The λs-calculus: its typed and its extended versions

Fairouz Kamareddine and Alejandro Ríos [*]

June 26, 1995

Abstract

We present in this paper the simply typed version of the λs-calculus (cf. [KR95]) and prove the strong normalisation of the well typed terms. We also present an extension of the λs-calculus: the λs_e-calculus and prove its local confluence on open terms and the weak normalisation of its corresponding calculus of substitutions s_e. The strong normalisation of s_e is still an open problem to challenge the rewriting community.

Introduction

In [KR95] we presented a new λ-calculus in de Bruijn notation with explicit substitutions: the λs-calculus, offering thus a different perspective on the problem of closing the gap between the classical λ-calculus and concrete implementations. We showed in [KR95] that the λs-calculus works pretty well with closed terms (the set of these terms is large enough to contain all the λ-terms). That is, we proved the confluence of the λs-calculus on closed terms and the preservation of strong normalisation with respect to classical λ-terms (this property is seldom present in calculi with explicit substitutions and states that every λ-term strongly normalising in the λ-calculus is also strongly normalising in the λs-calculus).

In the conclusion of [KR95] we presented two open problems:

Open Problem 1: *Are well typed terms strongly normalising in the simply typed λs-calculus?*

In section 2 we introduce the simply typed λs-calculus, show subject reduction and give an affirmative answer to the question of strong normalisation.

Open Problem 2: *Can the λs-calculus be extended to a confluent calculus on open terms?*

Let us explain briefly this question. The λs-calculus is not confluent (not even locally confluent) when working on open terms. In order to get local confluence one must add several rules which correspond to properties of the updating and substitution meta-operators (see lemmas 1-6 in [KR95]).

In section 3 we introduce the λs_e-calculus as the λs-calculus extended by the addition of these rules and prove its *local confluence* by studying the critical pairs. In order to prove *(global) confluence* using the interpretation technique (cf. [Har89] and [CHL91]) we are led to study the corresponding calculus of substitutions s_e (the calculus obtained by deleting the rule which starts β-reduction). We prove the weak normalisation of s_e by showing that innermost strategies always terminate. The strong normalisation is, as far as we know, still an open problem and a challenge to the rewriting specialists.

[*]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

The main interest in studying such an extension is to provide a calculus of explicit substitutions which would have both the property of preserving strong normalisation and a confluent extension on open terms. As far as we know no such calculus has yet been proposed. There are calculi of explicit substitutions which are confluent on open terms: the $\lambda\sigma_{\Uparrow}$- calculus (cf. [HL89] and [CHL91]), but the non-preservation of strong normalisation for $\lambda\sigma_{\Uparrow}$ 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].

1 The s- and λs-calculi

We recall in this section the syntax of the λs-terms, the rules of the λs-calculus and the results obtained in [KR95].

Definition 1 *The set of terms, noted Λs, of the λs-calculus is given as follows:*

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

$\mathbb{N}$ *denotes the set of positive natural numbers. We take a, b, c to range over Λs. A term of the form $a\,\sigma^i b$ is called a* closure. *Furthermore, a term containing neither σ's nor φ's is called a* pure term. *The set of pure terms is denoted by Λ.*

We assume the reader familiar with de Bruijn notation. Let us just say here that de Bruijn indices (or numbers) are used to explicit the bindings: to find the λ which binds a variable represented by the number `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 $\lambda\mathtt{43}$.

Besides the de Bruijn indices, application and abstraction we have the explicit substituions operators σ^i and the updating operators φ^i_k. The term $a\,\sigma^i b$ should be intuitively understood as the term a where the substitution of the variable corresponding to the de Bruijn number `i` by the term b must be performed. The update operators φ^i_k are intended to update the de Bruijn numbers in such a way that when substitutions take place, the bindings remain correct. We refer to [KR95] for an introduction of these operators.

