Explicit Substitution: on the Edge of Strong Normalization

Roel Bloo* bloo@win.tue.nl

Herman Geuvers herman@win.tue.nl

Abstract

We use the Recursive Path Ordering (RPO) technique of semantic labelling to show the *Preservation of Strong Normalization* (PSN) property for several calculi of explicit substitution. Preservation of Strong Normalization states that if a term M is strongly normalizing under ordinary β-reduction (using 'global' substitutions), then it is strongly normalizing if the substitution is made explicit ('local'). There are different ways of making global substitution explicit and PSN is a quite natural and desirable property for the explicit substitution calculus. Our method for proving PSN is very general and applies to several known systems of explicit substitutions, both with named variables and with De Bruijn indices: λv of Lescanne et al., λs of Kamareddine and Ríos and $\lambda\mathrm{x}$ of Rose and Bloo. We also look at two small extensions of the explicit substitution calculus that allow to permute substitutions. For one of these extensions PSN fails (using the counterexample in [Melliès 95]). For the other we can prove PSN using our method, thus showing the subtlety of the subject and the generality of our method.

One of the key ideas behind our proof is that, for $\lambda\mathrm{x}$ the set of terms of the explicit substitution calculus, we look at the set $\lambda\mathrm{x}^{<\infty}$, consisting of the terms A such that the substitution-normal-form of each subterm of A is β-SN. This is a kind of 'induction loading': if we prove that $\lambda\mathrm{x}$-reduction is SN on the set $\lambda\mathrm{x}^{<\infty}$, then we have proved PSN for $\lambda\mathrm{x}$. To prove $\lambda\mathrm{x}$-SN on the set $\lambda\mathrm{x}^{<\infty}$, we define the *$\beta$-size* of a term $A \in \lambda\mathrm{x}^{<\infty}$ as the maximum length of a β-reduction path from the substitution-normal-form of A. Using this β-size, we define a translation from $\lambda\mathrm{x}^{<\infty}$ to some well-founded order $>_{\mathrm{rpo}}$ on labelled terms, such that any infinite $\lambda\mathrm{x}$-reduction path starting from an $A \in \lambda\mathrm{x}^{<\infty}$ translates to an infinite $>_{\mathrm{rpo}}$-descending sequence. The well-founded order $>_{\mathrm{rpo}}$ is defined by using the technique of semantic labelling.

Keywords: lambda-calculus, explicit substitution, recursive path order.

1 Introduction

Explicit Substitution was first studied by Abadi, Cardelli, Curien and Lévy in [Abadi et al. 90]. They proposed a calculus $\lambda\sigma$ of explicit substitutions which can compose substitutions. Melliès has shown that simply typable terms can have infinite reduction paths in $\lambda\sigma$ ([Melliès 95]). Several people (see [BBLR 95], [Bloo & Rose 95], [Bloo 95], [Kamareddine & Rios 95], [Munoz 96]) have succeeded in defining calculi of explicit substitutions which have the nice property that every term which is strongly normalizing for β-reduction is also strongly normalizing in the explicit substitution calculus. We call this property: PSN (Preservation of Strong Normalization).

In this paper we present a method to prove PSN for explicit substitution calculi based on the recursive path order. In contrast to the work of Ferreira, Kesner and Puel (cf. [FKP 97]), our method is applicable to named calculi as well as to calculi based on De Bruijn indices. Furthermore, it yields direct proofs of PSN instead of reducing PSN for a new calculus to PSN for an old calculus. Zantema used semantic labelling and the recursive path order to show termination of

*address of both authors: Faculty of Mathematics and Computing Sience, Eindhoven University of Technology, P.O.Box 513, NL–5600 MB Eindhoven. The first author was supported by the Netherlands Computer Science Research Foundation (SION) with financial support from the Netherlands Organization for Scientific Research (NWO).

the substitution part of $\lambda\sigma$ [Zantema 95], but the technique he used doesn't apply to show PSN. We use a stronger technique called semantic labelling [Ferreira & Zantema 95] to show PSN for all explicit substitution calculi known to have the PSN property. We also show why our method doesn't work for $\lambda\sigma$. Our technique relies on introducing a first order term rewrite system where function symbols for application and substitution are labelled with natural numbers and where variables are represented by just one constant $*$. The recursive path order $>_{\mathrm{rpo}}$ on this labelled calculus is strongly normalizing (or: terminating).

Then we take a look at the explicit substitution calculus $\lambda\mathrm{x}$. Here the β-reduction is split up into a reduction $\rightarrow_{\mathrm{Beta}}$ (contracting the β-redex and creating an explicit substitution) and a reduction $\rightarrow_{\mathrm{x}}$ (moving the explicit substitutions through the term to perform the substitution). It is relatively easy (as usual in these calculi) to observe that $\rightarrow_{\mathrm{x}}$ is strongly normalizing and confluent. So, for terms A of $\lambda\mathrm{x}$, the $\rightarrow_{\mathrm{x}}$-normal form (*substitution normal form*) exists and is unique; we call it $\mathrm{x}(A)$.

Now—and this is a crucial point in the proof of PSN—we take a look at the terms in $\lambda\mathrm{x}$ for which the substitution normal form of all of its subterms is β-SN; we call this set $\lambda\mathrm{x}^{<\infty}$. An important fact to note is that all β-SN pure λ-terms are elements of $\lambda\mathrm{x}^{<\infty}$. For $A \in \lambda\mathrm{x}^{<\infty}$, we define the *$\beta$-size of A, $\hat{\beta}(A)$,* as the *maximum length of all paths from $\mathrm{x}(A)$ to its β-normal form.* Using this β-size, we then define a translation $\mathcal{T}$ from $\lambda\mathrm{x}^{<\infty}$ into the previously mentioned first order term rewriting system with labelled terms. This translation $\mathcal{T}$ is reduction preserving in the sense that, if $M \rightarrow_{\lambda\mathrm{x}} N$, then $\mathcal{T}(M) >_{\mathrm{rpo}} \mathcal{T}(N)$. Hence, using the fact that $>_{\mathrm{rpo}}$ is well-founded, we conclude that every $M \in \lambda\mathrm{x}^{<\infty}$ is $\lambda\mathrm{x}$-strongly normalizing. So, $\lambda\mathrm{x}$ has the PSN property, because every λ-term that is β-strongly-normalizing is an element of $\lambda\mathrm{x}^{<\infty}$.

For those more familiar with the RPO technique in the way it has been presented in [Klop 92], we also present, in the final section, a translation T from $\lambda\mathrm{x}^{<\infty}$ to commutative labelled trees. This translation is also reduction preserving in the following way (slightly different from the situation for $\mathcal{T}$). If $M \rightarrow_{\mathrm{x}} N$, then $\mathcal{T}(M) \Rrightarrow^{*} \mathcal{T}(N)$ and if $M \rightarrow_{\mathrm{Beta}} N$, then $\mathcal{T}(M) \Rrightarrow^{+} \mathcal{T}(N)$. Here, $\Rrightarrow$ is the rpo-reduction on commutative labelled trees, as defined in [Klop 92], and $\Rrightarrow^{+}$ and $\Rrightarrow^{*}$ are, respectively, its transitive and transitive reflexive closure. Now, PSN is obtained from the fact that $\Rrightarrow$ and $\rightarrow_{\mathrm{x}}$ are strongly normalizing.

To show the flexibility of our proof method we use it for different calculi of explicit substitution. We start off with a calculus with named variables (different from, e.g. [Abadi et al. 90], where De Bruijn-indices are used). We have chosen to use named variables because this makes the presentation better accessible for non-specialists. Moreover, it makes it easier to single out the places where the difficulties arise in the calculus of [Abadi et al. 90]. Hence, it helps clarifying the problem of PSN. It should be remarked that it is not always straightforward how to turn a calculus without named variables into a calculus with names, e.g. for $\lambda\sigma$ this is complicated because of the complex notion of scope. We also apply our proof method to the calculi $\lambda\upsilon$ of Lescanne et al. and λs of Kamareddine and Ríos. A well-known source of failure of PSN is the permutation of substitutions (to a specific extent). In section 5.2 we discuss two small extensions of $\lambda\mathrm{x}$ in which permutation of substitutions is allowed under some very restricted conditions. For one of these extensions, PSN can be proved using our method. For the other extension, PSN fails. It seems that the border between PSN and non-PSN lies between these two systems.

2 A calculus for explicit substitutions with named variables

In the standard definition of the untyped lambda calculus, substitution is a meta-operation, usually denoted by $[x{:=}N]$ or $[N/x]$, where x is a variable and N a term. In the following we use the notation $[N/x]$ for a (global) substitution of N for x. For M and N terms and x, y distinct variables, the term $M[N/x]$ is then defined by structural induction as follows.

$$\begin{aligned} x[N/x] &:= N, \\ y[N/x] &:= y, \text{ if } y \not\equiv x, \\ (PQ)[N/x] &:= P[N/x]Q[N/x], \end{aligned}$$

$$\begin{aligned}
(\lambda y.P)[N/x] &:= \lambda y'.P[y'/y][N/x], \text{ if } y' \notin FV(N) \cup \{x\} \cup (FV(P) \setminus \{y\}) \\
(\lambda x.P)[N/x] &:= \lambda x.P.
\end{aligned}$$

We assume the notions of *free variable* (FV) and *bound variable* (BV) to be known. Furthermore, $\equiv$ denotes syntactical equality modulo α-conversion, which is defined as the smallest equivalence relation such that

$$\begin{aligned}
& & x \equiv x, \\
P \equiv N \text{ and } Q \equiv M &\Rightarrow & PQ \equiv NM, \\
P \equiv Q, y \notin FV(Q) \setminus \{x\} &\Rightarrow & \lambda x.P \equiv \lambda y.Q[y/x].
\end{aligned}$$

In the definition of substitution, there is a choice for the variable y'. For this definition to make sense, it has to be shown that the specific choice for the variable y' is irrelevant. But this is a consequence of the definition of $\equiv$ and the following Lemma.

