J. Functional Programming **10** (4): 321–325, July 2000. Printed in the United Kingdom

The λs_e-calculus does not preserve strong normalisation

BRUNO GUILLAUME
LRI, Université Paris Sud, F-91405 Orsay, CEDEX, France

Abstract

Kamareddine, F., & Ríos (1997) conjecture that the λs_e-calculus preserves the strong normalisation of the λ-calculus. We prove here that this conjecture is false.

Capsule Review

Calculi with explicit substitutions are important in formal representations of abstract machines, and exploring the addition of metavariables to typed lambda-calculi. They remain however quite mysterious. After a lot of attempts to find a proof of strong normalisation for suitable typed version of these calculi, Mellies (1995) found a surprising counter-example: though all terms normalise, there are terms on which special reduction strategies are looping. A natural question is if such counter-example depends crucially on the current formulation of such calculi, or if, by suitable reformulations, we can get a strongly normalising system. The following note shows that it is not so easy to avoid the problem: a natural attempt of designing a strongly normalising calculus is shown to contain a looping term, similar to the one found by Mellies.

1 Introduction

The main challenge with calculi of explicit substitutions is to find a calculus which has both confluence on terms with metavariables and the Preservation of Strong Normalisation (PSN). The Melliès counter-example (Melliès, 1995) shows that reduction systems with full composition do not have the PSN property. This counter-example is valid in any system with full composition either with De Bruijn indices or named variables (Bloo, 1995).

New systems that use restricted composition have been proposed since then. λs_e is one of them. In this calculus, the composition rule (($\sigma\sigma$), cf. section 2) is constrained (*via* a condition on the indices), and thus avoids the Melliès counter-example.

We show that the PSN of λs_e (conjectured by Kamareddine, F., & Ríos, 1997) is false. We give a simply typed λ-term and an infinite derivation of this term in λs_e. The proof looks like the one of Melliès.

Zantema has proved that the ($\sigma\sigma$) rule terminates, but this is not enough to recover the PSN. To have a restricted ($\sigma\sigma$) rule seams to be the crucial point to keep strong normalisation, but this example shows that the statement *'update functions do not matter for termination issues'* (in Bloo and Geuvers, 1995) is not valid when

https://doi.org/10.1017/S0956796800003695 Published online by Cambridge University Press

(β)	$(\lambda a)\, b$	$\longrightarrow$	$a\,\sigma^1\, b$	
$(\sigma\lambda)$	$(\lambda a)\sigma^i b$	$\longrightarrow$	$\lambda(a\sigma^{i+1}b)$	
(σa)	$(a_1\, a_2)\sigma^i b$	$\longrightarrow$	$(a_1\sigma^i b)\,(a_2\sigma^i b)$	
(σn)	$\mathtt{n}\sigma^i b$	$\longrightarrow$	$\begin{cases} \mathtt{n}-\mathtt{1} & \text{if } n > i \\ \varphi_0^i\, b & \text{if } n = i \\ \mathtt{n} & \text{if } n < i \end{cases}$	
$(\varphi\lambda)$	$\varphi_k^i(\lambda a)$	$\longrightarrow$	$\lambda(\varphi_{k+1}^i\, a)$	
(φa)	$\varphi_k^i(a_1\, a_2)$	$\longrightarrow$	$(\varphi_k^i\, a_1)\,(\varphi_k^i\, a_2)$	
(φn)	$\varphi_k^i\, \mathtt{n}$	$\longrightarrow$	$\begin{cases} \mathtt{n}+\mathtt{i}-\mathtt{1} & \text{if } n > k \\ \mathtt{n} & \text{if } n \leqslant k \end{cases}$	
$(\sigma\sigma)$	$(a\sigma^i b)\sigma^j c$	$\longrightarrow$	$(a\sigma^{j+1}c)\sigma^i(b\sigma^{j-i+1}c)$	if $i \leqslant j$
$(\sigma\varphi_1)$	$(\varphi_k^i a)\sigma^j b$	$\longrightarrow$	$\varphi_k^{i-1} a$	if $k < j < k+i$
$(\sigma\varphi_2)$	$(\varphi_k^i a)\sigma^j b$	$\longrightarrow$	$\varphi_k^i(a\sigma^{j-i+1}b)$	if $k+i \leqslant j$
$(\varphi\sigma)$	$\varphi_k^i(a\sigma^j b)$	$\longrightarrow$	$(\varphi_{k+1}^i a)\sigma^j(\varphi_{k+1-j}^i b)$	if $j \leqslant k+1$
$(\varphi\varphi_1)$	$\varphi_k^i(\varphi_l^j a)$	$\longrightarrow$	$\varphi_l^j(\varphi_{k+1-j}^i a)$	if $l+j \leqslant k$
$(\varphi\varphi_2)$	$\varphi_k^i(\varphi_l^j a)$	$\longrightarrow$	$\varphi_l^{j+i-1} a$	if $l \leqslant k < l+j$