Definition 2 *The λs-*calculus *is given by the rewriting rules in Figure 1. We use λs to denote this set of rules. The calculus of substitutions associated with the λs-calculus is the rewriting system whose rules are $\lambda s - \{\sigma\text{-generation}\}$ and we call it the s-calculus.*

The *σ-generation* rule starts β-reduction by generating a substitution operator at the first level (σ^1). The *σ-app* and *σ-λ* rules allow this operator to travel throughout the term until its arrival to the variables. If a variable should be affected by the substituion, the *σ-destruction* rules (case $i = n$) 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 results obtained in [KR95]:

$$
\begin{array}{llcl}
\sigma\textit{-generation} & (\lambda a)\, b & \longrightarrow & a\, \sigma^1\, b \\
\sigma\text{-}\lambda\textit{-transition} & (\lambda a)\, \sigma^i b & \longrightarrow & \lambda(a\, \sigma^{i+1}\, b) \\
\sigma\textit{-app-transition} & (a_1\, a_2)\, \sigma^i b & \longrightarrow & (a_1\, \sigma^i b)\, (a_2\, \sigma^i b) \\
\sigma\textit{-destruction} & \mathtt{n}\, \sigma^i b & \longrightarrow & \begin{cases} \mathtt{n-1} & \textit{if} \quad n > i \\ \varphi_0^i\, b & \textit{if} \quad n = i \\ \mathtt{n} & \textit{if} \quad n < i \end{cases} \\
\varphi\text{-}\lambda\textit{-transition} & \varphi_k^i(\lambda a) & \longrightarrow & \lambda(\varphi_{k+1}^i\, a) \\
\varphi\textit{-app-transition} & \varphi_k^i(a_1\, a_2) & \longrightarrow & (\varphi_k^i\, a_1)\, (\varphi_k^i\, a_2) \\
\varphi\textit{-destruction} & \varphi_k^i\, \mathtt{n} & \longrightarrow & \begin{cases} \mathtt{n+i-1} & \textit{if} \quad n > k \\ \mathtt{n} & \textit{if} \quad n \le k \end{cases}
\end{array}
$$

Figure 1: The λs-calculus

Theorem 1 *The s-calculus is confluent and strongly normalising on Λs. Hence, every term a has a unique s-normal form denoted $s(a)$.*

Theorem 2 (Confluence) *The λs-calculus is confluent on Λs.*

Theorem 3 (Preservation of strong normalisation) *Pure terms which are strongly normalising in the λ-calculus are also strongly normalising in the λs-calculus.*

The λs-calculus is powerful enough to simulate β-reduction on pure terms:

Theorem 4 (Simulation of β-reduction) *Let a, $b \in \Lambda$, if $a \to_\beta b$ then $a \twoheadrightarrow_{\lambda s} b$.*

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:

Theorem 5 (Soundness) *Let a, $b \in \Lambda$, if $a \twoheadrightarrow_{\lambda s} b$ then $a \twoheadrightarrow_\beta b$.*

2 The typed λs-calculus

The proof of strong normalisation of well typed terms that we give in this section follows an original idea of Melliès (personal communication), which is based on the technique developped in [BBLRD95] to prove the preservation of strong normalisation in the $\lambda \upsilon$-calculus.

We shall recall first the syntax and typing rules for the simply typed λ-calculus in de Bruijn notation. The types are generated from a set of basic types K with the binary type operator $\to$. Environments are lists of types. Typed terms differ from the untyped ones only in the abstractions which are now marked with the type of the abstracted variable.