Lemma 2.1 *If $P \equiv Q$ and $M \equiv N$, then $P[M/x] \equiv Q[N/x]$.*

Remark: It is possible to first define α-conversion and then define substitution modulo α-conversion. However, in that case, substitution of variables for variables has to be defined first (before α-conversion), therefore we define it the slightly shorter way, as above.

In order to get a calculus $\lambda\mathrm{x}$ of explicit substitutions, two extensions have to be made. The first is extending the terms with substitutions:

Definition 2.2 *The set of terms $\lambda\mathrm{x}$ is defined by the following abstract syntax:*

$$A ::= x \mid AA \mid \lambda x.A \mid A\langle x{:=}A\rangle$$

Where x denotes an arbitrary variable.

Substitution is defined on $\lambda\mathrm{x}$-terms as for λ-terms but with the extra clauses that

$$M\langle y{:=}P\rangle[N/x] := M[y'/y][N/x]\langle y'{:=}P[N/x]\rangle \text{ if } y' \notin FV(N) \cup \{x\} \cup (FV(M) \setminus \{y\})$$

$$M\langle x{:=}P\rangle[N/x] := M\langle x{:=}P[N/x]\rangle$$

α-equivalence is defined on $\lambda\mathrm{x}$-terms as for λ-terms but with the extra clause that

$$M \equiv N, P \equiv Q, y \notin FV(Q) \setminus \{x\} \Rightarrow P\langle x{:=}M\rangle \equiv Q[y/x]\langle y{:=}N\rangle$$

$A \in \lambda\mathrm{x}$ is called pure *if $A \in \lambda$, i.e., A does not contain any substitution $\langle x{:=}B\rangle$.*

The second is refining the notion of β-reduction. Remember that the reduction relation $\rightarrow_\beta$ on pure terms is defined as the contextual closure of

$$(\lambda x.A)B \rightarrow_\beta A[B/x]$$

We make explicit the global substitution in $\rightarrow_\beta$ by splitting $\rightarrow_\beta$ into two parts. $\rightarrow_{\mathrm{Beta}}$ is for the creation of a substitution out of a β-redex; $\rightarrow_{\mathrm{x}}$ is for the proliferation of substitutions through a term to variables and for performing the actual substitution or throwing away the substitute if the substitution turns out to be void.

Definition 2.3 *The reduction relations $\rightarrow_{\mathrm{Beta}}$ and $\rightarrow_{\mathrm{x}}$ are defined to be the contextual closures modulo α-conversion the following rules (respectively)*

$$(\lambda x.A)B \rightarrow_{\mathrm{Beta}} A\langle x{:=}B\rangle$$

$$\begin{aligned}
(AB)\langle x{:=}C\rangle &\rightarrow_{\mathrm{x}} (A\langle x{:=}C\rangle)(B\langle x{:=}C\rangle) \\
(\lambda y.A)\langle x{:=}C\rangle &\rightarrow_{\mathrm{x}} \lambda y.A\langle x{:=}C\rangle \text{ if } x \not\equiv y \text{ and } y \notin FV(C) \\
x\langle x{:=}C\rangle &\rightarrow_{\mathrm{x}} C \\
A\langle x{:=}C\rangle &\rightarrow_{\mathrm{x}} A \text{ if } x \notin FV(A)
\end{aligned}$$

The explicit substitution reduction relation $\rightarrow_{\lambda\mathrm{x}}$ is the union of $\rightarrow_{\mathrm{Beta}}$ and $\rightarrow_{\mathrm{x}}$.

The reduction $A\langle x{:=}C\rangle \to_{\mathrm{x}} A$ if $x \notin FV(A)$ is also called garbage collection. Since we consider terms modulo α-equality, substitutions can always be distributed to variables, hence the rule $y\langle x{:=}C\rangle \to_{\mathrm{x}} y$ if $x \not\equiv y$ would already be sufficient. The more efficient garbage collection will do no harm however.

Remark: Working modulo α-conversion is no problem, because all operations that we define on $\lambda\mathrm{x}$ are modulo α-conversion (as usual for a calculus with named variables). We shall not mention this point anymore in the sequel.

The reduction relation $\to_{\mathrm{x}}$ is called the substitution calculus. It has nice properties:

Lemma 2.4 *The reduction $\to_{\mathrm{x}}$ is strongly normalizing, confluent and has unique normal forms.*

Proof: Strong normalization is shown by defining a map $h : \lambda\mathrm{x} \to I\!N$ which decreases on x-reduction; define

$$\begin{array}{rcl} h(x) & = & 1 \\ h(AB) & = & h(A) + h(B) + 1 \\ h(\lambda x.A) & = & h(A) + 1 \\ h(A\langle x{:=}B\rangle) & = & h(A) \cdot (h(B) + 1) \end{array}$$

then by induction on the structure of A: if $A \to_{\mathrm{x}} B$ then $h(A) > h(B)$.

To prove confluence, it is now sufficient to show weak confluence which is easy. □

Notation 2.5 *For R a reduction relation, we write $A \in SN_R$ if A is strongly normalizing with respect to R.*

Definition 2.6 *Let A be an element of $\lambda\mathrm{x}$.*

1. *If A is a pure term, we write $\beta(A)$ to denote the β-normal form of A, if it exists.*
2. *We write $\mathrm{x}(A)$ to denote the x-normal form of A.*
3. *The β-size of A, $\hat{\beta}(A)$, is defined as the maximal length of a β-reduction path starting from $\mathrm{x}(A)$, if $\mathrm{x}(A) \in SN_\beta$. If $\mathrm{x}(A) \notin SN_\beta$, we let $\hat{\beta}(A) := \infty$.*

Note that for $A \in \lambda\mathrm{x}$, $\mathrm{x}(A)$ is pure.

We now give some elementary but important properties of x and $\hat{\beta}$.

Lemma 2.7 (substitution) *For all terms A, B: $\mathrm{x}(A\langle x{:=}B\rangle) \equiv \mathrm{x}(A)[\mathrm{x}(B)/x]$.*

Proof: It is enough to prove the following property.

$$\mathrm{x}(A\langle x_1{:=}B_1\rangle \cdots \langle x_m{:=}B_m\rangle) \equiv \mathrm{x}(A)[\mathrm{x}(B_1)/x_1] \cdots [\mathrm{x}(B_m)/x_m],$$

for all terms $B_1, \ldots, B_m$ and all terms A that do not end with a substitution. (So, one takes A 'as small as possible', i.e. A is a variable, an application or an abstraction). This property is easily proved by induction on the number of symbols in the sequence $A, B_1, \ldots, B_m$. $\equiv \stackrel{IH}{\equiv}$ $\mathrm{x}(A\langle x_1{:=}B_1\rangle \cdots \langle x_m{:=}B_m\rangle) \equiv$ □

Lemma 2.8 (Projection) *For all terms A, B:*

1. *if $A \to_{\mathrm{x}} B$ then $\mathrm{x}(A) \equiv \mathrm{x}(B)$*
2. *if $A \to_{\mathrm{Beta}} B$ then $\mathrm{x}(A) \twoheadrightarrow_\beta \mathrm{x}(B)$*

Proof: The first is an immediate consequence of Lemma 2.4 and the second is by induction on the structure of A. Note that if $N \twoheadrightarrow_\beta N'$ then $M[N/x] \twoheadrightarrow_\beta M[N'/x]$. We treat two cases:

- $A \equiv (\lambda x.A_1)A_2$, $B \equiv A_1\langle x{:=}A_2\rangle$. Then $\mathrm{x}(A) \equiv (\lambda x.\mathrm{x}(A_1))\mathrm{x}(A_2) \to_\beta \mathrm{x}(A_1)[\mathrm{x}(A_2)/x] \stackrel{2.7}{\equiv} \mathrm{x}(A_1\langle x{:=}A_2\rangle) \equiv \mathrm{x}(B)$.
- $A \equiv A_1\langle x{:=}A_2\rangle$, $B \equiv A_1\langle x{:=}A_2'\rangle$. Then $\mathrm{x}(A) \stackrel{2.7}{\equiv} \mathrm{x}(A_1)[\mathrm{x}(A_2)/x] \stackrel{IH}{\twoheadrightarrow}_\beta \mathrm{x}(A_1)[\mathrm{x}(A_2')/x] \stackrel{2.7}{\equiv} \mathrm{x}(B)$.

□

Note: The projection lemma is not strong enough to give us PSN. The problem is that if $A \to_{\mathrm{Beta}} B$ then sometimes $\mathrm{x}(A) \equiv \mathrm{x}(B)$, as in $x\langle y{:=}(\lambda z.C)D\rangle \to_{\mathrm{Beta}} x\langle y{:=}C\langle z{:=}D\rangle\rangle$. A proof of PSN by analyzing what can happen inside 'void' substitutions such as in this example is given in [Bloo 95] and in [Bloo & Rose 95].

