{4}(\lambda\mathtt{41})\mathtt{1}$.

The following lemmas establish the properties of the meta-substitutions and updating functions. The Meta-substitution and Distribution lemmas are crucial to prove the confluence of λs. The proofs of lemmas 5 - 10 are obtained by induction on a. Furthermore, the proof of lemma 7 requires lemma 6 with $l = 0$; the proof of lemma 8 uses lemmas 5 and 7 both with $k = 0$; finally, lemma 9 with $l = 0$ is needed to prove lemma 10.

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

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

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

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

Proof: Because it is isomorphic to the classical λ-calculus with variable names, the confluence of the latter (cf. [Bar84] thm. 3.2.8) is transportable to the λ-calculus à la de Bruijn.
A proof which does not use the mentioned isomorphism is given in [Río93] (corol. 3.6). □

Finally, the following lemma ensures the good passage of the β-rule through the meta-substitutions and the U_k^i. It is crucial for the proof of the confluence of λs.

Lemma 11 *Let a, b, c, $d \in \Lambda$.*

1. *If $c \to_\beta d$ then $U_k^i(c) \to_\beta U_k^i(d)$.*
2. *If $c \to_\beta d$ then $a\{\!\{\mathtt{n} \leftarrow c\}\!\} \twoheadrightarrow_\beta a\{\!\{\mathtt{n} \leftarrow d\}\!\}$.*
3. *If $a \to_\beta b$ then $a\{\!\{\mathtt{n} \leftarrow c\}\!\} \to_\beta b\{\!\{\mathtt{n} \leftarrow c\}\!\}$.*

Proof:

1. Induction on c. Lemma 10 is needed to treat the case $c = (\lambda a)b \to a\{\!\{\mathtt{1} \leftarrow b\}\!\}$.
2. Induction on a using 1 above.
3. Induction on a. Now lemma 8 is useful to treat the case $a = (\lambda d)e \to d\{\!\{\mathtt{1} \leftarrow e\}\!\}$. □

2.2 The λs-calculus

We begin this subsection by recalling the syntax of the λs-terms. The idea is to handle explicitly the meta-operators defined in definitions 5 and 6. Therefore, the syntax of the λs-calculus is obtained by adding to the syntax of the λ-calculus à la de Bruijn two families of operators :

- $\{\sigma^j\}_{j\geq 1}$ This family is meant to denote 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 number j are to be substituted by the term b.
- $\{\varphi_k^i\}_{k\geq 0\ i\geq 1}$ This family is meant to denote the updating functions necessary when working with de Bruijn numbers to fix the variables of the term to be substituted.

Definition 8 *The* set of terms, *noted* Λs , of the λs-calculus *is given as follows:*

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

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 $\to$ 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 $\varphi_k^i a \to \varphi_k^i b$.*

The λs-calculus should carry out, besides β-reduction, the computations of updating and substitution explicitly. For that reason we include, besides the rule mimicking the β-rule (σ-*generation*), a set of rules which are the equations in definitions 5 and 6 oriented from left to right.

Definition 9 *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* calculus 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.

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*	$\varphi_k^i\, \mathtt{n}$	$\longrightarrow$	$\begin{cases} \mathtt{n+i-1} & \textit{if} \quad n > k \\ \mathtt{n} & \textit{if} \quad n \leq k \end{cases}$

Figure 2: The λs-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 φ-*rules* compute the updating.

We state now the main properties of the λs-calculus.

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 $s(a)$.*

Proof: Let us define recursively a weight function W:

$W(\mathtt{n}) = 1 \qquad W(a\, b) = W(a) + W(b) + 1 \qquad W(\lambda a) = W(a) + 1$

$W(\varphi_k^i a) = 2W(a) \quad W(a\, \sigma^j b) = 2W(a)(W(b) + 1)$

It is easy to show by induction on a that $a \to_s b$ implies $W(a) > W(b)$, hence the s-calculus is strongly normalising.

Since there are no critical pairs, the theorem of Knuth-Bendix (cf. [KB70] or [Hue80]) applies trivially to yield the local confluence of the s-calculus.

Finally, Newman's lemma (cf. lemma 2), provides the confluence of the s-calculus. □

Lemma 12 *The set of s-normal forms is exactly Λ.*

Proof: Check first by induction on a that $a\, \sigma^j b$ and $\varphi_k^i a$ are not normal forms. Then check by induction on a that if a is an s-nf then $a \in \Lambda$. Conclude by observing that every term in Λ is an s-nf. □

Lemma 13 *For all a, $b \in \Lambda s$ we have:*

$$s(a\, b) = s(a)s(b)\,, \quad s(\lambda a) = \lambda(s(a))\,, \quad s(\varphi_k^i a) = U_k^i(s(a))\,, \quad s(a\, \sigma^j b) = s(a)\{\!\{\mathtt{j} \leftarrow s(b)\}\!\}\,.$$

Proof: The first and second equalities are immediate since there are no s-rules whose left-hand side is an application or an abstraction.

Prove the third equality for terms in s-nf, i.e. use an inductive argument on $c \in \Lambda$ to show $s(\varphi_k^i c) = U_k^i(s(c))$. Let now $a \in \Lambda s$, $s(\varphi_k^i a) = s(\varphi_k^i s(a)) = U_k^i(s(s(a))) = U_k^i(s(a))$. Prove the fourth claim similarly using the third one. □

We give now the key result that allows us to use the Interpretation Method in order to get the confluence of the λs-calculus: the good passage of the σ-generation rule to the s-normal forms.

Lemma 14 *Let $a, b \in \Lambda s$, if $a \to_{\sigma-gen} b$ then $s(a) \twoheadrightarrow_\beta s(b)$.*

Proof: Induction on a. Use lemmas 13 and 11. □

Now, the following corollaries are immediate.

Corollary 1 *Let $a, b \in \Lambda s$, if $a \twoheadrightarrow_{\lambda s} b$ then $s(a) \twoheadrightarrow_\beta s(b)$.*

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

This last corollary says that the λs-calculus is correct with respect to the classical λ-calculus, i.e. derivations of pure terms ending with pure terms can also be derived in the classical λ-calculus.

Finally, before proving confluence, we verify that the λs-calculus is powerful enough to simulate β-reduction.

Lemma 15 (Simulation of β-reduction) *Let $a, b \in \Lambda$, if $a \to_\beta b$ then $a \twoheadrightarrow_{\lambda s} b$.*

Proof: Induction on a. □

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

Proof: Use the interpretation method (lemma 3) with $R_1 = \to_s$, $R_2 = \to_{\sigma-gen}$ and $R' = \to_\beta$. Lemmas 14 and 15 and theorem 1 ensure that the hypotheses of the interpretation method hold. □

Finally, for the sake of completeness, we state two other important results concerning the λs-calculus. The proofs are too long to be included here. The proof of the following theorem can be found in [KR95a].

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

In [KR95b] we introduced the simply typed λs-calculus and proved:

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

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*	$l \leq k < l+j$

Figure 3: The new rules of the λs_e-calculus

2.3 The λs_e-calculus

We end this section by introducing the set of open terms and the rules that should be added to λs to obtain the λs_e-calculus.

Definition 10 *The set of* open terms, *noted* Λs_{op} *is given as follows:*

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

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 $(X\sigma^1Y)\sigma^1\mathtt{1}$ 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 since $((\lambda X)\sigma^1\mathtt{1})(Y\sigma^1\mathtt{1}) \twoheadrightarrow (X\sigma^2\mathtt{1})\sigma^1(Y\sigma^1\mathtt{1})$, 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 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. [KR95b]). For instance, the rule corresponding to the Meta-substitution lemma (lemma 8) is the σ-σ-*transition* rule given below. The addition of this rule solves the critical pair in our counterexample, since now we have $(X\sigma^1Y)\sigma^1\mathtt{1} \to (X\sigma^2\mathtt{1})\sigma^1(Y\sigma^1\mathtt{1})$.

Definition 11 *The set of rules* λs_e *is obtained by adding the rules in Figure 3 to the rules of the* λs*-calculus given in 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 calculus of substitutions associated with the λs_e-calculus *is the rewriting system generated by the set of rules* $s_e = \lambda s_e - \{\sigma\textit{-generation}\}$ *and we call it* s_e-calculus.

Notice that when transcribing lemmas 5 - 10 as rewriting rules, instead of keeping the condition $l + j \leq k + 1$ for rule φ-φ-*transition 1*, we restricted it to $l + j \leq k$. The reason for

this alteration is the following: for the extreme case $i = 1$, $j = 1$ and $l + j = k + 1$ we would have:

$$\varphi_k^i(\varphi_l^j(a)) \to \varphi_l^j(\varphi_{k+1-j}^i(a)) \to \varphi_{k+1-j}^i(\varphi_{l+1-i}^j(a) = \varphi_k^i(\varphi_l^j(a))\,,$$

and we would get an infinite loop which would destroy strong normalisation. Furthermore, for $l + j = k + 1$ we have the *φ-φ-transition 2* that allow us to reduce $\varphi_k^i(\varphi_l^j(a))$.

Finally, we recall that only local confluence has been established so far for the λs_e-calculus. The proof was obtained by analysis of critical pairs (cf. [KR95b]):

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 [KR95a] and motivation for explicit substitution rules that belong to the same family can be found in [KN93]. 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. Notice 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_k^i, as for U_k^i, 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. Notice that $\varphi_1^i(\mathtt{1}\,\lambda\mathtt{2}) \twoheadrightarrow_s \mathtt{1}\,\lambda\mathtt{2}$ whereas $\varphi_0^4(\mathtt{1}\,\lambda\mathtt{2}) \twoheadrightarrow_s \mathtt{4}\,\lambda\mathtt{5}$.

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_0^j) 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.

This explains 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. Furthermore, in the rule *φ-σ-tr.* the transition of φ_k^i into φ_{k+1}^i and φ_{k+1-j}^i is explained in the same way.

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 $\varphi_k^i(\varphi_l^j\mathtt{n}) \to_s \varphi_k^i(\mathtt{n+j-1}) \to_s \mathtt{n+j+i-2}$, and we see here the condition at work. Now this double process of updating can be achieved by a single updating: $\varphi_k^{i+j-1}\mathtt{n} \to_s \mathtt{n+j+i-2}$, 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 $\varphi_k^i a$ 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 $\varphi_k^i a$ 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_k^i, whereas *φ-φ-tr.1* postpones the updating φ_l^j. 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. [KR95a]). 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 internalize the knowledge in the meta-level about the meta-operators they represent.