Definition 3 *The syntax for the simply typed λ-terms is given as follows:*

$$\begin{array}{lcll} \textbf{Types} & A & ::= & K \mid A \to A \\ \textbf{Environments} & E & ::= & nil \mid A, E \\ \textbf{Terms} & \Lambda & ::= & \mathtt{n} \mid \Lambda\,\Lambda \mid \lambda A.\Lambda \end{array}$$

The typing rules are given by the typing system **L1** *as follows:*

$$(\mathbf{L1}-var) \quad A, E \vdash \mathtt{1} : A \qquad\qquad (\mathbf{L1}-\lambda) \quad \frac{A, E \vdash b : B}{E \vdash \lambda A.b : A \to B}$$

$$(\mathbf{L1}-varn) \quad \frac{E \vdash \mathtt{n} : B}{A, E \vdash \mathtt{n}+\mathtt{1} : B} \qquad\qquad (\mathbf{L1}-app) \quad \frac{E \vdash b : A \to B \quad E \vdash a : A}{E \vdash b\, a : B}$$

Before presenting the simply typed λs-calculus we must introduce the following notation concerning environments. If E is the environment $E_1, E_2, \ldots, E_n$, we shall use the notation $E_{\geq i}$ for the environment $E_i, E_{i+1}, \ldots, E_n$, analogously $E_{\leq i}$ stands for $E_1, \ldots, E_i$, etc.

The typing rules for the σ and φ operators that we introduce below may be derived from the typing rules of the simply typed $\lambda\sigma$-calculus (cf. [ACCL91]) and the translation we gave for the λs-calculus into the $\lambda\sigma$-calculus (cf. [KR95]).

Definition 4 *The syntax for the simply typed λs-terms is given as follows (types and environments are defined as for λ-terms):*

$$\textbf{Terms} \quad \Lambda s_t \quad ::= \quad \mathbb{N} \mid \Lambda s_t\, \Lambda s_t \mid \lambda A.\Lambda s_t \mid \Lambda s_t\, \sigma^i \Lambda s_t \mid \varphi^i_k \Lambda s_t \qquad i \geq 1\,, \quad k \geq 0$$

The typing rules are given by the typing system **Ls1** *as follows:*

The rules **Ls1**-*var,* **Ls1**-*varn,* **Ls1**-λ *and* **Ls1**-*app are exactly the same as* **L1**-*var,* **L1**-*varn,* **L1**-λ *and* **L1**-*app, respectively. The new rules are:*

$$(\mathbf{Ls1}-\sigma) \quad \frac{E_{\geq i} \vdash b : B \quad E_{<i}, B, E_{\geq i} \vdash a : A}{E \vdash a\, \sigma^i b : A} \qquad\qquad (\mathbf{Ls1}-\varphi) \quad \frac{E_{\leq k}, E_{\geq k+i} \vdash a : A}{E \vdash \varphi^i_k a : A}$$

Definition 5 *We say that $a \in \Lambda s_t$ is a* well typed term *if there exists an environment E and a type A such that $E \vdash_{\mathbf{Ls1}} a : A$. We note Λs_{wt} the set of well typed terms.*

The aim of this section is to prove that every well typed λs-term a is strongly normalising in the λs-calculus (denoted $a \in \lambda s$-SN). We proceed by showing $\Lambda s_{wt} \subseteq S \subseteq \lambda s$-SN, where S is defined by:

Definition 6 $S = \{a \in \Lambda s_t :$ *for every subterm b of a, $s(b) \in \lambda$-SN$\}$ where λ-SN is the set of strongly normalising terms in the λ-calculus.*

To prove $\Lambda s_{wt} \subseteq S$ we need to establish some results:

Lemma 1 (Subject reduction) *If $E \vdash_{\mathbf{Ls1}} a : A$ and $a \to_{\lambda s} b$ then $E \vdash_{\mathbf{Ls1}} b : A$.*

Proof: By induction on a. If the reduction is not at the root, use the inductive hypothesis. If it is, check that for each rule $a \to b$ we have $E \vdash_{\mathbf{Ls1}} a : A$ implies $E \vdash_{\mathbf{Ls1}} b : A$. □

Corollary 1 *Let $E \vdash_{\mathbf{Ls1}} a : A$, if $a \twoheadrightarrow_{\lambda s} b$ then $E \vdash_{\mathbf{Ls1}} b : A$.*