Lemma 2.9 (Soundness) *For all pure terms A, B, if $A \to_\beta B$ then $A \twoheadrightarrow_{\lambda\mathrm{x}} B$.*

Proof: Induction on the structure of A, using Lemma 2.7 (substitution). We treat the case $A \equiv (\lambda x.A_1)A_2$, $B \equiv A_1[A_2/x]$. Then

$$A \to_{\mathrm{Beta}} A_1\langle x{:=}A_2\rangle \twoheadrightarrow_{\mathrm{x}} \mathrm{x}(A_1\langle x{:=}A_2\rangle) \stackrel{\text{Lemma 2.7}}{\equiv} \mathrm{x}(A_1)[\mathrm{x}(A_2)/x] \stackrel{A\text{ pure}}{\equiv} A_1[A_2/x].$$

□

A final property of $\lambda\mathrm{x}$ that can be shown easily is the confluence of $\twoheadrightarrow_{\lambda\mathrm{x}}$.

Theorem 2.10 (Confluence) *The reduction relation $\twoheadrightarrow_{\lambda\mathrm{x}}$ is confluent on $\lambda\mathrm{x}$.*

Proof: Let A, B_1, B_2 be $\lambda\mathrm{x}$-terms such that $A \twoheadrightarrow_{\lambda\mathrm{x}} B_1$ and $A \twoheadrightarrow_{\lambda\mathrm{x}} B_2$. Then by projection (Lemma 2.8) $\mathrm{x}(A) \twoheadrightarrow_\beta \mathrm{x}(B_i)$ $(i = 1, 2)$, so by confluence of $\twoheadrightarrow_\beta$ there is a pure term C such that $\mathrm{x}(B_i) \twoheadrightarrow_\beta C$ $(i = 1, 2)$. We also have $B_i \twoheadrightarrow_{\mathrm{x}} \mathrm{x}(B_i)$ (by definition of x) and $\mathrm{x}(B_i) \twoheadrightarrow_{\lambda\mathrm{x}} C$ by soundness (Lemma 2.9). So we conclude that $B_i \twoheadrightarrow_{\lambda\mathrm{x}} C$ $(i = 1, 2)$, so C is a common reduct of B_1 and B_2. □

3 The recursive path order

In this section we briefly introduce the recursive path order. For a more detailed description and proofs, the reader is referred to [Dershowitz 79], [Zantema 95] and [Ferreira & Zantema 95].

Definition 3.1 *Let $\mathcal{F}$ be a set of function symbols, $\mathcal{X}$ a set of variables such that $\mathcal{F} \cap \mathcal{X} = \emptyset$, let $T(\mathcal{F}, \mathcal{X})$ be the set of (open) terms over $\mathcal{F}$ and $\mathcal{X}$. Let $\rhd$ be a partial order on $\mathcal{F}$. Let τ be a map assigning to every function symbol $f \in \mathcal{F}$ one of the words* mult *or* lex .

The recursive path order $>_{\mathrm{rpo}}$ on $T(\mathcal{F}, \mathcal{X})$ induced by $\rhd$ and τ is defined by

$$\begin{array}{lll} f(s_1, \ldots, s_m) & >_{\mathrm{rpo}} & g(t_1, \ldots, t_n) \\ & \textit{iff} & \exists i[s_i = g(t_1, \ldots, t_n) \vee s_i >_{\mathrm{rpo}} g(t_1, \ldots, t_n)] \\ & & \vee \quad (f \rhd g \wedge \forall j[f(s_1, \ldots, s_m) >_{\mathrm{rpo}} t_j]) \\ & & \vee \quad (f = g \wedge \forall j[f(s_1, \ldots, s_m) >_{\mathrm{rpo}} t_j] \wedge \langle s_1, \ldots, s_m\rangle >_{\mathrm{rpo}}^{\tau(f)} \langle t_1, \ldots, t_n\rangle) \end{array}$$

Here $>_{\mathrm{rpo}}^{\mathrm{lex}}$ and $>_{\mathrm{rpo}}^{\mathrm{mult}}$ are respectively the lexicographic and the multiset extensions of $>_{\mathrm{rpo}}$, i.e.,

- *$\langle s_1, \ldots, s_m\rangle >_{\mathrm{rpo}}^{\mathrm{lex}} \langle t_1, \ldots, t_n\rangle$ iff for some $i \le m, n$, $s_1 = t_1, \ldots, s_{i-1} = t_{i-1}, s_i >_{\mathrm{rpo}} t_i$ or $s_1 = t_1, \ldots, s_m = t_m, m > n$.*
- *$\langle s_1, \ldots, s_m\rangle >_{\mathrm{rpo}}^{\mathrm{mult}} \langle t_1, \ldots, t_n\rangle$ iff the multiset $\{\{s_1, \ldots, s_m\}\}$ can be transformed into the multiset $\{\{t_1, \ldots, t_n\}\}$ by performing the operation* 'replace a member s of the multiset by finitely many terms t such that $s >_{\mathrm{rpo}} t$' *one or more times.*

In [Ferreira & Zantema 95], τ is called status function. More complex extensions of $>_{\mathrm{rpo}}$ than multiset or lexicographic are even possible.

Theorem 3.2 (Dershowitz) *Let $\rhd$ be a partial order and τ a status function on a set of function symbols $\mathcal{F}$, let $>_{\mathrm{rpo}}$ be the induced recursive path order. Then*

$$>_{\mathrm{rpo}} \textit{ is well-founded} \quad \Longleftrightarrow \quad \rhd \textit{ is well-founded}$$

Proof: see [Dershowitz 79] or [Ferreira & Zantema 95]. □

4 PSN for λx

In this section we use the recursive path order to show that λx has PSN. Since the recursive path order is about first order term rewrite systems, we need to translate terms of λx into a first order term rewrite system. (Due to the presence of variable binding, the system λx is not first order.) To be able to prove PSN this translation must in some sense preserve reductions. We do this by labelling (some) function symbols with maximal lengths of reduction sequences; therefore we restrict to terms where these lengths are finite for all subterms. It will turn out that these are exactly all the strongly normalizing λx-terms.

Definition 4.1 *We define the set $\lambda\mathrm{x}^{<\infty} \subset \lambda\mathrm{x}$ by*

$$\lambda\mathrm{x}^{<\infty} = \{A \in \lambda\mathrm{x} \mid \textit{for all subterms } B \textit{ of } A, \mathrm{x}(B) \in SN_\beta\}$$

Remark: For $A \in \lambda\mathrm{x}$, $A \in \lambda\mathrm{x}^{<\infty}$ if and only if: for all subterms B of A, $\hat{\beta}(B) < \infty$.

Lemma 4.2 *For $(\lambda x.A)B \in \lambda\mathrm{x}^{<\infty}$, $\hat{\beta}((\lambda x.A)B) > \hat{\beta}(A\langle x{:=}B\rangle)$.*

Proof: First note that $\mathrm{x}((\lambda x.A)B) \equiv (\lambda x.\mathrm{x}(A))\mathrm{x}(B)$ and $\mathrm{x}(A\langle x{:=}B\rangle) \equiv \mathrm{x}(A)[\mathrm{x}(B)/x]$. Now, every $\to_\beta$-reduction path of length n starting from $\mathrm{x}(A\langle x{:=}B\rangle)$ can be extended to a reduction path of length $n+1$ starting from $\mathrm{x}((\lambda x.A)B)$, by prefixing it with $(\lambda x.\mathrm{x}(A))\mathrm{x}(B) \to_\beta \mathrm{x}(A)[\mathrm{x}(B)/x]$. □

Lemma 4.3 *If $A \in \lambda\mathrm{x}^{<\infty}$ and $A \to_{\lambda\mathrm{x}} A'$ then $A' \in \lambda\mathrm{x}^{<\infty}$.*

Proof: Induction on the structure of A, using Lemma 2.8. □
Note: Lemma 4.3 is the crucial Lemma that does not hold for $\lambda\sigma$.

Definition 4.4 *The TRS λ^l, with λ^l as set of terms and reduction relation $\to_l$ is defined as follows. The set of labelled terms λ^l is defined by the following abstract syntax:*

$$A ::= * \mid A \cdot_n A \mid \lambda A \mid A\langle A\rangle_n$$

where n ranges over $\mathbb{N}$. The reduction relation $\to_l$ is defined by

$$\begin{array}{rcll}
(\lambda A) \cdot_m B & \to_l & A\langle B\rangle_n & \textit{if } m > n \\
(A \cdot_m B)\langle C\rangle_n & \to_l & (A\langle C\rangle_p) \cdot_q (B\langle C\rangle_r) & \textit{if } n \geq p, q, r \\
(\lambda A)\langle C\rangle_n & \to_l & \lambda(A\langle C\rangle_n) & \\
A\langle C\rangle_n & \to_l & C & \\
A\langle C\rangle_n & \to_l & A & \\
A \cdot_m B & \to_l & A \cdot_n B & \textit{if } m > n \\
A\langle B\rangle_m & \to_l & A\langle B\rangle_n & \textit{if } m > n
\end{array}$$

Note that $\rightarrow_l$ is not confluent (see the two rules for $A\langle C\rangle$); for our purposes this is no problem since $\rightarrow_l$ is only designed for proving strong normalization. The last two rules are called Decr in [Zantema 95] and are necessary to decrease the labels of applications and substitutions if inside of them a $\rightarrow_{\mathrm{Beta}}$-reduction is performed. Note that in the presence of the Decr rules we could also have $(\lambda A) \cdot_{n+1} B \rightarrow_l A\langle B\rangle_n$ for all n instead of $(\lambda A) \cdot_m B \rightarrow_l A\langle B\rangle_n$ for all $m > n$.