We end this section by comparing λs and λs_e with $\lambda\sigma$, $\lambda\upsilon$ and $\lambda\zeta$. The interpretations[2] T and S (cf. [KR95a]) of λs into $\lambda\sigma$ and $\lambda\upsilon$, respectively, verify: $a \to_s b \Rightarrow T(a) \to^+_\sigma T(b)$ and $a \to_{\lambda s} b \Rightarrow S(a) \twoheadrightarrow^+_{\lambda\upsilon} S(b)$. Moreover, they translate the new rules of λs_e into equalities:

Theorem 7 *If $a \to_{\lambda s_e} b$ then $T(a) =_{\lambda\sigma} T(b)$ and $S(a) =_{\lambda\upsilon} S(b)$.*

Proof: By induction on a, using the classical equalities of $\lambda\sigma$ and $\lambda\upsilon$. □

Notice 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 reductions needed to perform some β-reductions of pure terms in the different calculi. We just give two examples to show that for ceratin 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

[2] T and S were defined in [KR95a] as follows:

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

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

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 $\lambda\sigma$. Notice that $\lambda\zeta$ is less efficient than $\lambda\upsilon$ every time the new mechanism of application is started.

3 The weak normal forms

In [KR95b] we proved the weak normalisation of the s_e-calculus. We are going to give here a different presentation of that proof, since we shall need explicitly the inductive definitions of the weak normal forms.

First of all we state the following remark which shall be used frequently and without explicit mention. A glimpse at the rules in Figure 3 is enough to check it.

Remark 2 *Let $a, b \in \Lambda s_{op}$ then*

1. *$(\varphi_k^i a)\,\sigma^j b$ has a redex at the root iff $j > k$. In this case we say that σ^j* creates a redex *with φ_k^i.*
2. *$\varphi_k^i(\varphi_l^j a)$ has a redex at the root iff $k \geq l$. Now it is φ_k^i that* creates a redex *with φ_l^j.*

Next we recall the description of s_e-normal forms given in [KR95b]. The proof is by analysis of the structure of a and the restrictions for the cases $a = b\,\sigma^j c$ and $a = \varphi_k^i b$ are necessary to avoid redexes at the root.

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 $j \leq k$.*
- *$a = \varphi_k^i b$, 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* [KN95]. Let us just say here that in this notation we write $a\,b = (b\,\delta)a$, $\lambda a = (\lambda)a$, $a\,\sigma^i b = (b\,\sigma^i)a$ and $\varphi_k^i a = (\varphi_k^i)a$. The following nomenclature is used: $(b\,\delta)$, (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*. Notice that every term in Λs_{op} can be written as $\overline{s}\,\mathtt{n}$ or $\overline{s}X$ with a convenient 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:

$(\varphi_k^i)(\varphi_l^j)$ and $k < l$ $\quad$ $(\varphi_k^i)(b\,\sigma^j)$ and $k < j-1$ $\quad$ $(b\,\sigma^i)(c\,\sigma^j)$ and $i < j$ $\quad$ $(b\,\sigma^j)(\varphi_k^i)$ and $j \leq k$.

For example, $(\varphi_3^2)(\varphi_4^1)(\varphi_7^6)(b\sigma^9)(c\sigma^{11})(\varphi_{11}^2)(\varphi_{16}^5)$ and $(b\sigma^1)(c\sigma^3)(d\sigma^4)(\varphi_5^2)(\varphi_6^1)(\varphi_7^4)(a\sigma^{10})$ are normal $\sigma\varphi$-segments.

Finally, in order to explicit the dependence of a normal $\sigma\varphi$-segment on the bodies of the σ-items 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:

$$\overline{s'} = (\varphi_3^2)(\varphi_4^1)(\varphi_7^6)(b\sigma^9)(c\sigma^{11})(\varphi_{11}^2)(b\sigma^{14})(\varphi_{16}^5) \quad \overline{s''} = (b\sigma^1)(c\sigma^3)(d\sigma^4)(\varphi_5^2)(\varphi_6^1)(\varphi_7^4)(a\sigma^{10})$$

have the respective skeletons

$$\overline{\sigma\varphi'} = (\varphi_3^2)(\varphi_4^1)(\varphi_7^6)(\sigma^9)(\sigma^{11})(\varphi_{11}^2)(\sigma^{14})(\varphi_{16}^5) \quad \overline{\sigma\varphi''} = (\sigma^1)(\sigma^3)(\sigma^4)(\varphi_5^2)(\varphi_6^1)(\varphi_7^4)(\sigma^{10})\,,$$

and using the above mentioned convention, they should be 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 [KR95b]. 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).*

We define now our strategy to calculate normal forms. We do it in three steps:

1. We define a function s'_e to evaluate a normal form of $\varphi_k^i d$ 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 s^*_e, a function which evaluates an s_e-normal form of every $a \in \Lambda s_{op}$.

Notice that we use the notation $s^*_e(a)$ instead of $s_e(a)$ to bear in mind that $s^*_e(a)$ is *one* normal form and, in order to be coherent with remark 1, the notation $s_e(a)$ could be used only after having established the confluence of the s_e-calculus which ensures unicity of nf's.

Definition 12 *Let $d \in NF$, we define $s'_e(\varphi_k^i d)$ by induction on d as follows:*

$$\begin{aligned}
s'_e(\varphi_k^i X) &= \varphi_k^i X \\
s'_e(\varphi_k^i \mathtt{n}) &= \begin{cases} \mathtt{n+i-1} & \textit{if} \quad n > k \\ \mathtt{n} & \textit{if} \quad n \le k \end{cases} \\
s'_e(\varphi_k^i (a\,b)) &= s'_e(\varphi_k^i a)\, s'_e(\varphi_k^i b) \\
s'_e(\varphi_k^i (\lambda a)) &= \lambda s'_e(\varphi_{k+1}^i a)
\end{aligned}$$

$$s'_e(\varphi^i_k(\varphi^j_l a)) = \begin{cases} \varphi^i_k(\varphi^j_l a) & \textit{if} \quad k < l \\ \varphi^{j+i-1}_l a & \textit{if} \quad l \leq k < l+j \\ \varphi^j_l(s'_e(\varphi^i_{k+1-j} a)) & \textit{if} \quad l+j \leq k \end{cases}$$

$$s'_e(\varphi^i_k(a\,\sigma^j b)) = \begin{cases} \varphi^i_k(a\,\sigma^j b) & \textit{if} \quad j > k+1 \\ s'_e(\varphi^i_{k+1} a)\,\sigma^j s'_e(\varphi^i_{k+1-j} b) & \textit{if} \quad j \leq k+1 \end{cases}$$

Remark the analogy of these equalities with the φ-rules (the rules whose name begin with the symbol φ).

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

$$s''_e(X\,\sigma^j b) = X\,\sigma^j b$$

$$s''_e(\mathtt{n}\,\sigma^j b) = \begin{cases} \mathtt{n-1} & \textit{if} \quad n > j \\ s'_e(\varphi^j_0 b) & \textit{if} \quad n = j \\ \mathtt{n} & \textit{if} \quad n < j \end{cases}$$

$$s''_e((a\,c)\,\sigma^j b) = s''_e(a\,\sigma^j b)\,s''_e(c\,\sigma^j b)$$

$$s''_e((\lambda a)\,\sigma^j b) = \lambda s''_e(a\,\sigma^{j+1} b)$$

$$s''_e((\varphi^i_k a)\,\sigma^j b) = \begin{cases} (\varphi^i_k a)\,\sigma^j b & \textit{if} \quad j \leq k \\ \varphi^{i-1}_k a & \textit{if} \quad k < j < k+i \\ s'_e(\varphi^i_{k+1} a)\,\sigma^{k+1} s'_e(\varphi^i_0 b) & \textit{if} \quad j = k+i \\ \varphi^i_k(s''_e(a\,\sigma^{j+1-i} b)) & \textit{if} \quad j > k+i \end{cases}$$

$$s''_e((a\,\sigma^i c)\,\sigma^j b) = \begin{cases} (a\,\sigma^i c)\,\sigma^j b & \textit{if} \quad i > j \\ s''_e(a\,\sigma^{j+1} b)\,\sigma^i s''_e(c\,\sigma^{j+1-i} b) & \textit{if} \quad i \leq j \end{cases}$$

Remark again the analogy of these rules with the σ-rules (the rules whose name begin with σ). Only one equality seems to be out of 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 due to the fact that only when $j = k+i$ an application of σ-φ *tr.2* creates a new φ-σ *tr.*-redex:

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

Now we are ready to define our strategy:

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

$$s^*_e(X) = X \qquad s^*_e(a\,b) = s^*_e(a)\,s^*_e(b) \qquad s^*_e(\varphi^i_k a) = s'_e(\varphi^i_k s^*_e(a))$$
$$s^*_e(\mathtt{n}) = \mathtt{n} \qquad s^*_e(\lambda a) = \lambda s^*_e(a) \qquad s^*_e(a\,\sigma^j b) = s''_e(s^*_e(a)\,\sigma^j s^*_e(b))$$

We must prove now that in definitions 12, 13 and 14 we have really defined normal forms. We proceed by induction, but we need a powerful inductive hypothesis. For this reason we need the following definition.

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 $N(X) = 0$.*

The idea in defining such numbers is that those are the indices that really matter to decide the existence of redexes (see the definition of normal $\sigma\varphi$-segment above). The following remark precises this intuitive idea.

Remark 3 *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 every $c \in NF$.*

Proof: The first and second items are proved analogously. Let us prove the first one.

Since $b \in NF$ we only must check that φ^i_k does not create a redex with the principal operator (the one at the root) of b.

Now, since $\varphi^i_k a \in NF$, a is a variable or a $\sigma\varphi$-normal form, hence, by hypothesis, b is a variable or a $\sigma\varphi$-normal form of the same sort and number. Therefore, φ^j_k creates a redex with the principal operator of b iff φ^i_k creates a redex with the principal operator of a. Since $\varphi^i_k a \in NF$, we conclude that there is no redex at the root in $\varphi^i_k b$, hence $\varphi^i_k b \in NF$.

To check the third item, let us suppose that σ^{k+1} creates a redex with the principal operator of b, hence this operator must be σ^h with $h \leq k+1$ (see rule *σ-σ-transition*) or φ^j_l with $k+1 > l$ (see remark 2.1). But the principal operator of a has same sort and number as the principal operator of b and in both cases the hypothesis $\varphi^i_k a \in NF$ is contradicted. □

We can begin now our proof of weak normalisation.

Lemma 16 *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 have furthermore $N(a) = N(s'_e(\varphi^i_k a))$.

Proof: The cases $a = X$ and $a = \mathtt{n}$ are evident. The cases $a = b\,c$ and $a = \lambda b$ are immediate since there are no rules whose left members are applications or abstractions, and therefore if d, $e \in NF$ then $d\,e \in NF$ and $\lambda d \in NF$.