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

Lemma 2 (Typing of subterms) *If $a \in \Lambda s_{wt}$ and b is a subterm of a then $b \in \Lambda s_{wt}$.*

Proof: By induction on a. If b is not an immediate subterm of a, use the induction hypothesis. Otherwise, the last rule used to type a must contain a premise in which b is typed. □

Lemma 3 (Soundness of typing) *If $a \in \Lambda$ and $E \vdash_{\mathbf{Ls1}} a : A$ then $E \vdash_{\mathbf{L1}} a : A$.*

Proof: Easy induction on a. □

Proposition 1 $\Lambda s_{wt} \subseteq S$

Proof: Let $a \in \Lambda s_{wt}$ and let b a subterm of a. By lemma 2, $b \in \Lambda s_{wt}$ and by corollary 1, $s(b) \in \Lambda s_{wt}$. Since $s(b) \in \Lambda$ (cf. lemma 8 in [KR95]) lemma 3 yields that $s(b)$ is **L1**-typable, and it is well known that classical typable λ-terms are strongly normalising in the λ-calculus. Hence, $s(b) \in \lambda$-SN and therefore $a \in S$. □

We shall prove now $S \subseteq \lambda s$-SN. This proof is very close to the proof of preservation of normalisation we gave in [KR95], theorem 6. We remind here some definitions and results in [KR95]. In the remainder of this section $a \to b$ will mean $a \to_{\lambda s} b$. We use the notation $C[.]$ to mean a context, (a term with a hole), and $C[d]$ to mean the term obtained by filling the hole with the term d.

Lemma 4 *Let $a_1 \to \ldots \to a_n \to a_{n+1} = C[d\sigma^i e]$ then $a_1 = C'[d'\sigma^j e']$ or there exists $k \leq n$ such that $a_k = C''[(\lambda d')e']$ and $a_{k+1} = C''[d'\sigma^1 e']$. In both cases $e' \twoheadrightarrow e$.*

Proof: This is lemma 14 in [KR95]. □

Definition 7 *A reduction is* internal *if the redex is a subterm of the right operand of a σ-operator.*

Lemma 5 *If $a \in S$ then for every infinite λs-derivation $a \to_{\lambda s} b_1 \to_{\lambda s} \cdots \to_{\lambda s} b_n \to_{\lambda s} \cdots$, there exists N such that for $i \geq N$ all the reductions $b_i \to_{\lambda s} b_{i+1}$ are internal.*

Proof: The proof is almost the same as the proof of lemma 16 in [KR95]. □

Notation 1 *We write $a \underset{p}{\to} b$ in order to denote that p is the occurrence of the redex which is contracted. We denote by $\prec$ the prefix order between occurrences of a term. Therefore if a is a term and p, q are occurrences of a such that $p \prec q$, and we write a_p (resp. a_q) for the subterm of a at occurrence p (resp. q), then a_q is a subterm of a_p.*

For example, if $a = \mathtt{2}\sigma^3((\lambda\mathtt{1})\mathtt{4})$, we have $a_1 = \mathtt{2}$, $a_2 = (\lambda\mathtt{1})\mathtt{4}$, $a_{21} = \lambda\mathtt{1}$, $a_{211} = \mathtt{1}$, $a_{22} = \mathtt{4}$.

Definition 8 *An infinite λs-derivation $a_1 \to \cdots \to a_n \to \cdots$ is* minimal *if for every step of reduction $a_i \underset{p}{\to}_{\lambda s} a_{i+1}$, every other derivation beginning with $a_i \underset{q}{\to}_{\lambda s} a'_{i+1}$ where $p \prec q$, is finite.*

The intuitive idea of a minimal derivation is that if one rewrites at least one of its steps at a lower occurrence (i.e. within a proper subterm of the actual redex), an infinite derivation is then impossible.