Lemma 4.5 *There is a precedence relation $\rhd$ such that for all $A, B \in \lambda^l$,*

$$\textit{if } A \rightarrow_l B, \textit{ then } A >_{\mathrm{rpo}} B,$$

where $>_{\mathrm{rpo}}$ is the rpo ordering induced by $\rhd$. That is, $\rightarrow_l$ is a subrelation of some recursive path order.

Proof: For $n \in \mathbb{N}$, define the precedence $\rhd$ by

$$\cdot_{n+1} \rhd \langle\ \rangle_n \rhd \cdot_n \rhd \lambda, *$$

and the status function τ by $\tau(\cdot_n) = \tau(\lambda) = \tau(\langle\rangle_n) = \mathrm{lex}$. Then $\rightarrow_l$ is a subrelation of the induced recursive path order $>_{\mathrm{rpo}}$. □

Corrollary 4.6 *The reduction relation $\rightarrow_l$ on λ^l is SN.*

Proof: By Theorem 3.2, $>_{\mathrm{rpo}}$ of Lemma 4.5 is strongly normalizing, hence by Lemma 4.5 $\rightarrow_l$ is strongly normalizing. □

In order to prove SN for $\rightarrow_{\lambda\mathrm{x}}$, we now define a translation $\mathcal{T}$ from $\lambda\mathrm{x}^{<\infty}$ to λ^l that preserves $\rightarrow_{\lambda\mathrm{x}}$-reduction steps.

Definition 4.7 *We define the translation $\mathcal{T} : \lambda\mathrm{x}^{<\infty} \rightarrow \lambda^l$ by induction on the structure of terms as follows.*

$$\begin{array}{rcll}
\mathcal{T}(x) & = & * & \\
\mathcal{T}(AB) & = & \mathcal{T}(A) \cdot_n \mathcal{T}(B) & \textit{where } n = \hat{\beta}(AB) \\
\mathcal{T}(\lambda x.A) & = & \lambda\mathcal{T}(A) & \\
\mathcal{T}(A\langle x{:=}B\rangle) & = & \mathcal{T}(A)\langle\mathcal{T}(B)\rangle_n & \textit{where } n = \hat{\beta}(A\langle x{:=}B\rangle)
\end{array}$$

Note that for all $A \in \lambda\mathrm{x}^{<\infty}$, $\mathcal{T}(A)$ is well-defined.

Lemma 4.8 *For $A \in \lambda\mathrm{x}^{<\infty}$, if $A \rightarrow_{\lambda\mathrm{x}} A'$ then $\mathcal{T}(A) \rightarrow_l^+ \mathcal{T}(A')$.*

Proof: Induction on the structure of A; we treat some of the more interesting cases.

- $A \equiv (\lambda x.A_1)A_2 \rightarrow_{\mathrm{Beta}} A_1\langle x{:=}A_2\rangle \equiv A'$.
  Then $\mathcal{T}(A) \equiv (\lambda\mathcal{T}(A_1)) \cdot_m \mathcal{T}(A_2) \rightarrow_l \mathcal{T}(A_1)\langle x{:=}\mathcal{T}(A_2)\rangle_n \equiv \mathcal{T}(A')$ where $m = \hat{\beta}(A)$; $n = \hat{\beta}(A')$; note that $m > n$ by Lemma 4.2.

- $A \equiv (A_1A_2)\langle x{:=}A_3\rangle \rightarrow_{\mathrm{x}} (A_1\langle x{:=}A_3\rangle)(A_2\langle x{:=}A_3\rangle) \equiv A'$.
  Then $\mathcal{T}(A) \equiv (\mathcal{T}(A_1) \cdot_m \mathcal{T}(A_2))\langle\mathcal{T}(A_3)\rangle_n \rightarrow_l (\mathcal{T}(A_1)\langle\mathcal{T}(A_3)\rangle_p) \cdot_n (\mathcal{T}(A_2)\langle\mathcal{T}(A_3)\rangle_q) \equiv \mathcal{T}(A')$, where $m = \hat{\beta}(A_1A_2)$, $n = \hat{\beta}(A)$, $p = \hat{\beta}(A_1\langle x{:=}A_3\rangle)$, $q = \hat{\beta}(A_2\langle x{:=}A_3\rangle)$; note that $n \geq p$ and $n \geq q$.

- $A \equiv x\langle x{:=}A_1\rangle \rightarrow_{\mathrm{x}} A_1 \equiv A'$. Then $\mathcal{T}(A) \equiv *\langle\mathcal{T}(A_1)\rangle_m \rightarrow_l \mathcal{T}(A_1) \equiv \mathcal{T}(A')$ where $m = \hat{\beta}(A)$.

- $A \equiv A_1\langle x{:=}A_2\rangle \rightarrow_{\mathrm{gc}} A_1 \equiv A'$. Then $\mathcal{T}(A) \equiv \mathcal{T}(A_1)\langle\mathcal{T}(A_2)\rangle_m \rightarrow_l \mathcal{T}(A_1) \equiv \mathcal{T}(A')$ where $m = \hat{\beta}(A)$.

- $A \equiv (\lambda y.A_1)\langle x{:=}A_2\rangle \rightarrow_{\mathrm{x}} \lambda y.(A_1\langle x{:=}A_2\rangle) \equiv A'$. Then $\mathcal{T}(A) \equiv (\lambda\mathcal{T}(A_1))\langle\mathcal{T}(A_2)\rangle_m \rightarrow_l \lambda(\mathcal{T}(A_1)\langle\mathcal{T}(A_2)\rangle_m) \equiv \mathcal{T}(A')$ where $m = \hat{\beta}(A) = \hat{\beta}(A')$.

- $A \equiv A_1A_2 \rightarrow_{\lambda\mathrm{x}} A_1'A_2 \equiv A'$. Then $\mathcal{T}(A) \equiv \mathcal{T}(A_1)\cdot_m \mathcal{T}(A_2) \overset{\mathrm{IH}}{\rightarrow_l^+} \mathcal{T}(A_1')\cdot_m \mathcal{T}(A_2) \twoheadrightarrow_l \mathcal{T}(A_1')\cdot_n \mathcal{T}(A_2)$ where $m = \hat{\beta}(A) \geq n = \hat{\beta}(A')$.

$\square$

Theorem 4.9 (PSN) *1. For all $A \in \lambda\mathrm{x}$, $A \in \lambda\mathrm{x}^{<\infty} \iff A \in SN_{\lambda\mathrm{x}}$*

2. The system $\lambda\mathrm{x}$ preserves strong normalization

Proof: The Theorem is a corollary of Lemma 4.8. In the first item, the implication from left to right follows immediately from the Lemma, using the strong normalization of $\rightarrow_l$. The implication from right to left is also immediate: if $A \notin \lambda\mathrm{x}^{<\infty}$, then for some subterm B of A, $\mathrm{x}(B)$ has an infinite β-reduction path. This can easily be turned into an infinite $\rightarrow_{\lambda\mathrm{x}}$-reduction path of A. For the second item, let A be a pure λ-term with $A \in \mathrm{SN}_\beta$. Then $A \in \lambda\mathrm{x}^{<\infty}$, so $A \in \mathrm{SN}_{\lambda\mathrm{x}}$, using the first item. $\square$

5 $\lambda\upsilon$, λs and extensions

In this section we show that our method is general enough to show PSN for other calculi of explicit substitutions such as $\lambda\upsilon$ of [BBLR 95] and λs of [Kamareddine & Rios 95], and also some extensions of $\lambda\mathrm{x}$. Furthermore, we discuss some extensions of $\lambda\mathrm{x}$, giving a counterexample to PSN similar to the one of [Melliès 95], but less involved.

5.1 The calculi $\lambda\upsilon$ and λs

Definition 5.1 *Terms and substitutions of $\lambda\upsilon$ are defined by the following abstract syntaxes.*

$$\begin{array}{rcl} a & ::= & \underline{n} \;\mid\; (aa) \;\mid\; (\lambda a) \;\mid\; (a[s]), \\ s & ::= & a/ \;\mid\; \Uparrow(s) \;\mid\; \uparrow, \end{array}$$

where n ranges over $\mathbb{N}^+$.