If $a = \varphi^j_l b$ and $k < l$ the lemma is evident. When $l \leq k < l + j$, apply remark 3.1 to conclude $\varphi^{j+i-1}_l a \in NF$ from $\varphi^j_l a \in NF$. Let us check now the interesting case $l + j \leq k$. Now, $s'_e(\varphi^i_k a) = s'_e(\varphi^i_k(\varphi^j_l b)) \overset{D12}{=} \varphi^j_l s'_e(\varphi^i_{k+1-j} b)$. 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 $S(b) = S(s'_e(\varphi^i_{k+1-j} b))$. Remark that, since $a = \varphi^j_l b \in NF$, b is neither an application nor an abstraction, furthermore b is not a variable (otherwise $s'_e(\varphi^i_{k+1-j} b) = \varphi^i_{k+1-j} b$). Therefore b is a $\sigma\varphi$-normal form, and we also have $N(b) = N(s'_e(\varphi^i_k b))$. We conclude by remark 3.1 that $\varphi^j_l s'_e(\varphi^i_k b) \in NF$.

Finally, if $a = b\,\sigma^j c$ and $j > k + 1$ the lemma is evident. If $j \leq k + 1$, we reason as in the last subcase of the previous case, using now remark 3.2. □

Lemma 17 *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. The only differences are:

The previous lemma is needed to settle the case $a = \mathtt{j}$.

When $a = \varphi^i_k c$ and $k + i = j$, use lemma 16 and remark 3.3.

When $a = \varphi^i_k c$ and $k + i < j$, then $s''_e((\varphi^i_k c)\,\sigma^j b) \overset{D13}{=} \varphi^i_k s''_e(c\,\sigma^{j+1-i} b)$. If $s''_e(c\,\sigma^{j+1-i} b) = c\,\sigma^{j+1-i} b$, then $s''_e((\varphi^i_k c)\,\sigma^j b) = \varphi^i_k(c\,\sigma^{j+1-i} b)$, which is a normal form because $j+1-i > k+1$. If $s''_e(c\,\sigma^{j+1-i} b) \neq c\,\sigma^{j+1-i} b$, then in order to apply the IH, we consider:

- If $c \neq \varphi^h_m d$ with $h + m = j + 1 - i$, then the sort and number of $s''_e(c\,\sigma^{j+1-i} b)$ are the same as those of c, and, since $\varphi^i_k c \in NF$ we conclude $\varphi^i_k s''_e(c\,\sigma^{j+1-i} b) \in NF$.
- If $c = \varphi^h_m d$ with $h + m = j + 1 - i$, then the principal operator of $s''_e(c\,\sigma^{j+1-i} b)$ is σ^{m+1}. But $a = \varphi^i_k c = \varphi^i_k \varphi^h_m d \in NF$, hence $k < m$. Therefore φ^i_k does not create a redex with σ^{m+1}, and again $\varphi^i_k s''_e(c\,\sigma^{j+1-i} b) \in NF$.

Finally, when $a = c\,\sigma^h d$ and $h \leq j$ reason as in the case we have just considered. □

Since, as mentioned above, no left member of any s_e-rule is an application or an abstraction, the following theorem follows immediately from lemmas 16 and 17.

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.*

We have therefore a strategy to find s_e-normal forms. By looking carefully at the definitions of s^*_e, s'_e and s''_e the strategy appears as an *innermost* one, i.e. before reducing a redex all its subterms must have been already normalised.

Notice that for the out-of-the-pattern case we pointed out after definition 13 the strategy is also 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, by remark 3.3, $a\,\sigma^{k+1}\, b \in NF$. Therefore, the only redex in $\varphi^i_k(a\,\sigma^{k+1}\, b)$ is the φ-σ-*transition*-redex at the root.

The remainder of this section is the prize we have to pay because we don't know if s_e is strongly normalising. In fact, if we knew that s_e is SN, since we proved the local confluence of s_e (cf. theorem 6), we could apply Newman's lemma to obtain the confluence of s_e. In that case, all the lemmas which follow are trivial. The aim of these lemmas is proposition 1 which will translate into one of the conditions of the GIM lemma.

Lemma 18 *The following hold:*

1. $s^*_e(s^*_e(a)) = s^*_e(a)$
2. $s^*_e(\varphi^i_k a) = s^*_e(\varphi^i_k s^*_e(a))$
3. $s^*_e(a\,\sigma^j b) = s^*_e(s^*_e(a)\,\sigma^j s^*_e(b))$

Proof: The first item is immediate: since $s_e^*(s_e^*(a))$ is an s_e-nf of $s_e^*(a)$, $s_e^*(a) \twoheadrightarrow_{s_e} s_e^*(s_e^*(a))$. But, since $s_e^*(a)$ is an s_e-nf, this derivation must have null length. Therefore, $s_e^*(s_e^*(a)) = s_e^*(a)$.

The second item follows from the first, since by definition of s_e^* we have $s_e^*(\varphi_k^i a) = s'_e(\varphi_k^i s_e^*(a))$ and $s_e^*(\varphi_k^i s_e^*(a)) = s'_e(\varphi_k^i s_e^*(s_e^*(a)))$. The third item is proved analogously. □

Lemma 19 *For every rule $L \to R$ of the s_e-calculus, $s_e^*(L) = s_e^*(R)$.*

Proof: We divide the proof in two parts:

I. We prove the lemma with the extra hypothesis that all a, b, c involved in the rules are s_e-normal forms.

II. We use part I to prove the lemma for arbitrary a, b, c.

Part I is almost immediate for the six s-rules: only definitions 12, 13 and 14 are needed. We prove the lemma, for instance, for σ-λ-*tr* when a, $b \in NF$.

$$s_e^*((\lambda a)\,\sigma^j b) \overset{D.14}{=} s''_e(s_e^*(\lambda a)\,\sigma^j s_e^*(b)) \overset{D.14}{=} s''_e((\lambda s_e^*(a))\,\sigma^j s_e^*(b)) \overset{D.13}{=}$$

$$\lambda s''_e(s_e^*(a)\,\sigma^{j+1} s_e^*(b)) \overset{D.14}{=} \lambda s_e^*(a\sigma^{j+1} b) \overset{D.14}{=} s_e^*(\lambda(a\sigma^{j+1} b))$$

Part I, for the other six rules, involves an enormous amount of elementary calculations. Moreover, the rules must be chosen in a good order, since the proof of the lemma for one of them may require the validity of the lemma for another one. All the proofs are obtained by structural induction.

φ-φ-*tr.1*: We prove $s_e^*(\varphi_k^i(\varphi_l^j a)) = s_e^*(\varphi_l^j(\varphi_{k+1-j}^i a))$ for $a \in NF$ and $l+j \le k$ by induction on a. By definition 14, our thesis is equivalent to $s'_e(\varphi_k^i(s'_e(\varphi_l^j a))) = s'_e(\varphi_l^j(s'_e(\varphi_{k+1-j}^i a)))$.

The cases $a = X$ and $a = \mathtt{n}$ can be checked almost immediately using just definition 12. The cases $a = d\,e$ and $a = \lambda d$ require the inductive hypothesis (IH). Let us check, for instance the case $a = \lambda d$:

$$s'_e(\varphi_k^i(s'_e(\varphi_l^j(\lambda d)))) \overset{D.12}{=} s'_e(\varphi_k^i \lambda(s'_e(\varphi_{l+1}^j d))) \overset{D.12}{=} \lambda s'_e(\varphi_{k+1}^i(s'_e(\varphi_{l+1}^j d))) \overset{IH}{=}$$

$$\lambda s'_e(\varphi_{l+1}^j(s'_e(\varphi_{k+2-j}^i d))) \overset{D.12}{=} s'_e(\varphi_l^j \lambda(s'_e(\varphi_{k+2-j}^i d))) \overset{D.12}{=} s'_e(\varphi_l^j(s'_e(\varphi_{k+1-j}^i(\lambda d))))$$

The cases $a = \varphi_m^h d$ and $a = d\sigma^h e$ are the ones which involve the long calculations. To give an idea of these calculations we check $a = d\sigma^h e$. We introduce capitals A, B, $C\ldots$ as local definitions in order to save space and gain in clarity. Let $A = s'_e(\varphi_k^i(s'_e(\varphi_l^j(d\,\sigma^h e))))$ and $B = s'_e(\varphi_l^j(s'_e(\varphi_{k+1-j}^i(d\,\sigma^h e))))$. We want to prove $A = B$.

$$A \overset{D.12}{=} \begin{cases} s'_e(\varphi_k^i(\varphi_l^j(d\,\sigma^h e))) \overset{D.12}{=} \varphi_l^j s'_e(\varphi_{k+1-j}^j(d\,\sigma^h e)) = C & \text{if } h > l+1 \\ s'_e(\varphi_k^i(s'_e(\varphi_{l+1}^j d)\sigma^h s'_e(\varphi_{l+1-h}^j e))) = D & \text{if } h \le l+1 \end{cases}$$

$$C \overset{D.12}{=} \begin{cases} \varphi_l^j(\varphi_{k+1-j}^i(d\,\sigma^h e)) = E & \text{if } h > k+2-j \\ \varphi_l^j(s'_e(\varphi_{k+2-j}^i d)\sigma^h s'_e(\varphi_{k+2-j-h}^i e)) = F & \text{if } h \le k+2-j \end{cases}$$

$$D \overset{D.12}{=} \begin{cases} \varphi^i_k(s'_e(\varphi^j_{l+1}d)\sigma^h s'_e(\varphi^j_{l+1-h}e)) & \text{if } h > k+1 \\ s'_e(\varphi^i_{k+1}(s'_e(\varphi^j_{l+1}d)))\sigma^h s'_e(\varphi^i_{k+1-h}(s'_e(\varphi^j_{l+1-h}e))) = G & \text{if } h \leq k+1 \end{cases}$$

$$B \overset{D.12}{=} \begin{cases} s'_e(\varphi^j_l(\varphi^i_{k+1-j}(d\,\sigma^h e))) = H & \text{if } h > k+2-j \\ s'_e(\varphi^j_l(s'_e(\varphi^i_{k+2-j}d)\sigma^h s'_e(\varphi^i_{k+2-j-h}e))) = I & \text{if } h \leq k+2-j \end{cases}$$

$$I \overset{D.12}{=} \begin{cases} \varphi^j_l(s'_e(\varphi^i_{k+2-j}d)\sigma^h s'_e(\varphi^i_{k+2-j-h}e)) = J & \text{if } h > l+1 \\ s'_e(\varphi^j_{l+1}s'_e(\varphi^i_{k+2-j}d))\sigma^h s'_e(\varphi^j_{l+1-h}s'_e(\varphi^i_{k+2-j-h}e)) = K & \text{if } h \leq l+1 \end{cases}$$

Let us study our results recalling $l + j \leq k$:

- $\underline{h > l+1 \text{ and } h > k+2-j}$: Since $l < k+1-j$, we have $H = E$, hence $A = B$.
- $\underline{h > l+1 \text{ and } h \leq k+2-j}$: $F = J$, hence $A = B$.
- $\underline{h \leq l+1 \text{ and } h > k+2-j}$: Impossible since $l+1 \leq k-j+1$.
- $\underline{h \leq l+1 \text{ and } h \leq k+2-j}$: Since $j \geq 1$, $k+2-j \leq k+1$, hence $h \leq k+1$ and by IH we have $G = K$. Therefore, $A = B$.