Definition 9 Skeletons *are defined by the following syntax:*

$$\textbf{Skeletons} \quad K ::= \mathbb{N} \mid K\,K \mid \lambda K \mid K\,\sigma^i[.] \mid \varphi^i_k K$$

The skeleton of a term a *is defined by induction as follows:*

$$Sk(\mathtt{n}) = \mathtt{n} \qquad Sk(a\,b) = Sk(a)Sk(b) \qquad Sk(a\,\sigma^i b) = Sk(a)\,\sigma^i[.]$$
$$Sk(\lambda a) = \lambda Sk(a) \qquad Sk(\varphi^i_k a) = \varphi^i_k Sk(a)$$

Remark 1 *If $a \xrightarrow{\mathtt{int}}_{\lambda s} b$ then $Sk(a) = Sk(b)$.*

Proposition 2 *For every $a \in \Lambda s_t$, if $a \in S$ then $a \in \lambda s$-SN.*

Proof: Suppose there exists $a' \in S$ and $a' \notin \lambda s$-SN, then there must exist a term a of minimal length such that $a \in S$ and $a \notin \lambda s$-SN.

Let us consider a minimal infinite λs-derivation $\mathcal{D}\ :\ a \to a_1 \to \cdots \to a_n \to \cdots$. By lemma 5, there exists N, such that for $i \geq N$, $a_i \to a_{i+1}$ is internal. Therefore, by the previous remark, $Sk(a_i) = Sk(a_{i+1})$ for $i \geq N$. As there are only a finite number of closures in $Sk(a_N)$ and as the reductions within these closures are independent, an infinite subderivation of $\mathcal{D}$ must take place within the same and unique closure in $Sk(a_N)$ and, evidently, this subderivation is also minimal. Let us call it $\mathcal{D}'$ and let C be the context such that $a_N = C[c\,\sigma^i d]$ and $c\,\sigma^i d$ is the closure where $\mathcal{D}'$ takes place. Therefore we have:

$$\mathcal{D}'\ :\ a_N = C[c\,\sigma^i d] \xrightarrow{\mathtt{int}}_{\lambda s} C[c\,\sigma^i d_1] \xrightarrow{\mathtt{int}}_{\lambda s} \cdots \xrightarrow{\mathtt{int}}_{\lambda s} C[c\,\sigma^i d_n] \xrightarrow{\mathtt{int}}_{\lambda s} \cdots$$

Now two possibilities arise from lemma 4:

- There exists $I \leq N$ such that $a_I = C'[(\lambda c')d'] \to a_{I+1} = C'[c'\sigma^1 d']$ and $d' \twoheadrightarrow d$. But let us consider the following derivation:

  $$\mathcal{D}''\ :\ a \twoheadrightarrow a_I = C'[(\lambda c')d'] \twoheadrightarrow C'[(\lambda c')d] \to C'[(\lambda c')d_1] \to \cdots \to C'[(\lambda c')d_n] \to \cdots$$

  In this infinite derivation the redex in a_I is within d' which is a proper subterm of $(\lambda c')d'$, whereas in $\mathcal{D}$ the redex in a_I is $(\lambda c')d'$ and hence it is placed at an upper position. This contradicts the minimality of $\mathcal{D}$.

- $a = C'[c'\,\sigma^i d']$ where $d' \twoheadrightarrow d$. But now we have $d' \twoheadrightarrow d \to d_1 \to \cdots \to d_n \to \cdots$. Since d' is a subterm of a, $d' \in S$, contradicting our choice of a with minimal length. □

Therefore we conclude, using propositions 1 and 2:

Theorem 6 *Every well typed λs-term is strongly normalising in the λs-calculus.*

3 The s_e- and λs_e-calculi

We want now to extend the syntax by admitting variables. This will lead us to a new problem of confluence. Let us begin by giving explicitly the new syntax:

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

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

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 *and* pure terms *are defined as for Λs.*

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

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

In fact one loses more: local confluence. 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 straightforward: add to λs the rules obtained by orienting the equalities given by the lemmas 1 - 6 in [KR95]. For instance, the rule corresponding to the Meta-substitution lemma (lemma 4) is the *σ-σ-transition* rule given below. The addition of this rule would solve the critical pair in our counterexample, since now we have $(X\sigma^1 Y)\sigma^1 \mathtt{1} \to (X\sigma^2 \mathtt{1})\sigma^1(Y\sigma^1 \mathtt{1})$.