The reduction relation $\rightarrow_{\lambda\upsilon}$ is the union of $\rightarrow_{\upsilon\mathrm{Beta}}$ and $\rightarrow_\upsilon$ which are defined by

$$(\lambda a)b \;\rightarrow_{\upsilon\mathrm{Beta}}\; a[b/]$$

$$\begin{array}{rcl} (ab)[s] & \rightarrow_\upsilon & a[s]b[s] \\ (\lambda a)[s] & \rightarrow_\upsilon & \lambda(a[\Uparrow(s)]) \\ \underline{1}[a/] & \rightarrow_\upsilon & a \\ \underline{n+1}[a/] & \rightarrow_\upsilon & \underline{n} \\ \underline{1}[\Uparrow(s)] & \rightarrow_\upsilon & \underline{1} \\ \underline{n+1}[\Uparrow(s)] & \rightarrow_\upsilon & \underline{n}[s][\uparrow] \\ \underline{n}[\uparrow] & \rightarrow_\upsilon & \underline{n+1} \end{array}$$

Some initial intuition to motivate the reduction rules of $\lambda\upsilon$: $a[b/]$ stands for ‘substitute b for 1 in a’, $[\Uparrow(s)]$ stands for the substitution obtained by first raising all the indices in s by 1 and replacing not the index 1, but the index 2, and $[\uparrow]$ stands for the substitution that raises all numbers (in the term in front of it) by 1. An example to explain these intuitive motivations is the following. (For reasons of legibility we have removed some brackets.)

$$
\begin{array}{rll}
(\lambda(\lambda(\underline{12})))(\underline{11}) & \to_{\upsilon\mathrm{Beta}} & (\lambda(\underline{12}))[\underline{11}/] \\
 & \to_{\upsilon} & \lambda((\underline{12})[\Uparrow(\underline{11}/)]) \\
 & \to_{\upsilon} & \lambda(\underline{1}[\Uparrow(\underline{11}/)]\underline{2}[\Uparrow(\underline{11}/)]) \\
 & \to_{\upsilon} & \lambda(\underline{12}[\Uparrow(\underline{11}/)]) \\
 & \to_{\upsilon} & \lambda(\underline{1}(\underline{1}[\underline{11}/][\uparrow])) \\
 & \to_{\upsilon} & \lambda(\underline{1}(\underline{11})[\uparrow]) \\
 & \to_{\upsilon} & \lambda(\underline{1}(\underline{1}[\uparrow]\underline{1}[\uparrow])) \\
 & \to_{\upsilon} & \lambda(\underline{1}(\underline{22}))
\end{array}
$$

For a detailed explanation and motivation of the system $\lambda\upsilon$ we refer to [BBLR 95].

Definition 5.2 *Terms and substitutions of λs are defined by the following abstract syntaxes:*

$$a \quad ::= \quad \underline{n} \quad | \quad (aa) \quad | \quad (\lambda a) \quad | \quad (\phi^i_j a) \quad | \quad (a\sigma^i a)$$

where n, i range over $\mathbb{N}^+$ and j ranges over $\mathbb{N}$.

The reduction relation $\to_{\lambda s}$ is the union of $\to_{s\mathrm{Beta}}$ and $\to_s$ which are defined by

$$
\begin{array}{rll}
(\lambda a)b & \to_{s\mathrm{Beta}} & a\sigma^1 b \\
(\lambda a)\sigma^i b & \to_s & \lambda(a\sigma^{i+1}b) \\
(a_1a_2)\sigma^i b & \to_s & (a_1\sigma^i b)(a_1\sigma^i b) \\
\underline{n}\sigma^i b & \to_s & \left\{\begin{array}{lll} \underline{n-1} & \text{if} & n > i \\ \phi^i_0(b) & \text{if} & n = i \\ \underline{n} & \text{if} & n < i \end{array}\right. \\
\phi^i_k(\lambda a) & \to_s & \lambda(\phi^i_{k+1}a) \\
\phi^i_k(a_1a_2) & \to_s & (\phi^i_k a_1)(\phi^i_k a_2) \\
\phi^i_k\underline{n} & \to_s & \left\{\begin{array}{lll} \underline{n+i-1} & \text{if} & n > k \\ \underline{n} & \text{if} & n \leq k \end{array}\right.
\end{array}
$$

Again, we don't give a detailed explanation and motivation for the rules of this calculus, but refer to [Kamareddine & Rios 95]. Some initial intuition: $\sigma^i(b)$ stands for the substitution of b for i, $\phi^i_k(a)$ stands for 'raise all the numbers $n > k$ in the term a with $i - 1$'. To explain the rules, we treat the same example as for $\lambda\upsilon$.

$$
\begin{array}{rll}
(\lambda(\lambda(\underline{12})))(\underline{11}) & \to_{s\mathrm{Beta}} & (\lambda(\underline{12}))\sigma^1(\underline{11}) \\
 & \to_s & \lambda((\underline{12})\sigma^2(\underline{11})) \\
 & \to_s & \lambda(\underline{1}\sigma^2(\underline{11}))(\underline{2}\sigma^2(\underline{11})) \\
 & \to_s & \lambda(\underline{1}(\underline{2}\sigma^2(\underline{11}))) \\
 & \to_s & \lambda(\underline{1}\phi^2_0(\underline{11})) \\
 & \to_s & \lambda(\underline{1}(\phi^2_0(\underline{1})\phi^2_0(\underline{1}))) \\
 & \twoheadrightarrow_s & \lambda(\underline{1}(\underline{22}))
\end{array}
$$

The calculus λs is very similar to $\lambda\upsilon$. The difference is mainly in the moment of updating: in $\lambda\upsilon$ every step $\underline{n+1}[\Uparrow(s)] \to_\upsilon \underline{n}[s][\uparrow]$ creates an update substitution $[\uparrow]$ whereas in λs the update functionsymbol ϕ_k^i is only created at the actual moment of substitution in $n\sigma^n a \to_s \phi_0^n a$. Also, in the reductions $\underline{n}\sigma^i b \to_s \underline{n-1}$ $(n > i)$and $\underline{n}\sigma^i b \to_s \underline{n}$ $(n < i)$, there is no update function generated whereas in $\underline{n+1}[\Uparrow (s)] \to_v \underline{n}[s][\uparrow]$ an update substitution is created regardless of whether the substitution $[\uparrow (s)]$ is binding $\underline{n+1}$ or is void.

In [BBLR 95] it is shown that λv has PSN by contradicting the existence of a minimal infinite λv-reduction of a term which is SN for $\to_\beta$; in [Kamareddine & Rios 95] PSN is shown to hold for λs in a similar way.

We show that λv and λs are PSN by using the labelled calculus λ^l. The proof is very similar to the proof of PSN for λx that we gave in the previous section.

For λv and λs we have the usual properties such as SN, CR, UN for $\to_v$ respectively $\to_s$, substitution lemma, projection lemma, soundness lemma and confluence for $\to_{\lambda v}$ respectively $\to_{\lambda s}$. We denote the $\to_v$-normal form respectively $\to_s$-normal form of a term b by $v(b)$ respectively $s(b)$. Note that a substitution of λv is of the form $\Uparrow^n (b/)$ or $\Uparrow^n (\uparrow)$ for some n.

We denote β-reduction on λv-terms as well as on λs-terms by $\to_\beta$; for a λv- respectively λs-term a we write $\hat{\beta}(a)$ to denote the maximal number of β-reduction steps starting from $v(a)$ respectively $s(a)$, if this number exists.

Definition 5.3

$$\begin{array}{rcl} \lambda v^{<\infty} & := & \{a \in \lambda v \mid \forall b \subseteq a[v(b) \in SN_\beta]\} \\ \lambda s^{<\infty} & := & \{a \in \lambda s \mid \forall b \subseteq a[s(b) \in SN_\beta]\} \end{array}$$

Lemma 5.4 *1. $\lambda v^{<\infty}$ is closed under $\to_{\lambda v}$-reduction*

2. $\lambda s^{<\infty}$ is closed under $\to_{\lambda s}$-reduction

Definition 5.5 *1. $\mathcal{T}_v : \lambda v^{<\infty} \longrightarrow \lambda^l$ is defined by*

$$\begin{array}{rcll} \mathcal{T}_v(\underline{n}) & = & * & \\ \mathcal{T}_v(ab) & = & \mathcal{T}_v(a) \cdot_p \mathcal{T}_v(b) & \quad \textit{where } p = \hat{\beta}(ab) \\ \mathcal{T}_v(\lambda a) & = & \lambda \mathcal{T}_v(a) & \\ \mathcal{T}_v(a[\Uparrow^n (b/)]) & = & \mathcal{T}_v(a)\langle \mathcal{T}_v(b)\rangle_p & \quad \textit{where } p = \hat{\beta}(a[\Uparrow^n (b/)]) \\ \mathcal{T}_v(a[\Uparrow^n (\uparrow)]) & = & \mathcal{T}_v(a) & \end{array}$$

2. $\mathcal{T}_s : \lambda s^{<\infty} \longrightarrow \lambda^l$ is defined by

$$\begin{array}{rcll} \mathcal{T}_s(\underline{n}) & = & * & \\ \mathcal{T}_s(ab) & = & \mathcal{T}_s(a) \cdot_p \mathcal{T}_s(b) & \quad \textit{where } p = \hat{\beta}(ab) \\ \mathcal{T}_s(\lambda a) & = & \lambda \mathcal{T}_s(a) & \\ \mathcal{T}_s(a\sigma^i b) & = & \mathcal{T}_s(a)\langle \mathcal{T}_s(b)\rangle_p & \quad \textit{where } p = \hat{\beta}(a\sigma^i b) \\ \mathcal{T}_s(\phi_j^i a) & = & \mathcal{T}_s(a) & \end{array}$$

Lemma 5.6 *1. If $a \in \lambda v^{<\infty}$ and $a \to_v b$ then $\mathcal{T}_v(a) \twoheadrightarrow_l \mathcal{T}_v(b)$*

2. If $a \in \lambda v^{<\infty}$ and $a \to_{v\mathrm{Beta}} b$ then $\mathcal{T}_v(a) \twoheadrightarrow_l^+ \mathcal{T}_v(b)$

3. If $a \in \lambda s^{<\infty}$ and $a \to_s b$ then $\mathcal{T}_s(a) \twoheadrightarrow_l \mathcal{T}_s(b)$

4. If $a \in \lambda s^{<\infty}$ and $a \to_{s\mathrm{Beta}} b$ then $\mathcal{T}_s(a) \twoheadrightarrow_l^+ \mathcal{T}_s(b)$

Proof: Induction on the structure of a. □

Theorem 5.7 *1. $a \in \lambda v^{<\infty} \Longleftrightarrow a \in SN_{\lambda v}$*

2. $\to_{\lambda v}$ has PSN

3. $a \in \lambda s^{<\infty} \Longleftrightarrow a \in SN_{\lambda s}$

4. $\rightarrow_{\lambda s}$ *has PSN*

Proof:

1. $\Rightarrow$ by projection; $\Leftarrow$: since $\rightarrow_{\upsilon}$ is SN, any infinite $\rightarrow_{\lambda\upsilon}$-reduction must contain infinitely many $\rightarrow_{\upsilon\mathrm{Beta}}$-steps. Therefore an infinite reduction of a pure term which is SN for $\rightarrow_{\beta}$ translates by $\mathcal{T}_{\upsilon}$ into an infinite $\rightarrow_{l}$-reduction which is impossible by 4.6.