φ-φ-tr.2: $s^*_e(\varphi^i_k(\varphi^j_l a)) = s^*_e(\varphi^{j+i-1}_l a)$ for $a \in NF$ and $l \leq k < l+j$. By induction on a.

σ-φ-tr.1: $s^*_e((\varphi^i_k a)\,\sigma^j b) = s^*_e(\varphi^{i-1}_k a)$ for a, $b \in NF$ and $k < j < k+i$. By induction on a.

σ-φ-tr.2: $s^*_e((\varphi^i_k a)\,\sigma^j b) = s^*_e(\varphi^i_k(a\sigma^{j-i+1}b))$ for a, $b \in NF$, $k+i \leq j$. By induction on a.

When considering the cases $a = \mathtt{j-i+1}$ and $a = \varphi^h_l d$ with $l < k < l+h$ (namely in the subcase $l+h = j-i+1$), use the fact that the lemma has already been established for the rule *φ-φ-tr.2*.

φ-σ-tr.: $s^*_e((\varphi^i_k a)\,\sigma^j b) = s^*_e((\varphi^i_{k+1}a)\,\sigma^j(\varphi^i_{k+1-j}b))$ for a, $b \in NF$ and $j \leq k+1$. By induction on a.

Cases $a = \mathtt{k+1}$ and $a = \varphi^h_l d$ with $j = l+h = k+1$, require the lemma for the rule *φ-φ-tr.2*, whereas the cases $a = \mathtt{j}$ with $j \leq k$ and $a = \varphi^h_l d$ with $j = l+h < k+1$, require the lemma for the rule *φ-φ-tr.1*.

σ-σ-tr.: $s^*_e((a\,\sigma^i b)\,\sigma^j c) = s^*_e((a\,\sigma^{j+1}c)\,\sigma^i(b\,\sigma^{j-i+1}c))$ for a, b, $c \in NF$ and $i \leq j$. By induction on a.

Now the lemma is required for the rules *σ-φ-tr.1* and *σ-φ-tr.2*. We encourage the reader to find out in what cases.

We state here some general remarks concerning the proof of part I for the rules which are not s-rules.

- All the proofs are by induction on a for a convenient a, as stated above.
- The case $a = X$ is always easy and proved using just the definitions 12 and 13.
- The cases $a = d\,e$ and $a = \lambda d$ are always treated using definitions 12 and 13 and the IH.
- The cases $a = \mathtt{n}$, $a = \varphi^h_m d$ and $a = d\,\sigma^h e$ always involve case analyses, in particular the last two cases require rather long ones (the only example we gave for *φ-φ-tr. 1* is one of the simplest). The proofs may use, besides definitions 12 and 13 and the IH the fact that the lemma is true for another rule.

Part II is easy. We just check only two rules. The proof for the other rules is analogous.

σ-λ-tr: In part one we have proved $s_e^*((\lambda a)\,\sigma^j b) = s_e^*(\lambda(a\,\sigma^{j+1}b))$ for every a, $b \in NF$. Let now a, $b \in \Lambda s_{op}$.

$$s_e^*((\lambda a)\,\sigma^j b) \overset{L.18.3}{=} s_e^*(s_e^*(\lambda a)\,\sigma^j s_e^*(b)) \overset{D.14}{=} s_e^*((\lambda s_e^*(a))\,\sigma^j s_e^*(b)) \overset{\text{Part I}}{=}$$

$$s_e^*(\lambda(s_e^*(a)\,\sigma^{j+1} s_e^*(b))) \overset{D.14}{=} \lambda(s_e^*(s_e^*(a)\,\sigma^{j+1} s_e^*(b))) \overset{L.18.3}{=} \lambda(s_e^*(a\,\sigma^{j+1}b)) \overset{D.14}{=} s_e^*(\lambda(a\,\sigma^{j+1}b))$$

σ-φ-tr 2: In part one we have proved $s_e^*((\varphi_k^i a)\,\sigma^j b) = s_e^*(\varphi_k^i(a\sigma^{j-i+1}b))$ for every a, $b \in NF$ and $k + i \leq j$. Let now a, $b \in \Lambda s_{op}$.

$$s_e^*((\varphi_k^i a)\,\sigma^j b) \overset{L.18.3}{=} s_e^*(s_e^*(\varphi_k^i a)\,\sigma^j s_e^*(b)) \overset{L.18.2}{=} s_e^*(s_e^*(\varphi_k^i s_e^*(a))\,\sigma^j s_e^*(b)) \overset{L.18.1}{=}$$

$$s_e^*(s_e^*(\varphi_k^i s_e^*(a))\,\sigma^j s_e^*(s_e^*(b))) \overset{L.18.3}{=} s_e^*((\varphi_k^i s_e^*(a))\,\sigma^j s_e^*(b)) \overset{\text{Part I}}{=} s_e^*(\varphi_k^i(s_e^*(a)\,\sigma^{j-i+1} s_e^*(b)))$$

$$\overset{L.18.2}{=} s_e^*(\varphi_k^i s_e^*(s_e^*(a)\,\sigma^{j-i+1} s_e^*(b))) \overset{L.18.3}{=} s_e^*(\varphi_k^i s_e^*(a\,\sigma^{j-i+1}b)) \overset{L.18.2}{=} s_e^*(\varphi_k^i(a\,\sigma^{j-i+1}b)) \qquad \square$$

We show now that our s_e-normal forms are preserved by s_e-reduction.

Proposition 1 *Let a, $b \in \Lambda s_{op}$, if $a \to_{s_e} b$ then $s_e^*(a) = s_e^*(b)$.*

Proof: By induction on a. If the reduction is at the root the proposition coincides with lemma 19. Otherwise use the IH and either definition 14 (for the cases $a = d\,e$ and $a = \lambda d$) or lemma 18 (for the cases $a = d\,\sigma^j e$ and $a = \varphi_k^i d$). $\square$

As a corollary of this proposition we obtain the confluence of the s_e-calculus:

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 $c \twoheadrightarrow_{s_e} s_e^*(c)$. $\square$

Therefore, 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 $a \in \Lambda s_{op}$, $s_e(\varphi_k^i b) = s_e'(\varphi_k^i b)$ 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

We recall that our aim is to apply the generalized interpretation method (lemma 4) to obtain the confluence of the λs_e-calculus. Our interpretation function will be s_e. Coming back to the notation of lemma 4, we intend to apply the GIM with: $f = s_e$, $R = \lambda s_e$, $R_1 = \to_{s_e}$ and $R_2 = \to_{\sigma-gen}$.

In the previous section we proved (proposition 1) that condition 2 of lemma 4 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 are also satisfied. We postpone the confluence of R' until the next section.

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

We take $\to_{\beta'}$ as R'. Condition 1 in lemma 4 is immediate:

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

Proof: Because $a \to_{\sigma-gen} d \twoheadrightarrow_{s_e} s_e(d) = b$. □

The remainder of this section is a series of lemmas aiming to proposition 3 which states that condition 3 of lemma 4 holds. If the reader wishes, she/he can read now the proof of proposition 3 and backtrack the necessity of the lemmas.

We begin by showing that β' is compatible with applications and abstractions.

Lemma 20 *Let $a, b, c \in NF$.*

1. *If $a \to_{\beta'} b$, then $a\,c \to_{\beta'} b\,c$.*
2. *If $a \to_{\beta'} b$, then $c\,a \to_{\beta'} c\,b$.*
3. *If $a \to_{\beta'} b$, then $\lambda a \to_{\beta'} \lambda b$.*
4. *Si $a \twoheadrightarrow_{\beta'} b$ et $c \twoheadrightarrow_{\beta'} d$, alors $ac \twoheadrightarrow_{\beta'} bd$.*
5. *Si $a \twoheadrightarrow_{\beta'} b$, alors $\lambda a \twoheadrightarrow_{\beta'} \lambda b$.*

Proof: We check the first item. If $a \to_{\beta'} b$, there exists $d \in \Lambda s_{op}$ such that $a \to_{\sigma-gen} d$ and $b = s_e(d)$, hence $a\,c \to_{\sigma-gen} d\,c$ and, since $s_e(d\,c) = s_e(d)s_e(c) = bc$, we conclude $a\,c \to_{\beta'} b\,c$.

The second and third ones are shown analogously.

The fourth one is proved by induction on the length of the derivation $c \twoheadrightarrow_{\beta'} d$. If it is null, we must show $ac \twoheadrightarrow_{\beta'} bc$. This is done by induction on the length of the derivation $a \twoheadrightarrow_{\beta'} b$ using the first item. For the inductive step of the main induction use the second item.

The fifth one is shown by induction on the length of the derivation using the third item.
□

Lemma 21 *If a is a $\sigma\varphi$-normal form and $a \to_{\sigma-gen} d$ then $S(a) = S(s_e(d))$ and $N(a) = N(s_e(d))$ (cf. def. 15).*

Proof: By induction on a. Since a is a $\sigma\varphi$-normal form we have to consider only two cases:

$a = \varphi_k^i b$: Hence $d = \varphi_k^i c$ with $b \to_{\sigma-gen} c$. By IH we have $S(b) = S(s_e(c))$ and $N(b) = N(s_e(c))$ and, since $\varphi_k^i b$ is a normal form, we conclude, by remark 3.1, that $\varphi_k^i s_e(c)$ is also a normal form. Therefore, $\varphi_k^i s_e(c) = s_e(\varphi_k^i s_e(c)) = s_e(\varphi_k^i c)$ and we conclude $S(a) = \varphi = S(s_e(d))$ and $N(a) = k = N(s_e(d))$.