Definition 11 *The* λs_e-calculus *is obtained by adding the rules in Figure 2 to the rules of the λs-calculus given in Figure 1.*

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

Figure 2: The new rules of the λs_e-calculus

We use λs_e to denote this set of rules. The calculus of substitutions associated with the λs_e-calculus is the rewriting system whose rules are $\lambda s_e - \{\sigma$-generation$\}$ and we call it s_e-calculus.

We prove the local confluence of the s_e and λs_e calculi by analysis of critical pairs.

Theorem 7 *The s_e-calculus is locally confluent.*

Proof: Local confluence can be obtained by the Knuth-Bendix Theorem (cf. [KB70], [Hue80]). Therefore we must check that every critical pair is convergent.

We shall only enumerate the superpositions generating the critical pairs, check the convergence of the first one and write the departing term for the other ones.

We remark that when the destruction rules must be handled, several subcases should be considered according to the relationship between the indexes of the operators and the de Bruijn numbers.

1. $\underline{\sigma\text{-}\sigma\text{-}tr.\text{ and }\sigma\text{-}\lambda\text{-}tr.}$:

$$((\lambda a)\,\sigma^i b)\,\sigma^j\,c \longrightarrow ((\lambda a)\,\sigma^{j+1}\,c)\;\sigma^i\;(b\,\sigma^{j-i+1}\,c) \longrightarrow (\lambda(a\,\sigma^{j+2}\,c))\;\sigma^i\;(b\,\sigma^{j-i+1}\,c)$$
$$\longrightarrow \lambda((a\,\sigma^{j+2}\,c)\;\sigma^{i+1}\;(b\,\sigma^{j-i+1}\,c))$$
$$((\lambda a)\,\sigma^i b)\,\sigma^j\,c \longrightarrow (\lambda(a\,\sigma^{i+1}\,b))\,\sigma^j\,c \longrightarrow \lambda((a\,\sigma^{i+1}\,b)\,\sigma^{j+1}\,c)$$
$$\longrightarrow \lambda((a\,\sigma^{j+2}\,c)\;\sigma^{i+1}\;(b\,\sigma^{j-i+1}\,c))$$

<u>*σ-σ-tr.* and:</u>
2. *σ-app-tr.* $(((a_1 a_2)\,\sigma^i b)\,\sigma^j\,c)$; 3. *σ-dest.* $((\mathtt{n}\,\sigma^i b)\,\sigma^j\,c)$; 4. *σ-σ-tr.* $(((a_1\,\sigma^h\,a_2)\,\sigma^i b)\,\sigma^j\,c)$;
5. *σ-φ-tr.1* $(((\varphi^h_k a)\,\sigma^i b)\,\sigma^j\,c)$; 6. *σ-φ-tr.2* $(((\varphi^h_k a)\,\sigma^i b)\,\sigma^j\,c)$; 7. *φ-σ-tr.* $(\varphi^h_k((a\,\sigma^i b)\,\sigma^j\,c))$.
<u>*φ-σ-tr.* and:</u>
8. *σ-λ-tr.* $(\varphi^i_k((\lambda a)\,\sigma^j\,b))$; 9. *σ-app-tr.* $(\varphi^i_k((a\,b)\,\sigma^j\,c))$; 10. *σ-dest.* $(\varphi^i_k(\mathtt{n}\,\sigma^j\,b))$;
11. *σ-φ-tr.1* $(\varphi^i_k((\varphi^h_l a)\,\sigma^j\,b))$; 12. *σ-φ-tr.2* $(\varphi^i_k((\varphi^h_l a)\,\sigma^j\,b))$.
<u>*φ-φ-tr.1* and:</u>
13. *φ-λ-tr.* $(\varphi^i_k \varphi^h_l(\lambda a))$; 14. *φ-app-tr.* $(\varphi^i_k \varphi^h_l(a\,b))$; 15. *φ-dest.* $(\varphi^i_k(\varphi^h_l \mathtt{n}))$;
16. *φ-φ-tr.1* $(\varphi^i_k(\varphi^h_l(\varphi^j_p a)))$; 17. *φ-φ-tr.2* $(\varphi^i_k(\varphi^h_l(\varphi^j_p a)))$.
<u>*φ-φ-tr.2* and:</u>
18. *φ-λ-tr.* $(\varphi^i_k \varphi^h_l(\lambda a))$; 19. *φ-app-tr.* $(\varphi^i_k \varphi^h_l(a\,b))$; 20. *φ-dest.* $(\varphi^i_k(\varphi^h_l \mathtt{n}))$;
21. *φ-φ-tr.1* $(\varphi^i_k(\varphi^h_l(\varphi^j_p a)))$; 22. *φ-φ-tr.2* $(\varphi^i_k(\varphi^h_l(\varphi^j_p a)))$. □