2. follows from 1.

3. & 4. similar to 1. & 2.

□

5.2 Extensions of λx

In this section we consider several extensions of λx with some kind of composition. The calculus $\lambda\sigma$ of [Abadi et al. 90] was designed to be able to compose substitutions. The price however is not having PSN (cf. [Melliès 95]). Since λx has no composition but does have PSN, it is an interesting question where the borderline is between PSN and composition of substitutions.

We start with a short discussion of $\lambda\sigma$. For the precise definition of $\lambda\sigma$, the reader is referred to [Abadi et al. 90]. The composition of substitutions in $\lambda\sigma$ is mainly performed by two rules, *Comp* and *Map*. The first glues two substitutions together: $a[s][t] \xrightarrow{\text{Comp}} a[s\circ t]$, while Map allows the distribution of the second substitution over the first: $(b\cdot c\cdot s')\circ t \xrightarrow{\text{Map}} b[t]\cdot((c\cdot s')\circ t) \xrightarrow{\text{Map}} b[t]\cdot c[t]\cdot(s'\circ t)$.

As was pointed out in [Kamareddine & Nederpelt 93], the substitutions of $\lambda\sigma$ are roughly the same as simultaneous parallel substitutions in the following extension of λx:

$$\text{terms} \quad t ::= \quad x \;\;\big|\;\; tt \;\;\big|\;\; \lambda x.t \;\;\big|\;\; t\langle\vec{x}{:=}\vec{t}\rangle$$

where $\langle\vec{x}{:=}\vec{t}\rangle$ is shorthand for $\langle x_1,\ldots,x_m{:=}t_1,\ldots,t_m\rangle$; reductions are similar as for λx plus the composition rule

$$a\langle\vec{x}{:=}\vec{b}\rangle\langle\vec{y}{:=}\vec{c}\rangle \longrightarrow a\langle\vec{x},\vec{y}{:=}b_1\langle\vec{y}{:=}\vec{c}\rangle,\ldots,b_m\langle\vec{y}{:=}\vec{c}\rangle,\vec{c}\rangle$$

In this calculus one can imitate the counterexample to PSN of $\lambda\sigma$ (cf [Melliès 95]). In fact, even the calculus λx extended with the rule

$$a\langle x{:=}b\rangle\langle y{:=}c\rangle \longrightarrow a\langle x{:=}b\langle y{:=}c\rangle\rangle \text{ if } y \notin FV(a)$$

(no simultaneous substitutions required) doesn't have PSN. We give an infinite derivation starting from the term $(\lambda y.(\lambda y.a)b)\big((\lambda y.a)b\big)$. Note that this term is even simpler than the term used in [Melliès 95].

First we define substitutions α_i for $m \in I\!N$ by

$$\begin{array}{rcl} \alpha_0 & \equiv & \langle y{:=}(\lambda y.a)b\rangle \\ \alpha_{m+1} & \equiv & \langle y{:=}b\alpha_m\rangle \end{array}$$

Now consider the following three reductions. (For simplicity we forget about the variable convention during this counterexample; furthermore, we freely change bound variables if convenient.)

$$\begin{array}{rcl} (\lambda y.(\lambda y.a)b)\big((\lambda y.a)b\big) & \twoheadrightarrow & \big((\lambda y.a)b\big)\langle y{:=}(\lambda y.a)b\rangle \\ & \twoheadrightarrow & \big(\lambda y.a\langle y{:=}(\lambda y.a)b\rangle\big)\big(b\langle y{:=}(\lambda y.a)b\rangle\big) \\ & \equiv & (\lambda y.a\alpha_0)(b\alpha_0) \\ & \rightarrow & a\alpha_0\langle y{:=}b\alpha_0\rangle \\ & \equiv & a\alpha_0\alpha_1 \end{array}$$

$$\begin{array}{rcl}a\alpha_0\alpha_{m+1} & \equiv & a\langle y' {:=} (\lambda y.a)b\rangle\langle y{:=}b\alpha_m\rangle & \rightarrow & a\langle y'{:=}((\lambda y.a)b)\langle y{:=}b\alpha_m\rangle\rangle \\ & \twoheadrightarrow & a\langle y'{:=}(\lambda y.a\langle y{:=}b\alpha_m\rangle)(b\langle y{:=}b\alpha_m\rangle)\rangle & \equiv & a\langle y'{:=}(\lambda y.a\alpha_{m+1})(b\alpha_{m+1})\rangle \\ & \rightarrow & a\langle y'{:=}a\alpha_{m+1}\langle y{:=}b\alpha_{m+1}\rangle\rangle & \equiv & a\langle y'{:=}a\alpha_{m+1}\alpha_{m+2}\rangle\end{array}$$

$$a\alpha_{m+1}\alpha_{n+1} \equiv a\langle y'{:=}b\alpha_m\rangle\langle y{:=}b\alpha_n\rangle \rightarrow a\langle y'{:=}b\alpha_m\langle y{:=}b\alpha_n\rangle\rangle \equiv a\langle y'{:=}b\alpha_m\alpha_{n+1}\rangle$$

These combine into an infinite derivation in the following way.
$(\lambda y.(\lambda y.a)b)\,((\lambda y.a)b) \twoheadrightarrow$

$$\begin{array}{cccccc} & a\alpha_0\alpha_1 & \twoheadrightarrow & \ldots a\alpha_1\alpha_2\ldots & \twoheadrightarrow & \ldots a\alpha_0\alpha_2\ldots \\ \twoheadrightarrow & \ldots a\alpha_2\alpha_3\ldots & \twoheadrightarrow & \ldots a\alpha_1\alpha_3\ldots & \twoheadrightarrow & \ldots a\alpha_0\alpha_3\ldots \\ & \vdots & & \vdots & & \vdots \\ \twoheadrightarrow & \ldots a\alpha_0\alpha_{m+1}\ldots & \twoheadrightarrow & \ldots a\alpha_{m+1}\alpha_{m+2}\ldots & \twoheadrightarrow & \ldots a\alpha_0\alpha_{m+2}\ldots\end{array}$$

Recall that we proved PSN by showing that for every term A: if the x-normal forms of all its subterms are in SN_β, then $A \in \mathrm{SN}_{\rightarrow_{\lambda\mathrm{x}}}$. With the extra composition reduction defined above, there is an easy counterexample to that: the term $x\langle y{:=}zz\rangle\langle z{:=}\lambda w.ww\rangle$ has x-normal form x (and is also SN for λx-reduction), but it has Ω as a subterm of a reduct if composition is allowed. Observe that this term violates Lemma 4.3.

This example also shows why our method fails for the system extended with the extra composition rule, and hence also for $\lambda\sigma$: $\mathcal{T}(x\langle y{:=}zz\rangle\langle z{:=}\lambda w.ww\rangle) \equiv *\langle * \cdot_0 *\rangle_0\langle\lambda(* \cdot_0 *)\rangle_0$ whereas after composition of the two substitutions, the label of the innermost substitution does not exist: $\mathcal{T}(x\langle y{:=}(zz)\langle z{:=}\lambda w.ww\rangle\rangle) \equiv *\langle(* \cdot_0 *)\langle\lambda(* \cdot_0 *)\rangle_\infty\rangle_0$. So reduction in $\lambda\sigma$ does not always decrease $\mathcal{T}$-images.

One can try to give a rule for composition of substitutions such that reduction still decreases $\mathcal{T}$-images, the following rule seems best fit for this purpose:

$$a\langle x{:=}b\rangle\langle y{:=}c\rangle \rightarrow a\langle x{:=}b\langle y{:=}c\rangle\rangle \text{ if } y \notin FV(\mathrm{x}(a)), x \in FV(\mathrm{x}(a))$$

The idea behind this rule is that, if $x \in FV(\mathrm{x}(a))$, then $b\langle y{:=}c\rangle$ will occur as a subterm of some $\rightarrow_{\lambda\mathrm{x}}$-reduct of $a\langle x{:=}b\rangle\langle y{:=}c\rangle$. Hence allowing to create $b\langle y{:=}c\rangle$ at this point will not spoil PSN. Below we show that indeed this calculus has PSN.