$a = b\,\sigma^j c$: Two cases according to the position of the redex:

- If $b \to_{\sigma-gen} b'$ then the proof is analogous to the previous case. Use remark 3.2.
- If $c \to_{\sigma-gen} c'$ then $d = b\,\sigma^j c'$. Now, since $b\,\sigma^j c$ is a nf, $b\,\sigma^j s_e(c')$ is also a nf (a glimpse at the rules is enough to verify that σ operators can only create redexes with the left operand). Therefore, $b\,\sigma^j s_e(c') = s_e(b\,\sigma^j s_e(c')) = s_e(b\,\sigma^j c')$ and we conclude $S(a) = \sigma = S(s_e(d))$ and $N(a) = j = N(s_e(d))$. □

Lemma 22 *For a, b, c, $e \in NF$ the following hold:*

1. *If $a \to_{\beta'} b$ and $\varphi_k^i a \in NF$ then $\varphi_k^i b \in NF$ and $\varphi_k^i a \to_{\beta'} \varphi_k^i b$.*
2. *If $a \to_{\beta'} b$ and $a\,\sigma^j c \in NF$ then $b\,\sigma^j c \in NF$ and $a\,\sigma^j c \to_{\beta'} b\,\sigma^j c$.*
3. *If $a \to_{\beta'} b$ and $c\,\sigma^j a \in NF$ then $c\,\sigma^j b \in NF$ and $c\,\sigma^j a \to_{\beta'} c\,\sigma^j b$.*
4. *If $a \twoheadrightarrow_{\beta'} b$ and $\varphi_k^i a \in NF$ then $\varphi_k^i b \in NF$ and $\varphi_k^i a \twoheadrightarrow_{\beta'} \varphi_k^i b$.*
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: Let $d \in \Lambda s_{op}$ such that $a \to_{\sigma-gen} d$ and $b = s_e(d)$.

1. Since $\varphi_k^i a \in NF$, a is neither an abstraction nor an application. Furthermore, since a has a redex, a is not a variable, hence a is a $\sigma\varphi$-normal form. By lemma 21, we have $S(a) = S(s_e(d))$ and $N(a) = N(s_e(d))$. By remark 3.1, $\varphi_k^i s_e(d) \in NF$, hence $\varphi_k^i b \in NF$. Now, $\varphi_k^i a \to_{\sigma-gen} \varphi_k^i d$ and $s_e(\varphi_k^i d) = s_e(\varphi_k^i s_e(d)) = s_e(\varphi_k^i b) = \varphi_k^i b$, hence $\varphi_k^i a \to_{\beta'} \varphi_k^i b$.
2. Analogous to the previous item. Use now remark 3.2.
3. Since $c\,\sigma^j a \in NF$, $c\,\sigma^j s_e(d) \in NF$ (as we remarked above, a glimpse at the rules is enough to verify that σ operators can only create redexes with the left operand). Therefore, $s_e(c\,\sigma^j d) = c\,\sigma^j s_e(d)$ and since $c\,\sigma^j a \to_{\sigma-gen} c\,\sigma^j d$, we conclude $c\,\sigma^j a \to_{\beta'} c\,\sigma^j b$.
4. By induction on the length of the derivation using item 1.
5. By induction on the length of the derivation $c \twoheadrightarrow_{\beta'} e$.

   When the length is null, check $a\,\sigma^j c \twoheadrightarrow_{\beta'} b\,\sigma^j c$ by induction on the length of the derivation $a \twoheadrightarrow_{\beta'} b$ using item 2.

   For the inductive step use item 3. □

Lemma 23 *Let $a \in NF$ and $d \in \Lambda s_{op}$, if $a \to_{\sigma-gen} d$ then $s_e(\varphi_k^i a) \to_{\beta'} s_e(\varphi_k^i d)$.*

Proof: By induction on a. If $a = X$ or $a = \mathtt{n}$ there are no *σ-generation* redexes and therefore the lemma is trivial.

$a = b\,c$ **:** Three cases according to the position of the redex:

- If $b \to_{\sigma-gen} b'$ then $d = b'c$.

$$s_e(\varphi_k^i(b\,c)) = s_e(\varphi_k^i b) s_e(\varphi_k^i c) \overset{IH\ \&\ L\,20.1}{\to_{\beta'}} s_e(\varphi_k^i b') s_e(\varphi_k^i c) = s_e(\varphi_k^i(b'c))$$

- If $c \to_{\sigma-gen} c'$, it is analogous to the previous case. Instead of lemma 20.1, use now lemma 20.2.
- If $b = \lambda b'$ and the redex is at the root, then $d = b'\,\sigma^1 c$.

$$s_e(\varphi_k^i((\lambda b')c)) = (\lambda s_e(\varphi_{k+1}^i b')) s_e(\varphi_k^i c) \to_{\beta'} s_e(s_e(\varphi_{k+1}^i b')\,\sigma^1 s_e(\varphi_k^i c)) =$$

$$s_e((\varphi_{k+1}^i b')\,\sigma^1(\varphi_k^i c)) = s_e(\varphi_k^i(b'\,\sigma^1 c))$$

We remark once and for all that the last equality holds because

$$\varphi^i_k(b'\,\sigma^1 c) \to_{\varphi\text{-}\sigma\text{-}tr} (\varphi^i_{k+1}b')\,\sigma^1(\varphi^i_k c)$$

and proposition 1 (or lemma 19) applies.

$a = \lambda b$ **:** The redex must be internal, and it is anologous to the first subcase above.

$a = \varphi^j_l b$**:** Necessarily $b \to_{\sigma-gen} b'$ and $d = \varphi^j_l b'$. Remember that $s_e(\varphi^i_k a) = s^*_e(\varphi^i_k a)$ and, since $a \in NF$, $s^*_e(\varphi^i_k a) \stackrel{D14}{=} s'_e(\varphi^i_k(\varphi^j_l b))$. Three subcases must be considered:

- If $k < l$ then $s'_e(\varphi^i_k(\varphi^j_l b)) \stackrel{D12}{=} \varphi^i_k(\varphi^j_l b) \to_{\beta'} s_e(\varphi^i_k(\varphi^j_l b'))$.
- If $l \le k < l + j$ then $s'_e(\varphi^i_k(\varphi^j_l b)) \stackrel{D12}{=} \varphi^{i+j-1}_l b \to_{\beta'} s_e(\varphi^{i+j-1}_l b') = s_e(\varphi^i_k(\varphi^j_l b'))$.
- If $l+j \le k$ then $s'_e(\varphi^i_k(\varphi^j_l b)) \stackrel{D12}{=} \varphi^j_l s'_e(\varphi^i_{k+1-j} b) \stackrel{D14}{=} \varphi^j_l s^*_e(\varphi^i_{k+1-j} b) = \varphi^j_l s_e(\varphi^i_{k+1-j} b)$. But by IH we know $s_e(\varphi^i_{k+1-j} b) \to_{\beta'} s_e(\varphi^i_{k+1-j} b')$ and therefore, by lemma 22.1:

  $$\varphi^j_l s_e(\varphi^i_{k+1-j} b) \to_{\beta'} \varphi^j_l s_e(\varphi^i_{k+1-j} b')$$

  Furthermore, lemma 22.1 ensures $\varphi^j_l s_e(\varphi^i_{k+1-j} b') \in NF$. Hence

  $$\varphi^j_l s_e(\varphi^i_{k+1-j} b') = s_e(\varphi^j_l(\varphi^i_{k+1-j} b')) = s_e(\varphi^i_k(\varphi^j_l b'))$$

$a = b\,\sigma^j c$**:** There are two cases according to the position of the redex, namely, $b \to_{\sigma-gen} b'$ and $c \to_{\sigma-gen} c'$. Since the treatment of the second is analogous to that of the first one, we just consider $b \to_{\sigma-gen} b'$ and therefore $d = b'\,\sigma^j c$.

We consider two subcases:

- If $j > k + 1$ then $s_e(\varphi^i_k(b\,\sigma^j c)) = \varphi^i_k(b\,\sigma^j c) \to_{\beta'} s_e(\varphi^i_k(b'\,\sigma^j c))$.
- If $j \le k + 1$ then, recalling that $s_e(\varphi^i_k(b\,\sigma^j c)) = s^*_e(\varphi^i_k(b\,\sigma^j c))$, we have:

  $$s_e(\varphi^i_k(b\,\sigma^j c)) \stackrel{D14\,\&\,D12}{=} s'_e(\varphi^i_{k+1} b)\,\sigma^j s'_e(\varphi^i_{k+1-j} c) \stackrel{D14}{=} s_e(\varphi^i_{k+1} b)\,\sigma^j s_e(\varphi^i_{k+1-j} c).$$

  But by IH we know $s_e(\varphi^i_{k+1} b) \to_{\beta'} s_e(\varphi^i_{k+1} b')$ and therefore, by lemma 22.2:

  $$s_e(\varphi^i_{k+1} b)\,\sigma^j s_e(\varphi^i_{k+1-j} c) \to_{\beta'} s_e(\varphi^i_{k+1} b')\,\sigma^j s_e(\varphi^i_{k+1-j} c)$$

  Furthermore, lemma 22.2 ensures $s_e(\varphi^i_{k+1} b')\,\sigma^j s_e(\varphi^i_{k+1-j} c) \in NF$. Hence

  $$s_e(\varphi^i_{k+1} b')\,\sigma^j s_e(\varphi^i_{k+1-j} c) = s_e(s_e(\varphi^i_{k+1} b')\,\sigma^j s_e(\varphi^i_{k+1-j} c)) =$$
  $$s_e((\varphi^i_{k+1} b')\,\sigma^j(\varphi^i_{k+1-j} c)) = s_e(\varphi^i_k(b'\,\sigma^j c))$$

  We remark that when considering $c \to_{\sigma-gen} c'$ lemma 22.3, instead of lemma 22.2, must be applied. □

Lemma 24 *Let a, $b \in NF$ and $d \in \Lambda s_{op}$, if $a \to_{\sigma-gen} d$ then $s_e(a\,\sigma^j b) \to_{\beta'} s_e(d\,\sigma^j b)$.*

Proof: The proof, by induction on a, is similar to that of the previous lemma. □

Lemma 25 *Let a, $b \in NF$ and $d \in \Lambda s_{op}$, if $b \to_{\sigma-gen} d$ then $s_e(a\,\sigma^j b) \twoheadrightarrow_{\beta'} s_e(a\,\sigma^j d)$.*

Proof: The proof is also similar to that of lemma 23. We remark that the induction is on a and the cases $a = X$ and $a = \mathtt{n}$ must now be considered. In particular, when $a = \mathtt{j}$, lemma 23 is required.

Lemma 20.4 is used in the case $a = c\,e$ and lemma 20.5 is useful to treat $a = \lambda c$.

For the subcases $k + i = j$ and $k + i < j$ of the case $a = \varphi_k^i c$ use lemma 22.3 and 22.4, respectively.