Fig. 1. Rules of the λs_e.

dealing with (even constrained) composition. Actually, in λs_e, interaction between substitutions (σ) and updating functions (φ) may generate infinite sequences of β-reduction. Therefore, much care must be taken when dealing with updating; and if we want to use named variables to do explicit substitutions, we have to say how the renaming is done without only using a *Barendregt convention*.

2 The λs_e-calculus

The terms of the λs_e-calculus are

$$\Lambda s_e ::= \mathbf{N} \mid \Lambda s_e \Lambda s_e \mid \lambda \Lambda s_e \mid \Lambda s_e \sigma^i \Lambda s_e \mid \varphi_k^i \Lambda s_e \qquad \text{where} \quad i \geqslant 1\,,\ \ k \geqslant 0\,.$$

We recall here the set of rules of the λs_e-calculus in figure 1.

3 The λs_e-calculus is not PSN

Theorem 3.1
The term $t = \lambda((\lambda((\lambda((\lambda 2)3))2))a)$ where $a = (\lambda((\lambda 2)2))1$ (*cf.* figure 2) is β strongly normalisable, but not λs_e strongly normalisable.

Remark 1
The term t is typable in $\lambda_{\rightarrow}$.

https://doi.org/10.1017/S0956796800003695 Published online by Cambridge University Press

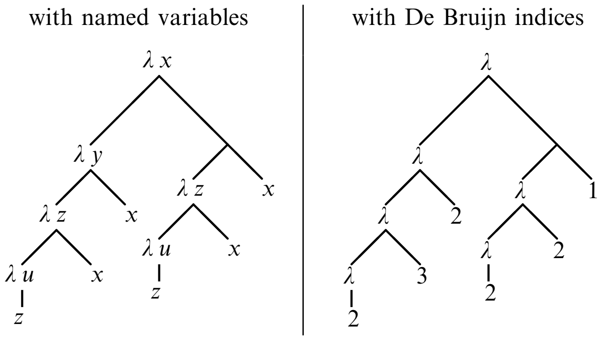

Fig. 2. The term t.

$$\begin{array}{rcl}
t & = & \lambda((\lambda((\lambda((\lambda 2)3))2))a) \\
 & \xrightarrow{\beta} & \lambda(((\lambda((\lambda 2)3))2)\sigma^1 a) \\
 & \xrightarrow{\beta} & \lambda((((\lambda 2)3)\sigma^1 2)\sigma^1 a) \\
 & \xrightarrow{\sigma a} & \lambda((((\lambda 2)\sigma^1 2)(3\sigma^1 2))\sigma^1 a) \\
 & \xrightarrow{\sigma\lambda} & \lambda(((\lambda(2\sigma^2 2))(3\sigma^1 2))\sigma^1 a) \\
 & \xrightarrow{\sigma a} & \lambda(((\lambda(2\sigma^2 2))\sigma^1 a)((3\sigma^1 2)\sigma^1 a)) \\
 & \xrightarrow{\sigma\lambda} & \lambda((\lambda((2\sigma^2 2)\sigma^2 a))((3\sigma^1 2)\sigma^1 a)) \\
 & = & \lambda((\lambda((2\sigma^2 2)\sigma^2 a))u_0) \\
 & \xrightarrow{\beta} & \lambda(((2\sigma^2 2)\sigma^2 a)\sigma^1 u_0) \\
 & \xrightarrow{\sigma n} & \lambda(((\varphi_0^2 2)\sigma^2 a)\sigma^1 u_0) \\
 & \xrightarrow{\sigma\varphi_2} & \lambda((\varphi_0^2(2\sigma^1 a))\sigma^1 u_0) \\
 & \xrightarrow{\varphi\sigma} & \lambda(((\varphi_1^2 2)\sigma^1(\varphi_0^2 a))\sigma^1 u_0) \\
 & \xrightarrow{\sigma\sigma} & \lambda(((\varphi_1^2 2)\sigma^2 u_0)\sigma^1((\varphi_0^2 a)\sigma^1 u_0)) \\
 & \supset & (\varphi_0^2 a)\sigma^1 u_0
\end{array}$$

Fig. 3. Proof of 3.2(i).