Definition 5.8 $\lambda\mathrm{xc}^-$ *is the calculus with as terms the terms of* λx *and reduction rules those of* λx *and the extra rule*

$$a\langle x{:=}b\rangle\langle y{:=}c\rangle \rightarrow_{\mathrm{c}^-} a\langle x{:=}b\langle y{:=}c\rangle\rangle \qquad \textit{if } x \in FV(\mathrm{x}(a)), y \notin FV(\mathrm{x}(a)).$$

First of all, we show that adding the c^--rule does not spoil our substitution calculus:

Lemma 5.9 *Let a, b, c be terms of* λx *and x, y variables such that $x \in FV(\mathrm{x}(a))$ and $y \notin FV(\mathrm{x}(a))$. Then $\mathrm{x}(a\langle x{:=}b\rangle\langle y{:=}c\rangle) \equiv \mathrm{x}(a\langle x{:=}b\langle y{:=}c\rangle\rangle)$.*

Proof: By Lemma 2.7 we have $\mathrm{x}(a\langle x{:=}b\rangle\langle y{:=}c\rangle) \equiv \mathrm{x}(a)[\mathrm{x}(b)/x][\mathrm{x}(c)/y]$ and $\mathrm{x}(a\langle x{:=}b\langle y{:=}c\rangle\rangle) \equiv \mathrm{x}(a)[\mathrm{x}(b)[\mathrm{x}(c)/y]/x]$. By an elementary lemma about substitution in ordinary λ-calculus we have $\mathrm{x}(a)[\mathrm{x}(b)/x][\mathrm{x}(c)/y] \equiv \mathrm{x}(a)[\mathrm{x}(c)/y][\mathrm{x}(b)[\mathrm{x}(c)/y]/x]$ and since $y \notin FV(\mathrm{x}(a))$, the latter expression equals $\mathrm{x}(a)[\mathrm{x}(b)[\mathrm{x}(c)/y]/x]$. □

Lemma 5.10 xc^-*-reduction is SN.*

Proof: We translate $\lambda\mathrm{xc}^-$-terms into λ^l as follows:

$$\begin{array}{rcl}\mathcal{T}'(x) & \equiv & * \\ \mathcal{T}'(ab) & \equiv & \mathcal{T}'(a) \cdot_0 \mathcal{T}'(b) \\ \mathcal{T}'(\lambda x.a) & \equiv & \lambda\mathcal{T}'(a) \\ \mathcal{T}'(a\langle x{:=}b\rangle) & \equiv & \mathcal{T}'(a)\langle\mathcal{T}'(b)\rangle_0\end{array}$$

Let $\rhd$ be the precedence defined in Lemma 4.5 (so $\langle\ \rangle_0 \rhd \cdot_0 \rhd \lambda, *$), and $>_{\rm rpo}$ the induced rpo on λ^l. Now it is straightforward to show that for λxc$^-$-terms a, b, if $a \to_{\rm xc^-} b$ then $\mathcal{T}'(a) >_{\rm rpo} \mathcal{T}'(b)$. We show the crucial case $a\langle x{:=}b\rangle\langle y{:=}c\rangle \to_{\rm c^-} a\langle x{:=}b\langle y{:=}c\rangle\rangle$. Then $\mathcal{T}'(a\langle x{:=}b\rangle\langle y{:=}c\rangle) \equiv \mathcal{T}'(a)\langle \mathcal{T}'(b)\rangle_0\langle \mathcal{T}'(c)\rangle_0$ and $\mathcal{T}'(a\langle x{:=}b\langle y{:=}c\rangle\rangle) \equiv \mathcal{T}'(a)\langle \mathcal{T}'(b)\langle \mathcal{T}'(c)\rangle_0\rangle_0$. Thus we are done if we show the following three inequalities:

$$\begin{array}{lll}
\mathcal{T}'(a)\langle \mathcal{T}'(b)\rangle_0\langle \mathcal{T}'(c)\rangle_0 & >_{\rm rpo} & \mathcal{T}'(a) \\
\mathcal{T}'(a)\langle \mathcal{T}'(b)\rangle_0\langle \mathcal{T}'(c)\rangle_0 & >_{\rm rpo} & \mathcal{T}'(b)\langle \mathcal{T}'(c)\rangle_0 \\
\big(\mathcal{T}'(a)\langle \mathcal{T}'(b)\rangle_0, \mathcal{T}'(c)\big) & >^{\rm lex}_{\rm rpo} & \big(\mathcal{T}'(a), \mathcal{T}'(b)\langle \mathcal{T}'(c)\rangle_0\big)
\end{array}$$

The first inequality holds since $\mathcal{T}'(a)$ is a subterm of the left hand side, the third inequality holds since $\mathcal{T}'(a)$ is a subterm of $\mathcal{T}'(a)\langle \mathcal{T}'(b)\rangle_0$; note that the lexicographic extension is crucial here. The second inequality holds if we show the following three inequalities:

$$\begin{array}{lll}
\mathcal{T}'(a)\langle \mathcal{T}'(b)\rangle_0\langle \mathcal{T}'(c)\rangle_0 & >_{\rm rpo} & \mathcal{T}'(b) \\
\mathcal{T}'(a)\langle \mathcal{T}'(b)\rangle_0\langle \mathcal{T}'(c)\rangle_0 & >_{\rm rpo} & \mathcal{T}'(c) \\
\big(\mathcal{T}'(a)\langle \mathcal{T}'(b)\rangle_0, \mathcal{T}'(c)\big) & >^{\rm lex}_{\rm rpo} & \big(\mathcal{T}'(b), \mathcal{T}'(c)\big)
\end{array}$$

The first two inequalities hold since the right hand side is a subterm of the left hand side, the third holds since $\mathcal{T}'(b)$ is a subterm of $\mathcal{T}'(a)\langle \mathcal{T}'(b)\rangle_0$. □

Lemma 5.11 $\to_{\rm xc^-}$ *and* $\to_{\lambda{\rm xc}^-}$ *are confluent.*

Proof: We can imitate the proof of Theorem 2.10 since by Lemma 5.9, $\to_{\rm c^-}$ doesn't change x-normalforms. □

Lemma 5.12 λxc$^-$ *has PSN.*

Proof: We extend the reduction relation $\to_l$ on λ^l with the following rule:

$$A\langle B\rangle_m\langle C\rangle_n \to_l A\langle B\langle C\rangle_p\rangle_q \qquad \text{if } n \geq p, q.$$

In order to show that Lemma 4.5 and Corollary 4.6 still hold for this extended reduction $\to_l$, we only need to check that for $n \geq p, q$, $A\langle B\rangle_m\langle C\rangle_n >_{\rm rpo} A\langle B\langle C\rangle_p\rangle_q$. This can be shown similar to the proof of Lemma 5.10. Again, it is crucial that $\langle\rangle$ be given the lexicographic extension by τ.

Now we can show that $\to_{\lambda{\rm xc}^-}$-reduction is preserved by $\mathcal{T}$ of Definition 4.7: all we need to check is that if $M \to_{\rm c^-} N$ at the root, then $\mathcal{T}(M) \to_l \mathcal{T}(N)$. So, suppose that $M \equiv A\langle x{:=}B\rangle\langle y{:=}C\rangle$ and $N \equiv A\langle x{:=}B\langle y{:=}C\rangle\rangle$ with $x \in FV(\mathrm{x}(A))$, $y \notin FV(\mathrm{x}(A))$. Then $\mathcal{T}(M) \equiv \mathcal{T}(A)\langle \mathcal{T}(B)\rangle_m\langle \mathcal{T}(C)\rangle_n$ with $m = \hat\beta(A\langle x{:=}B\rangle)$, $n = \hat\beta(M)$ and $\mathcal{T}(N) \equiv \mathcal{T}(A)\langle \mathcal{T}(B)\langle \mathcal{T}(C)\rangle_p\rangle_q$ with $p = \hat\beta(B\langle y{:=}C\rangle)$, $q = \hat\beta(N)$. Then $n = q$, by Lemma 5.9, and $p \leq n$ because $\mathrm{x}(B\langle y{:=}C\rangle)$ is a subterm of $\mathrm{x}(M)$, due to the occurrence of x in $\mathrm{x}(A)$. Therefore $\mathcal{T}(M) \to_l \mathcal{T}(N)$.

Now, similar to Theorem 4.9 we have as consequences that the sets $\lambda\mathrm{x}^{<\infty}$ and $SN_{\lambda{\rm xc}^-}$ are the same and hence we conclude that PSN holds for λxc$^-$. □

that been

6 Proof of PSN using labelled trees

In this section we outline a proof of PSN, again using the RPO technique, but now in the way it has been presented in [Klop 92]. One then looks at the collection of *commutative finite labelled trees* Tree (i.e. trees are identified upto permutation of branches: there is no order from left to right in the subtrees). The labels are taken from $\mathbb{N}$. Furthermore, one looks at the set Tree*, where some nodes in a tree may have a marker $\star$. It is convenient to denote the tree with root node n and subtrees $t_1, \ldots, t_p$ by $n(t_1, \ldots, t_p)$, and similarly, if the root node has a marker, by $n^\star(t_1, \ldots, t_p)$. In the following, we abbreviate $t_1, \ldots, t_p$ to $\vec{t}$. On these commutative labelled trees with markers (the set Tree*), a reduction relation $\Rrightarrow$ is defined.