Finally, lemma 22.5 is needed when $a = c\,\sigma^i e$ and $i \geq j$. □

Lemma 26 *For a, b, $c \in NF$ the following hold:*

1. *If $a \to_{\beta'} b$ then $s_e(\varphi_k^i a) \to_{\beta'} s_e(\varphi_k^i b)$.*
2. *If $a \twoheadrightarrow_{\beta'} b$ then $s_e(\varphi_k^i a) \twoheadrightarrow_{\beta'} s_e(\varphi_k^i b)$.*
3. *If $a \to_{\beta'} b$ then $s_e(a\,\sigma^j c) \to_{\beta'} s_e(b\,\sigma^j c)$.*
4. *If $a \twoheadrightarrow_{\beta'} b$ then $s_e(a\,\sigma^j c) \twoheadrightarrow_{\beta'} s_e(b\,\sigma^j c)$.*
5. *If $b \to_{\beta'} c$ then $s_e(a\,\sigma^j b) \twoheadrightarrow_{\beta'} s_e(a\,\sigma^j c)$.*
6. *If $b \twoheadrightarrow_{\beta'} c$ then $s_e(a\,\sigma^j b) \twoheadrightarrow_{\beta'} s_e(a\,\sigma^j c)$.*

Proof: Let $d \in \Lambda s_{op}$ such that $a \to_{\sigma-gen} d$ and $b = s_e(d)$.

1. $s_e(\varphi_k^i a) \overset{L\,23}{\to}_{\beta'} s_e(\varphi_k^i d) = s_e(\varphi_k^i(s_e(d))) = s_e(\varphi_k^i b)$.
2. By induction on the length of the derivation using item 1.
3. As in item 1 using lemma 24.
4. By induction on the length of the derivation using item 3.
5. As in item 1 using lemma 25.
6. By induction on the length of the derivation using item 5. □

The following proposition states that condition 3 of the GIM 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 the IH and lemma 20.4. 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 20.5.

$a = \varphi_k^i c$**:** Use the IH and lemma 26.2.

$a = c\,\sigma^j d$**:** Use the IH and either lemma 26.4, if the reduction is within c, or lemma 26.6, if the reduction is within d. □

5 Confluence results

In this section we prove the confluence of the calculus of the interpretation $(NF, \to_{\beta'})$ in order to obtain the confluence of the λs_e-calculus via the GIM.

The confluence of $(NF, \to_{\beta'})$ is obtained via a parallelisation à la Tait-Martin-Löf (cf. the proof of the confluence of the classical λ-calculus in [Bar84]).

We define the parallelisation as follows:

Definition 17 *Let a, b, c, d, $a_1, \ldots, a_n \in NF$ The reduction $\Rightarrow$ is defined on NF by the following rules:*

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

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

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

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

We begin by proving that the transitive closures of $\to_{\beta'}$ and $\Rightarrow$ coincide. We must first establish two lemmas.

Lemma 27 *Let a, $b \in NF$, if $a \Rightarrow b$ then $a \twoheadrightarrow_{\beta'} b$.*

Proof: By induction on the lenght of the deduction $a \Rightarrow b$. Therefore we analise the last rule used in this deduction:

REFL: Obvious.

ABST: Hence $a = \lambda a'$, $b = \lambda b'$ and $a' \Rightarrow b'$. By IH, $a' \twoheadrightarrow_{\beta'} b'$, and conclude by lemma 20.5.

APPL: Use the IH and lemma 20.4.

SPHI: Now $a = \overline{\sigma\varphi}(a_1, \ldots, a_n)X$, $b = \overline{\sigma\varphi}(b_1, \ldots, b_n)X$ and $a_h \Rightarrow b_h$ for $1 \leq h \leq n$. By IH, $a_h \twoheadrightarrow_{\beta'} b_h$, and we conclude using the following:

Fact: For every normal skeleton $\overline{\sigma\varphi}$ of arity n and for every a_h, $b_h \in NF$ $(1 \leq h \leq n)$, if $a_h \twoheadrightarrow_{\beta'} b_h$ for $1 \leq h \leq n$, then $\overline{\sigma\varphi}(a_1, \ldots, a_n)X \twoheadrightarrow_{\beta'} \overline{\sigma\varphi}(b_1, \ldots, b_n)X$.

wich is proved by induction on the length oof the segment $\overline{\sigma\varphi}$ using lemma 22.4 and 22.5.

BETA: Now $a = (\lambda a')b'$, $b = s_e(c'\sigma^1 d')$, $a' \Rightarrow c'$ and $b' \Rightarrow d'$. By IH, $a' \to_{\beta'} c'$ and $b' \to_{\beta'} d'$ and therefore $(\lambda a')b' \to_{\beta'} s_e(a'\sigma^1 b') \overset{L\ 26.4}{\twoheadrightarrow_{\beta'}} s_e(c'\sigma^1 b') \overset{L\ 26.6}{\twoheadrightarrow_{\beta'}} s_e(c'\sigma^1 d')$. □

Remark 4 *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 $\overline{\sigma\varphi}(a_1,\ldots,a_n)X \twoheadrightarrow_{s_e} \overline{\sigma\varphi}(s_e(a_1),\ldots,s_e(a_n))X$ and this last term is an s_e-nf, we conclude by unicity of s_e-normal forms. □

Lemma 28 *Let $a \in NF$ and $d \in \Lambda s_{op}$, if $a \to_{\sigma-gen} d$ 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 $d = \overline{\sigma\varphi}(a_1,\ldots,a'_i,\ldots,a_n)$ with $a_i \to_{\sigma-gen} a'_i$. By IH, $a_i \Rightarrow s_e(a'_i)$ and, since $a_h \Rightarrow a_h$, applying rule *SPHI* we obtain

$$a \Rightarrow \overline{\sigma\varphi}(a_1,\ldots,s_e(a'_i),\ldots,a_n) \stackrel{R4}{=} s_e(\overline{\sigma\varphi}(a_1,\ldots,a'_i,\ldots,a_n))$$ □

Lemma 29 *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 28, $a \Rightarrow b$. Therefore, $\to_{\beta'} \subseteq \Rightarrow$.

Now, by lemma 27, $\Rightarrow \subseteq \twoheadrightarrow_{\beta'}$, hence $\to_{\beta'} \subseteq \Rightarrow \subseteq \twoheadrightarrow_{\beta'}$. Therefore, $\twoheadrightarrow_{\beta'} = \Rightarrow^*$. □

Remark 5 *If we prove now that $\Rightarrow$ is strongly confluent (SCR), lemma 1 would ensure that $\Rightarrow^*$ is SCR, hence $\twoheadrightarrow_{\beta'}$ would be SCR, which is equivalent to the confluence of $\to_{\beta'}$.*

To prove that $\Rightarrow$ is SCR we must first establish some facts. Again, we invite the reader, if she/he wishes, to read now the proof of theorem 12 and backtrack the necessity of the following lemmas.

Lemma 30 *For every $i \geq 1$, $k \geq 0$ and normal skeleton $\overline{\sigma\varphi}$ of arity n, there exist 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 $k_h \geq 0$ (for all h such that $1 \leq h \leq m$) 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 $\overline{\sigma\varphi}$.

We remark that $s_e(\varphi^i_k\,\overline{\sigma\varphi}(a_1,\ldots,a_n)X) = s'_e(\varphi^i_k\,\overline{\sigma\varphi}(a_1,\ldots,a_n)X)$, since $\overline{\sigma\varphi}(a_1,\ldots,a_n)X$ is a normal form.

If $\overline{\sigma\varphi}$ has length 1, we consider:

$\overline{\sigma\varphi} = (\varphi^j_l)$**:** A glimpse at definition 12 is sufficient to check that $\overline{\sigma\varphi_1} = (\varphi^i_k)(\varphi^j_l)$, for $k < l$; $\overline{\sigma\varphi_1} = (\varphi^{i+j-1}_l)$, for $l \leq k < l+j$ and $\overline{\sigma\varphi_1} = \varphi^j_l\varphi^i_{k+1-j}$, for $l+j \leq k$, are good choices. Remark that in this case $n = 0$, and therefore $m = 0$.

$\overline{\sigma\varphi} = (\,\sigma^j)$**:** Again definition 12 ensures that $\overline{\sigma\varphi_1} = (\varphi^i_k)(\,\sigma^j)$ and $m = 0$ (for $j > k+1$) and $\overline{\sigma\varphi_1} = (\,\sigma^j)(\varphi^i_{k+1})$, $m = 1$, $i_1 = i$, $k_1 = k+1-j$ (for $j \leq k+1$) are good choices. Remark that now $n = 1$.

For the inductive step we consider two cases acording to the first symbol of $\overline{\sigma\varphi}$:

$\overline{\sigma\varphi} = (\varphi^j_l)\overline{\sigma\varphi'}$**:** Definition 12 guarantees that $\overline{\sigma\varphi_1} = (\varphi^i_k)(\varphi^j_l)\overline{\sigma\varphi'}$ and $m = 0$ (for $k < l$) and $\overline{\sigma\varphi_1} = (\varphi^{i+j-1}_l)\overline{\sigma\varphi'}$ and $m = 0$ (for $l \leq k < l+j$) are good choices. Now for the remaining case ($l+j \leq k$) the IH is also necessary:

$$s_e(\varphi^i_k\ \overline{\sigma\varphi}(a_1,\ldots,a_n)X) = s_e(\varphi^i_k(\varphi^j_l\ \overline{\sigma\varphi'}(a_1,\ldots,a_n)X)) \stackrel{D12}{=}$$

$$\varphi^j_l s_e(\varphi^i_{k+1-j}\ \overline{\sigma\varphi'}(a_1,\ldots,a_n)X) \stackrel{IH}{=} (\varphi^j_l)\overline{\sigma\varphi'_1}(s_e(\varphi^{i_1}_{k_1}a_1),\ldots,s_e(\varphi^{i_m}_{k_m}a_m),a_{m+1},\ldots,a_n)X$$

$\overline{\sigma\varphi} = (\ \sigma^j)\overline{\sigma\varphi'}$**:** By definition 12 it is sufficient to take $\overline{\sigma\varphi_1} = (\varphi^i_k)(\ \sigma^j)\overline{\sigma\varphi'}$ and $m = 0$ for $j > k+1$. For $j \leq k+1$ also the IH is needed:

$$s_e(\varphi^i_k\ \overline{\sigma\varphi}(a_1,\ldots,a_n)X) = s_e(\varphi^i_k(\overline{\sigma\varphi'}(a_2,\ldots,a_n)X\ \sigma^j a_1)) \stackrel{D12}{=}$$

$$s_e(\varphi^i_{k+1}\ \overline{\sigma\varphi'}(a_2,\ldots,a_n)X)\ \sigma^j s_e(\varphi^i_{k+1-j}a_1) \stackrel{IH}{=}$$

$$(\ \sigma^j)\overline{\sigma\varphi'_1}(s_e(\varphi^i_{k+1-j}a_1), s_e(\varphi^{i_1}_{k_1}a_2),\ldots,s_e(\varphi^{i_m}_{k_m}a_{m+1}),a_{m+2},\ldots,a_n)X \qquad \Box$$