Notation 1

If b is a subterm of a, we write $a \supset b$. $a \succ b$ means that there is a term c with $a \longrightarrow^* c$ and $c \supset b$.

Lemma 3.2

We define: $\begin{cases} u_0 = (3\sigma^1 2)\sigma^1 a \\ u_{n+1} = ((\varphi_1^2 2)\sigma^1(\varphi_0^2 1))\sigma^1 u_n \;\; if \;\; n \geqslant 0 \end{cases}$

(i) $t \succ (\varphi_0^2 a)\sigma^1 u_0$

(ii) $(\varphi_0^2 a)\sigma^1 u_n \succ (\varphi_0^2 u_n)\sigma^1 u_{n+1}$ *for* $n \geqslant 0$

(iii) $(\varphi_0^2 u_0)\sigma^1 u_n \succ (\varphi_0^2 a)\sigma^1 u_n$ *for* $n \geqslant 1$

(iv) $(\varphi_0^2 u_m)\sigma^1 u_n \succ (\varphi_0^2 u_{m-1})\sigma^1 u_n$ *for* $n, m \geqslant 1$

Proof

The two first points of the lemma are proved in figures 3 and 4 (at each step, the redex which is reduced is underligned):

https://doi.org/10.1017/S0956796800003695 Published online by Cambridge University Press

$$
\begin{array}{rcl}
(\varphi_0^2 a)\sigma^1 u_n & = & (\varphi_0^2((\lambda((\lambda 2)2))1))\sigma^1 u_n \\
& \xrightarrow{\varphi a} & ((\varphi_0^2(\lambda((\lambda 2)2)))(\varphi_0^2 1))\sigma^1 u_n \\
& \xrightarrow{\varphi\lambda} & ((\lambda(\varphi_1^2((\lambda 2)2)))(\varphi_0^2 1))\sigma^1 u_n \\
& \xrightarrow{\varphi a} & ((\lambda((\varphi_1^2(\lambda 2))(\varphi_1^2 2)))(\varphi_0^2 1))\sigma^1 u_n \\
& \xrightarrow{\varphi\lambda} & ((\lambda((\lambda(\varphi_2^2 2))(\varphi_1^2 2)))(\varphi_0^2 1))\sigma^1 u_n \\
& \xrightarrow{\varphi n} & ((\lambda((\lambda 2)(\varphi_1^2 2)))(\varphi_0^2 1))\sigma^1 u_n \\
& \xrightarrow{\beta} & (((\lambda 2)(\varphi_1^2 2))\sigma^1(\varphi_0^2 1))\sigma^1 u_n \\
& \xrightarrow{\sigma a} & (((\lambda 2)\sigma^1(\varphi_0^2 1))((\varphi_1^2 2)\sigma^1(\varphi_0^2 1)))\sigma^1 u_n \\
& \xrightarrow{\sigma\lambda} & ((\lambda(2\sigma^2(\varphi_0^2 1)))((\varphi_1^2 2)\sigma^1(\varphi_0^2 1)))\sigma^1 u_n \\
& \xrightarrow{\sigma a} & ((\lambda(2\sigma^2(\varphi_0^2 1)))\sigma^1 u_n)(((\varphi_1^2 2)\sigma^1(\varphi_0^2 1))\sigma^1 u_n) \\
& \xrightarrow{\sigma\lambda} & (\lambda((2\sigma^2(\varphi_0^2 1))\sigma^2 u_n))(((\varphi_1^2 2)\sigma^1(\varphi_0^2 1))\sigma^1 u_n) \\
& = & (\lambda((2\sigma^2(\varphi_0^2 1))\sigma^2 u_n))u_{n+1} \\
& \xrightarrow{\beta} & ((2\sigma^2(\varphi_0^2 1))\sigma^2 u_n)\sigma^1 u_{n+1} \\
& \xrightarrow{\sigma n} & ((\varphi_0^2(\varphi_0^2 1))\sigma^2 u_n)\sigma^1 u_{n+1} \\
& \xrightarrow{\sigma\varphi_2} & (\varphi_0^2((\varphi_0^2 1)\sigma^1 u_n))\sigma^1 u_{n+1} \\
& \xrightarrow{\varphi\sigma} & ((\varphi_1^2(\varphi_0^2 1))\sigma^1(\varphi_0^2 u_n))\sigma^1 u_{n+1} \\
& \xrightarrow{\sigma\sigma} & ((\varphi_1^2(\varphi_0^2 1))\sigma^2 u_{n+1})\sigma^1((\varphi_0^2 u_n)\sigma^1 u_{n+1}) \\
& \supset & (\varphi_0^2 u_n)\sigma^1 u_{n+1}
\end{array}
$$