Theorem 8 *The λs_e-calculus is locally confluent.*

Proof: There are only two critical pairs arising from the introduction of the *σ-generation* rule which are also convergent:
1. *σ-gen.* and *σ-app-tr.* $(((\lambda a)b)\,\sigma^i c)$; 2. *σ-gen.* and *φ-app-tr.* $(\varphi^i_k((\lambda a)b))$. □

We have shown in [KR95] that the s-normal forms of the λs-terms are exactly the pure terms in Λ. We shall describe now the s_e-normal forms (s_e-nf) of the open terms. and use this description to establish the weak normalisation of the s_e-calculus.

Theorem 9 *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 $\varphi^i_k d$ with $j \le k$.*
- *$a = \varphi^i_k b$, where b is an s_e-nf of the form X, or $c\,\sigma^j d$ with $j > k+1$, or $\varphi^j_l c$ with $k < l$.*

Proof: Proceed by analising the structure of a. When a is an application or an abstraction there are no restrictions since there are no s_e-rules with applications or abstractions at the root. When $a = b\,\sigma^j c$ or $a = \varphi^i_k b$, the restrictions on b are necessary to avoid σ-redexes (rules whose name begin with σ) or φ-redexes (rules whose name begin with φ), respectively. □

There is a simple way to describe the s_e-nf's using item notation [KN95]. Let us just say here that with this notation we have $a\,b = (b\delta)a$, $\lambda a = (\lambda)a$, $a\,\sigma^i b = (b\,\sigma^i)a$ and $\varphi^i_k a = (\varphi^i_k)a$. the following nomenclature is used: $(b\delta)$, (λ), $(c\,\sigma^i)$, (φ^i_k) 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.*

A *normal σφ-segment* $\overline{s}$ is a sequence of σ- and φ-items such that every pair of adjacent items in $\overline{s}$ are of the form:

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

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

We can now describe the s_e-nf's in a syntactical simple way.

Theorem 10 *The s_e-nf's can be described by the following syntax:*

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

where $\overline{s}$ is a normal $\sigma\varphi$-segment whose bodies belong to NF.

Proof: It is easy to see that these are in fact normal forms since the conditions on the inidices of a normal $\sigma\varphi$-segment prevent the existence of redexes. To check that if a term is an s_e-nf then it is generated by this grammar, use Theorem 9. $\square$

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

Lemma 6 *If $a \in NF$ then $\varphi^i_k a$ has a normal form denoted $s'_e(\varphi^i_k a)$.*
Moreover, if $\varphi^i_k a \notin NF$ 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: Induction on a.