Lemma 31 *For every $j \geq 1$ and normal skeleton $\overline{\sigma\varphi}$ of arity n, there exist a normal skeleton $\overline{\sigma\varphi_2}$, m, $i_1,\ldots,i_m$ such that $0 \leq m \leq n$ and $i_h \geq 1$ (for all h such that $1 \leq h \leq m$) and such that 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 $m \leq p \leq n$, $i_h \geq 1$ and $k_h \geq 0$ (for all h such that $m+1 \leq h \leq p$) and such that for every $a_1,\ldots,a_n,c \in NF$ the following holds:*

$$s_e(\overline{\sigma\varphi}(a_1,\ldots,a_n)X\ \sigma^j\ c) =$$

$$\overline{\sigma\varphi_2}(s_e(a_1\sigma^{i_1}c),\ldots,s_e(a_m\sigma^{i_m}c),s_e(\varphi^{i_0}_0 c),s_e(\varphi^{i_{m+1}}_{k_{m+1}}a_{m+1}),\ldots,s_e(\varphi^{i_p}_{k_p}a_p),a_{p+1},\ldots,a_n)X$$

- *for every $a_1,\ldots,a_n,c \in NF$ the following holds:*

$$s_e(\overline{\sigma\varphi}(a_1,\ldots,a_n)X\ \sigma^j\ c) = \overline{\sigma\varphi_2}(s_e(a_1\sigma^{i_1}c),\ldots,s_e(a_m\sigma^{i_m}c),\ c,a_{m+1},\ldots,a_n)X$$

- *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 $\overline{\sigma\varphi}$. The proof follows the lines of the proof of the previous lemma. Now, definition 13 is used everywhere. When considering the case $\overline{\sigma\varphi} = (\varphi^i_k)\overline{\sigma\varphi'}$ lemma 30 is needed for the subcase $j = k+i$, whereas the IH is useful for the subcase $j > k+i$. Finally, when considering the case $\overline{\sigma\varphi} = (\ \sigma^i)\overline{\sigma\varphi'}$, the IH is needed if $i \leq j$. $\Box$

Lemma 32 *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 $a \Rightarrow b$. When the last rule is *REFL*, it is obvious and when it is *ABST* or *APPL* it is easy: just use the IH. Therefore we just consider the other two rules.

*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 30 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

$$\overline{\sigma\varphi_1}(s_e(\varphi_{k_1}^{i_1} a_1), \ldots, s_e(\varphi_{k_m}^{i_m} a_m), a_{m+1}, \ldots, a_n)X \Rightarrow$$

$$\overline{\sigma\varphi_1}(s_e(\varphi_{k_1}^{i_1} b_1), \ldots, s_e(\varphi_{k_m}^{i_m} b_m), b_{m+1}, \ldots, b_n)X \stackrel{L30}{=} s_e(\varphi_k^i\, \overline{\sigma\varphi}(b_1, \ldots, b_n)X)$$

BETA: Hence $a = (\lambda a_1)a_2$, $b = s_e(b_1\sigma^1 b_2)$, $a_1 \Rightarrow b_1$ and $a_2 \Rightarrow b_2$.

$$s_e(\varphi_k^i((\lambda a_1)a_2)) = (\lambda s_e(\varphi_{k+1}^i a_1))s_e(\varphi_k^i a_2) \stackrel{IH}{\Rightarrow} s_e(s_e(\varphi_{k+1}^i b_1)\sigma^1 s_e(\varphi_k^i b_2)) =$$
$$s_e((\varphi_{k+1}^i b_1)\sigma^1(\varphi_k^i b_2)) = s_e(\varphi_k^i(b_1\sigma^1 b_2)) = s_e(\varphi_k^i s_e(b_1\sigma^1 b_2)) \qquad \square$$

The following corollary is immediate:

Corollary 3 *Let $a, b \in NF$ such that $\varphi_k^i a, \varphi_k^i b \in NF$, if $a \Rightarrow b$ then $\varphi_k^i a \Rightarrow \varphi_k^i b$.*

Lemma 33 *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. The cases $a = X$, $a = d\,e$, $a = \lambda d$, and $a = \mathtt{n} \neq \mathtt{j}$ are easy and only require eventually the IH. Therefore we just consider the other cases.

$a = \mathtt{j}$: $s_e(\mathtt{j}\,\sigma^j b) = s_e(\varphi_0^j b) \stackrel{L32}{\Rightarrow} s_e(\varphi_0^j c) = s_e(\mathtt{j}\,\sigma^j c)$.

$a = \varphi_k^i d$: We consider four cases:

- If $j \leq k$ then $s_e((\varphi_k^i d)\,\sigma^j b) = (\varphi_k^i d)\,\sigma^j b \stackrel{SPHI}{\Rightarrow} (\varphi_k^i d)\,\sigma^j c = s_e((\varphi_k^i d)\,\sigma^j c)$.
  We precise once and for all how we applied the rule *SPHI*.
  If $d = \overline{\sigma\varphi}(d_1, \ldots, d_n)X$ then $(\varphi_k^i d)\,\sigma^j b = (\,\sigma^j)(\varphi_k^i)\overline{\sigma\varphi}(b, d_1, \ldots, d_n)X$, hence *SPHI* has been applied with premises $b \Rightarrow c$ and $d_h \Rightarrow d_h$ for all h to the skeleton $(\,\sigma^j)(\varphi_k^i)\overline{\sigma\varphi}$.
  If $d = X$, *SPHI* applies with the single premise $b \Rightarrow c$.
- If $k < j < k + i$ then $s_e((\varphi_k^i d)\,\sigma^j b) = \varphi_k^{i-1} d \stackrel{REFL}{\Rightarrow} \varphi_k^{i-1} d = s_e((\varphi_k^i d)\,\sigma^j c)$.
- If $j = k + i$, remark that for all $e' \in NF$ we have:
  $$s_e((\varphi_k^i d)\,\sigma^j e') = s_e^*((\varphi_k^i d)\,\sigma^j e') \stackrel{D14}{=} s_e''((\varphi_k^i d)\,\sigma^j e') \stackrel{D13}{=}$$
  $$s_e'(\varphi_{k+1}^i d)\sigma^{k+1} s_e'(\varphi_0^i e') = s_e(\varphi_{k+1}^i d)\sigma^{k+1} s_e(\varphi_0^i e')$$
  Therefore,
  $$s_e((\varphi_k^i d)\,\sigma^j b) = s_e(\varphi_{k+1}^i d)\sigma^{k+1} s_e(\varphi_0^i b) \stackrel{L32\,\&\,SPHI}{\Rightarrow}$$
  $$s_e(\varphi_{k+1}^i d)\sigma^{k+1} s_e(\varphi_0^i c) = s_e((\varphi_k^i d)\,\sigma^j c)$$
- If $j > k + i$, remark (using definitions 14 and 13) that for all $e' \in NF$, we have
  $$s_e((\varphi_k^i d)\,\sigma^j e') = \varphi_k^i s_e(d\,\sigma^{j+1-i} e'),$$
  hence
  $$s_e((\varphi_k^i d)\,\sigma^j b) = \varphi_k^i s_e(d\,\sigma^{j+1-i} b) \stackrel{IH\,\&\,C3}{\Rightarrow} \varphi_k^i s_e(d\,\sigma^{j+1-i} c) = s_e((\varphi_k^i d)\,\sigma^j c)$$

$a = d\,\sigma^i e$: We consider two cases:

- If $i > j$ then $s_e((d\,\sigma^i e)\,\sigma^j b) = (d\,\sigma^i e)\,\sigma^j b \stackrel{SPHI}{\Rightarrow} (d\,\sigma^i e)\,\sigma^j c = s_e((d\,\sigma^i e)\,\sigma^j c)$.

- If $i \leq j$, remark (using definitions 14 and 13) that for all $e' \in NF$ we have:

$$s_e((d\,\sigma^i e)\,\sigma^j e') = s_e(d\,\sigma^{j+1} e')\,\sigma^i s_e(e\,\sigma^{j-i+1} e') \quad (1)$$

Now, since a is a normal form, d is a $\sigma\varphi$-normal form or a variable, hence $s_e(d\,\sigma^{j+1} b)$ is a $\sigma\varphi$-normal form (it cannot be a variable since $s_e(a') = X$ implies $a' = X$ which can be easily checked using definitions 12, 13 and 14). By IH, $s_e(d\,\sigma^{j+1} b) \Rightarrow s_e(d\,\sigma^{j+1} c)$. Therefore, the last rule in this deduction must be *SPHI* or *REFL*. Let us write $s_e(d\,\sigma^{j+1} b) = \overline{\sigma\varphi}(b_1, \ldots, b_n)X$, hence $s_e(d\,\sigma^{j+1} c) = \overline{\sigma\varphi}(c_1, \ldots, c_n)X$ and $b_h \Rightarrow c_h$ for $1 \leq h \leq n$ (if the last rule was $REFL\ b_h = c_h$). But the IH also ensures $s_e(e\,\sigma^{j+1} b) \Rightarrow s_e(e\,\sigma^{j+1} c)$, and we can apply *SPHI* to obtain

$$(\,\sigma^i)\overline{\sigma\varphi}(s_e(e\,\sigma^{j-i+1} b), b_1, \ldots, b_n) \Rightarrow (\,\sigma^i)\overline{\sigma\varphi}(s_e(e\,\sigma^{j-i+1} c), c_1, \ldots, c_n)$$

which can be written as

$$s_e(d\,\sigma^{j+1} b)\,\sigma^i s_e(e\,\sigma^{j-i+1} b) \Rightarrow s_e(d\,\sigma^{j+1} c)\,\sigma^i s_e(e\,\sigma^{j-i+1} c)$$

and we settle the lemma by using equation 1 with $e' = b$ and $e' = c$. □

Lemma 34 *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 $a \Rightarrow b$.