Fig. 4. Proof of 3.2(ii).

The third part is proved by (for $n \geqslant 1$):

$$
\begin{array}{rcl}
(\varphi_0^2 u_0)\sigma^1 u_n & = & (\varphi_0^2((3\sigma^1 2)\sigma^1 a))\sigma^1 u_n \\
& \xrightarrow{\varphi\sigma} & ((\varphi_1^2(3\sigma^1 2))\sigma^1(\varphi_0^2 a))\sigma^1 u_n \\
& \xrightarrow{\sigma\sigma} & ((\varphi_1^2(3\sigma^1 2))\sigma^2 u_n)\sigma^1((\varphi_0^2 a)\sigma^1 u_n) \\
& \supset & (\varphi_0^2 a)\sigma^1 u_n
\end{array}
$$

The last assertion of the lemma is proved by the derivation (for $n, m \geqslant 1$):

$$
\begin{array}{rcl}
(\varphi_0^2 u_m)\sigma^1 u_n & = & (\varphi_0^2(((\varphi_1^2 2)\sigma^1(\varphi_0^2 1))\sigma^1 u_{m-1}))\sigma^1 u_n \\
& \xrightarrow{\varphi\sigma} & ((\varphi_1^2((\varphi_1^2 2)\sigma^1(\varphi_0^2 1)))\sigma^1(\varphi_0^2 u_{m-1}))\sigma^1 u_n \\
& \xrightarrow{\sigma\sigma} & ((\varphi_1^2((\varphi_1^2 2)\sigma^1(\varphi_0^2 1)))\sigma^2 u_n)\sigma^1((\varphi_0^2 u_{m-1})\sigma^1 u_n) \\
& \supset & (\varphi_0^2 u_{m-1})\sigma^1 u_n
\end{array}
$$

□

Lemma 3.3

For all $n \geqslant 0, (\varphi_0^2 a)\sigma^1 u_n \succ (\varphi_0^2 a)\sigma^1 u_{n+1}$.

Proof

$(\varphi_0^2 a)\sigma^1 u_n \succ (\varphi_0^2 u_n)\sigma^1 u_{n+1}$ (Lemma 3.2(ii)).

$(\varphi_0^2 u_n)\sigma^1 u_{n+1} \succ (\varphi_0^2 u_{n-1})\sigma^1 u_{n+1} \succ \cdots \succ (\varphi_0^2 u_0)\sigma^1 u_{n+1}$ (Lemma 3.2(iv)).

$(\varphi_0^2 u_0)\sigma^1 u_{n+1} \succ (\varphi_0^2 a)\sigma^1 u_{n+1}$ (Lemma 3.2(iii)). □

Proof of Theorem 3.1 $t \succ (\varphi_0^2 a)\sigma^1 u_0$ (by lemma 3.2.i) and the lemma 3.3 gives an infinite λs_e derivation of $(\varphi_0^2 a)\sigma^1 u_0$.

https://doi.org/10.1017/S0956796800003695 Published online by Cambridge University Press

References

Bloo, R. (1995) *Preservation of strong normalisation for explicit substitution.* PhD thesis, Eindhoven University of Technology.

Bloo, R. and Geuvers, H. (1995) Explicit substitution: on the edge of strong normalisation. *Technical report 95-08*, Department of Mathematics and Computing Science, Eindhoven University of Technology.

Kamareddine, F. and Ríos, A. (1997) Extending a λ-calculus with explicit substitution which preserves strong normalisation into a confluent calculus on open terms. *J. Functional Programming*, **7**(4), 395–420.

Melliès, P.-A. (1995) Typed λ-calculi with explicit substitutions may not terminate. *Proceedings of TLCA'95: Lecture Notes in Computer Science*, **902**, 328–334.