The delicate point is when $a = \varphi^j_l b$, $l + j \leq k$ and $\varphi^i_{k+1-j} b \notin NF$. In this case take $s'_e(\varphi^i_k a) = \varphi^j_l s'_e(\varphi^i_{k+1-j} b)$. To check that it really is a normal form, our additional hypothesis is useful. By inductive hypothesis, we know $N(b) = N(s'_e(\varphi^i_{k+1-j} b))$, and since $\varphi^j_l b \in NF$ we conclude that also $\varphi^j_l s'_e(\varphi^i_{k+1-j})b \in NF$.

An analogous argument should apply when $a = b\,\sigma^j c$, $k \geq j-1$ and $(\varphi^i_{k+1} b) \notin NF$. $\square$

Lemma 7 *If a, $b \in NF$ then $a\,\sigma^j b$ has a normal form denoted $s''_e(a\,\sigma^j b)$.*
Moreover, if $a\,\sigma^j b \notin NF$ and $a \neq \mathtt{j}$ then:

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

Again the additional hypotheses are useful to treat the cases $a = \varphi^j_l b$ and $a = b\,\sigma^j c$. Lemma 6 must be used when $a = \mathtt{j}$ (take $s''_e(a\,\sigma^j b) = s'_e(\varphi^j_0 b)$) and when $a = \varphi^i_k c$ with $i + k = j$ (take $s''_e(a\,\sigma^j b) = s'_e(\varphi^i_{k+1} c)\sigma^{k+1} s'_e(\varphi^i_0 b)$). $\square$

Theorem 11 *The s_e-calculus is weakly normalising on open terms, i.e. every open term a has an s_e-normal form denoted $s_e(a)$.*

Proof: By induction on a. Take $s_e(X) = X$, $s_e(\mathtt{n}) = \mathtt{n}$, $s_e(b\,c) = s_e(b)s_e(c)$ (which is a normal form because the s_e-rules have no left members of sort δ), $s_e(\lambda b) = \lambda s_e(b)$ (idem for sort λ), $s_e(b\,\sigma^i c) = s''_e(s_e(b)\,\sigma^i s_e(c))$ and $s_e(\varphi^i_k b) = s'_e(\varphi^i_k s_e(b))$. $\square$

The proofs of the lemmas and the theorem gives us a choice for a sure strategy to reach the normal forms: either *leftmost-innermost* or *rightmost-innermost* strategies will do the work.

Conclusion

We sumarize in this section the results obtained so far and state the open problems concerning the λs-calculus.

In [KR95] we have proved its power to simulate β-reductions, its soundness with respect to classical λ-calculus, its confluence and the preservation of strong normalisation.

In this paper we presented the simply typed λs-calculus and proved the strong normalisation of the well typed terms. We introduced an extended version, the λs_e-calculus, and proved its local confluence and its weak normalisation.

Two main problems are still open: the confluence of the λs_e-calculus and the strong normalisation of the s_e-calculus. We believe that an affirmative answer to the latter would yield a solution to the former using the interpretation technique. However the strong normalisation of the s_e-calculus seems to be a difficult problem (at least after a big effort from our side). We hope that this paper will challenge the rewrite community to solve it.

Work is now in progress to prove the confluence of the λs_e-calculus on open terms using the weak normalisation of the s_e-calculus. If we succeed, the λs-calculus would be the first λ-calculus with explicit substitutions in de Bruijn notation, as far as we know, enjoying preservation of strong normalisation and having a confluent extension on open terms. We believe that this is the case due to the fact that composition of substitutions (in the sense of the $\lambda\sigma$-calculi) is handleld indirectly in the λs_e-calculus in a very subtle way via the *σ-σ-transition* rule.

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

References

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

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

[CHL91] P.-L. Curien, T. Hardin, and J.-J. Lévy. Confluence properties of weak and strong calculi of explicit substitutions. Technical report, To appear in the JACM, 1991.

[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.

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

[KR95] 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.

[Mel95] P.-A. Melliès. Typed λ-calculi with explicit substitutions may not terminate. *Submitted to TCLA'95*, 1995.