We study the last rule applied. If it is *REFL* our lemma is just lemma 33. If it is *APPL* or *ABST* the lemma follows directly from the IH. Let us study the other two rules:

BETA: Hence $a = (\lambda a_1)a_2$, $b = s_e(b_1 \sigma^1 b_2)$, $a_1 \Rightarrow b_1$ and $a_2 \Rightarrow b_2$.

$$s_e(((\lambda a_1)a_2)\,\sigma^j c) = (\lambda s_e(a_1\,\sigma^{j+1} c))s_e(a_2\,\sigma^j c) \overset{BETA}{\Rightarrow} s_e(s_e(b_1\,\sigma^{j+1} d)\sigma^1 s_e(b_2 \sigma^j d)) =$$
$$= s_e((b_1\,\sigma^{j+1} d)\sigma^1 (b_2 \sigma^j d)) = s_e((b_1 \sigma^1 b_2)\,\sigma^j d) = s_e(s_e(b_1 \sigma^1 b_2)\,\sigma^j d)$$

SPHI: Hence $a = \overline{\sigma\varphi}(a_1, \ldots, a_n)X$, $b = \overline{\sigma\varphi}(b_1, \ldots, b_n)X$ and $a_h \Rightarrow b_h$ for $1 \leq h \leq n$. Now, lemma 31 offers three possibilities which can be treated analogously. Let us choose, for example, 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 since $c \Rightarrow d$ and $a_h \Rightarrow b_h$ for $m+1 \leq h \leq n$, we can apply *SPHI* to obtain

$$\overline{\sigma\varphi_2}(s_e(a_1 \sigma^{i_1} c), \ldots, s_e(a_m \sigma^{i_m} c),\ c, a_{m+1}, \ldots, a_n)X \Rightarrow$$
$$\overline{\sigma\varphi_2}(s_e(b_1 \sigma^{i_1} d), \ldots, s_e(b_m \sigma^{i_m} d),\ d, b_{m+1}, \ldots, b_n)X$$

Finally, using lemma 31 again, we have:

$$s_e(\overline{\sigma\varphi}(b_1, \ldots, b_n)X\ \sigma^j\, d) = \overline{\sigma\varphi_2}(s_e(b_1 \sigma^{i_1} d), \ldots, s_e(b_m \sigma^{i_m} d),\ d, b_{m+1}, \ldots, b_n)X$$

Remark that, in order to check the first option of lemma 31, also lemma 32 must be used. □

Theorem 12 *The parallelisation $\Rightarrow$ is strongly confluent, i.e. if $a \Rightarrow b$ and $a \Rightarrow c$ then there exists d such that $b \Rightarrow d$ and $c \Rightarrow d$.*

Proof: By induction on the length of the deduction $a \Rightarrow b$.

REFL: Hence $a = b$, and take $d = c$.

ABST: Hence $a = \lambda a'$, $b = \lambda b'$ and $a' \Rightarrow b'$. Remark that $c = \lambda c'$ and $a' \Rightarrow c'$, since the last rule applied to obtain $\lambda a' \Rightarrow c$ must be either *ABST* or *REFL*. By IH there exists d' such that $b' \Rightarrow d'$ and $c' \Rightarrow d'$ and it is sufficient to take $d = \lambda d'$.

APPL: Hence $a = a_1 a_2$ and $b = b_1 b_2$ with $a_1 \Rightarrow b_1$ and $a_2 \Rightarrow b_2$. Now there are two possibilities for c according to the last rule applied to obtain $a \Rightarrow c$.

- If the last rule is *APPL* or *REFL* then $c = c_1 c_2$ with $a_1 \Rightarrow c_1$ and $a_2 \Rightarrow c_2$. By IH there exist d_1 and d_2 such that $b_1 \Rightarrow d_1$, $c_1 \Rightarrow d_1$, $b_2 \Rightarrow d_2$ and $c_2 \Rightarrow d_2$. Take $d = d_1 d_2$.
- If the last rule is *BETA* then $c = s_e(c_1\, \sigma^1 c_2)$, $a_1 = \lambda a'$, $a' \Rightarrow c_1$ and $a_2 \Rightarrow c_2$. Since $\lambda a' \Rightarrow b_1$, according to the remark we have made when studying the *ABST* case, $b_1 = \lambda b'$ and $a' \Rightarrow b'$. By IH, there exists d_1 and d_2 such that $b' \Rightarrow d_1$, $c_1 \Rightarrow d_1$, $b_2 \Rightarrow d_2$ and $c_2 \Rightarrow d_2$. Take $d = s_e(d_1\, \sigma^1 d_2)$. Applying *BETA* we get $b \Rightarrow d$ and lemma 34 guarantees $c \Rightarrow d$.

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

BETA: Hence $a = (\lambda a_1)a_2$, $b = s_e(b_1 \sigma^1 b_2)$, $a_1 \Rightarrow b_1$ and $a_2 \Rightarrow b_2$. Reasoning as in the *APPL* case, there are two possibilities for c.

- If $c = c_1 c_2$ the procedure is symetric to the one used to treat the second subcase of the *APPL* case.
- If $c = s_e(c_1 \sigma^1 c_2)$, with $a_1 \Rightarrow c_1$ and $a_2 \Rightarrow c_2$, then by IH there exist d_1, d_2 such that $b_1 \Rightarrow d_1$, $c_1 \Rightarrow d_1$, $b_2 \Rightarrow d_2$ and $c_2 \Rightarrow d_2$. Take $d = s_e(d_1 \sigma^1 d_2)$ and use lemma 34 to settle the theorem. □

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

Proof: The proof is given by remark 5. □

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

Proof: All the conditions are satisfied (see the four propositions of this article) and the generalized interpretation method (cf. lemma 4) can be applied as we proposed at the beginning of section 4. □

Since all the λs_e-rules preserve closed terms, we have:

Corollary 4 *The λs_e-calculus is confluent on Λs.*

6 Soundness

We end our work by applying some of our results to establish a soundness theorem, namely we show that the λs_e-calculus 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.

Remark 6 *For all $a \in \Lambda s$, we have $s_e(a) = s(a)$.*

Proof: $a \twoheadrightarrow_{s_e} s(a)$, since $a \twoheadrightarrow_s s(a)$. But, by lemma 12, $s(a) \in \Lambda$, and since $\Lambda \subset NF$, the confluence of s_e (theorem 11) implies $s_e(a) = s(a)$. □

Lemma 35 *For all $a, b \in \Lambda$, $a \to_{\beta'} b$ iff $a \to_\beta b$.*

Proof: Both implications can be proved by induction on a. We just check ($\Rightarrow$) which is the one we are going to use.

Let $d \in \Lambda s$, such that $a \to_{\sigma-gen} d$ and $b = s_e(d)$.

$a = a_1 a_2$**:** If the reduction takes place within a_1, say $a_1 \to_{\sigma-gen} a_1'$, then $d = a_1' a_2$ and $b = s_e(a_1' a_2) = s_e(a_1') a_2$. Since $a_1 \to_{\beta'} s_e(a_1')$, by IH we have $a_1 \to_\beta s_e(a_1')$. The lemma is settled using the compatibility of β.

If the reduction takes place within a_2, the proof is analogous.

If the reduction takes place at the root, i.e. $a_1 = \lambda c_1$ and $b = s_e(c_1\, \sigma^1 a_2)$. By remark 6, $b = s(c_1\, \sigma^1 a_2)$, and since $c_1,\ a_2 \in \Lambda$, by lemma 13, $b = c_1 \{\!\{ 1 \leftarrow a_2 \}\!\}$. Therefore, $a \to_\beta b$.

$a = \lambda c$**:** Analogous to the first part of the previous case. □

Corollary 5 *For all $a, b \in \Lambda$, $a \twoheadrightarrow_{\beta'} b$ iff $a \twoheadrightarrow_\beta b$.*

Proof: By induction on the length of the derivations. □

Lemma 36 *Let $a, b \in \Lambda s_{op}$, if $a \twoheadrightarrow_{\lambda s_e} b$ then $s_e(a) \twoheadrightarrow_{\beta'} s_e(b)$.*

Proof: By induction on the length of the derivation $a \twoheadrightarrow_{\lambda s_e} b$. When the last reduction is $a_n \to_{s_e} b$, conclude by proposition 1. When it is $a_n \to_{\sigma-gen} b$, conclude by proposition 3. □

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

Proof: If $a \twoheadrightarrow_{\lambda s_e} b$, by lemma 36, we get $s_e(a) \twoheadrightarrow_{\beta'} s_e(b)$, and, since $a, b \in \Lambda \subset NF$, we conclude using corollary 5. □

Conclusion

We think that λs is an interesting alternative to calculi of explicit substitutions in the $\lambda\sigma$-style: it preserves SN (cf. [KR95a]), has a confluent extension on open terms (cf. theorem 13) and simulates one step β-reduction (cf. lemma 15). 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 [MH95], 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. [Klo91]):

Lemma 37 *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., $\exists$ a function $f : T \longrightarrow \mathbb{N}$ where $a \rightarrow b \Rightarrow f(a) < f(b)$.*

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 38 *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 $\forall r = 1, \cdots, n$ 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 39 *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 $\forall r = 1, \cdots, n$ 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 38. □

Lemma 40 (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 39 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 the proof above is the first one he ever found establishing strong normalisation from weak normalisation. Finally, he remarks that lemma 37 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

[ACCL91] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit Substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[Bar84] H. Barendregt. *The Lambda Calculus : Its Syntax and Semantics (revised edition)*. North Holland, 1984.

[BBLRD95] Z. Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. λv, a calculus of explicit substitutions which preserves strong normalisation. *Personal communication*, 1995.

[Blo95] R. Bloo. Preservation of Strong Normalisation for Explicit Substitution. Technical Report 95-08, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1995.

[CHL92] 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. To appear in the JACM.

[Cur86] P.-L. Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming*. Pitman, 1986. Revised edition : Birkhäuser (1993).

[dB78] N. G. de Bruijn. A namefree lambda calculus with facilities for internal definition of expressions and segments. Technical Report TH-Report 78-WSK-03, Department of Mathematics, Eindhoven University of Technology, 1978.

[Har89] T. Hardin. Confluence Results for the Pure Strong Categorical Logic CCL : λ-calculi as Subsystems of CCL. *Theoretical Computer Science*, 65(2):291–342, 1989.

[HL89] T. Hardin and J.-J. Lévy. A Confluent Calculus of Substitutions. *France-Japan Artificial Intelligence and Computer Science Symposium*, December 1989.

[Hue80] G. Huet. Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. *Journal of the Association for Computing Machinery*, 27:797–821, October 1980.

[KB70] D. Knuth and P. Bendix. Simple Word Problems in Universal Algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[Klo91] J.-W. Klop. Term rewriting systems. *Handbook of Theortetical Computer Science*, II, 1991.

[KN93] F. Kamareddine and R. P. Nederpelt. On stepwise explicit substitution. *International Journal of Foundations of Computer Science*, 4(3):197–240, 1993.

[KN95] F. Kamareddine and R. P. Nederpelt. Refining reduction in the λ-calculus. *Journal of Functional Programming*, 5(4), 1995.

[KR95a] F. Kamareddine and A. Ríos. A λ-calculus à la de Bruijn with explicit substitutions. To appear in the Proceedings of PLILP'95. *Lecture Notes in Computer Science*, 1995.

[KR95b] F. Kamareddine and A. Ríos. The λs-calculus: its typed and its extended versions. Technical report, Department of Computing Science, University of Glasgow, 1995.

[Mel95] P.-A. Melliès. Typed λ-calculi with explicit substitutions may not terminate in Proceedings of TLCA'95. *Lecture Notes in Computer Science*, 902, 1995.

[MH95] C. A. Muñoz Hurtado. Confluence and preservation of strong normalisation in an explicit substitutions calculus. Technical report, INRIA, Rocquencourt, December 1995. 2762.

[Río93] A. Ríos. *Contribution à l'étude des λ-calculs avec substitutions explicites.* PhD thesis, Université de Paris 7, 1993.