Definition 6.1 *The relation* $\Rrightarrow$ *on* $\mathsf{Tree}^\star$ *is defined as follows.*

$$\begin{array}{rcl}
n(\vec{t}) & \Rrightarrow & n^\star(\vec{t}), \\
n^\star(\vec{t}) & \Rrightarrow & m(n^\star(\vec{t}),\ldots,n^\star(\vec{t})), \\
 & & \textit{if } m < n, \textit{ zero or more copies of } n^\star(\vec{t}), \\
n^\star(s,\vec{t}) & \Rrightarrow & n(s^\star,\ldots,s^\star,\vec{t}), \\
 & & \textit{zero or more copies of } s, \\
n^\star(\vec{t}) & \Rrightarrow & t_i, \\
 & & 1 \le i \le p.
\end{array}$$

Furthermore, the relation $\Rrightarrow$ *is compatible with the tree-forming operations, that is, if* $t_i \Rrightarrow t_i'$, *then* $n(t_1,\ldots,t_i,\ldots,t_p) \Rrightarrow n(t_1,\ldots,t_i',\ldots,t_p)$.

As usual, the relation $\Rrightarrow^+$ *denotes the transitive closure of* $\Rrightarrow$ *and* $\Rrightarrow^*$ *denotes the transitive reflexive closure of* $\Rrightarrow$.

For examples on the use of these rules we refer to [Klop 92]. we just mention the main result, which will be applied here to the problem of PSN for explicit substituion.

Theorem 6.2 ([Klop 92],[Dershowitz 79]) *The relation* $\Rrightarrow^+$ *is well-founded on* Tree *(the set of trees without markers).*

To prove PSN for the calculus λx, we now proceed by defining a reduction preserving mapping T from $\lambda\mathrm{x}^{<\infty}$ to Tree: if $M \to_{\mathrm{x}} N$, then $M \Rrightarrow^* N$ and if $M \to_{\mathrm{Beta}} N$, then $M \Rrightarrow^+ N$. Hence, using the fact that $\to_{\mathrm{x}}$ is strongly normalizing, we can again conclude that every $M \in \lambda\mathrm{x}^{<\infty}$ is λx-SN and so that λx has the PSN property.

For notational convenience, we abbreviate the sequence of definitions $\langle x_1{:=}P_1\rangle\cdots\langle x_n{:=}P_n\rangle$ to $\overline{\langle x{:=}P\rangle}$.

Definition 6.3 *For* $M \in \lambda\mathrm{x}^{<\infty}$, *we define the tree* $T(M)$ *by induction on the length of* M *as follows.*

$$T(x) = 0,$$

$$T(QN) = \begin{tikzcd} & \hat{\beta}(QN) \arrow[dl] \arrow[dr] & \\ T(Q) & & T(N) \end{tikzcd}$$

$$T(\lambda y.N) = T(N)$$

$$T(y\overline{\langle x{:=}P\rangle}) = \begin{tikzcd} & 0 \arrow[dl] \arrow[dr] & \\ T(P_1) & \cdots & T(P_n) \end{tikzcd}$$

if $y \notin \{x_1,\ldots,x_n\}$

$$T(x_i\overline{\langle x{:=}P\rangle}) = \begin{tikzcd} & & & 0 \arrow[dlll] \arrow[dl] \arrow[dr] & \\ T(P_1) & \cdots & T(P_{i-1}) & & T(P_i\langle x_{i+1}{:=}P_{i+1}\rangle\cdots\langle x_n{:=}P_n\rangle) \end{tikzcd}$$

$$T((QN)\overline{\langle x{:=}P\rangle}) \quad = \quad \begin{tikzcd}[column sep=small] & \hat{\beta}((QN)\overline{\langle x{:=}P\rangle}) \arrow[dl] \arrow[dr] & \\ T(Q\overline{\langle x{:=}P\rangle}) & & T(N\overline{\langle x{:=}P\rangle}) \end{tikzcd}$$

$$T((\lambda y.N)\overline{\langle x{:=}P\rangle}) \quad = \quad T(N\overline{\langle x{:=}P\rangle})$$

The following Lemmas show that T preserves reductions (in the right sense as announced above). The proofs of these Lemmas are not difficult, the main complication being to find out the right induction loadings (and the right order in which the induction should be done). We just outline the proofs.

Lemma 6.4 *For $M \in \lambda\mathrm{x}^{<\infty}$, if $M \to_{\mathrm{x}} N$, then $T(M) \mathrel{=\!\!\rhd}^{*} T(N)$.*

Proof: By induction on the length of M, distinguishing subcases according to the structure of M. Note that we need Lemma 4.3 to make sure that $N \in \lambda\mathrm{x}^{<\infty}$ and hence that $T(N)$ is well-defined. □

The following two Lemmas are sublemmas necessary for the proof of preservation of $\to_{\mathrm{Beta}}$-reduction by T.

Lemma 6.5 *For $N\overline{\langle x{:=}P\rangle} \in \lambda\mathrm{x}^{<\infty}$, $T(N\overline{\langle x{:=}P\rangle}) \mathrel{=\!\!\rhd}^{*} T(N)$.*

Proof: By induction on the length of N. □

Lemma 6.6 *For $((\lambda y.N)Q)\overline{\langle x{:=}P\rangle} \in \lambda\mathrm{x}^{<\infty}$, $T(((\lambda y.N)Q)\overline{\langle x{:=}P\rangle}) \mathrel{=\!\!\rhd}^{+} T(N\langle y{:=}Q\rangle\overline{\langle x{:=}P\rangle})$.*

Proof: By induction on the length of N, using Lemma 6.5. First write N as $R\overline{\langle y{:=}Q\rangle}$, with R not a term that ends with a substitution item. (So, the sequence $\overline{\langle y{:=}Q\rangle}$ should be taken as long as possible.) Then distinguish cases according to the structure of R. □

Corrollary 6.7 *For $M \in \lambda\mathrm{x}^{<\infty}$, if $M \to_{\mathrm{Beta}} N$, then $T(M) \mathrel{=\!\!\rhd}^{+} T(N)$.*

Proof: By induction on the structure of M, using Lemma 6.6 for the base case when M itself is the contracted Beta-redex. □

Theorem 6.8 *The calculus $\lambda\mathrm{x}$ has the PSN property.*

Proof: If M is a β-SN pure λ-term, then $M \in \lambda\mathrm{x}^{<\infty}$. If M has an infinite λx-reduction path, then $T(M)$ has an infinite $\mathrel{=\!\!\rhd}$-reduction path, due to Lemma 6.4 and Corollary 6.7, contradicting Theorem 6.2. □

7 Conclusions

We have introduced a new method for proving PSN for λ-calculi with explicit substitution. The method involves four steps:

- determine a suitable set contained in the set of strongly normalizing terms in the explicit substitution calculus, containing the pure β-SN terms and closed under explicit substitution reduction,
- give a translation from this set into a first order term rewrite system,
- define a strongly normalizing reduction relation on this TRS by giving a well-founded precedence,

- show that the translation preserves infinite reduction paths.

For named calculi, the translation identifies all variables; for calculi using de Bruijn indices the translation identifies all indices and erases update functions, giving evidence for the statement 'update functions do not matter for termination issues'. Kruskal's theorem ensures that a well-founded precedence yields a strongly normalizing term rewrite system.

Further applications of this method that are under investigation:

- give a maximal strategy for λx-reduction and an inductive characterization of the set $\lambda x^{<\infty}$.
- give a general PSN proof for combinatory reduction systems with explicit substitution (cf. [Rose 95], [Bloo & Rose 96])
- give a (first order) calculus with explicit substitution which has PSN as well as confluence on open terms.

8 Acknowledgements

Thanks to Thomas Arts for making us aware of current notations for the recursive path orders, using semantic labelling. We have also benefitted from discussions with the following people: Hans Zantema, Gilles Barthe, Daniel Briaud, Twan Laan, Pierre Lescanne, Rob Nederpelt and Kristoffer Rose.

References

[Abadi et al. 90] Abadi, M., Cardelli, L., Curien, P.-L., and Lévy, J.-J., Explicit substitutions, in POPL '90—Seventeenth Annual ACM Symposium on Principles of Programming Languages (San Francisco, California, jan. 1990).

[Abramsky et al. 1992] Abramsky, S., Gabbay, Dov M., and Maibaum, T. S. E. (eds.), *Handbook of Logic in Computer Science, Vol. II*, Oxford University Press, 1992.

[BBLR 95] Benaissa, Z.E.A., Briaud, D., Lescanne, P. and Rouyer-Degli, J., λv, a calculus of explicit substitutions which preserves strong normalization, *Journal of Functional Programming*, vol. 6, nr. 5, 1996.

[Bloo 95] Bloo, R., Preservation of Strong Normalization for Explicit Substitution, Computing Science Report 95-08, Eindhoven University of Technology.

[Bloo & Rose 95] Bloo, R., and Rose, K. H., Preservation of Strong Normalization in Named Lambda Calculi with Explicit Substitution and Garbage Collection, in: J.C. van Vliet, ed., *Proceedings of CSN'95* (Computing Science in the Netherlands), ISBN 90 6196 460 1, also available as technical report via WWW; URL: ftp://ftp.diku.dk/diku/semantics/papers/D-246.ps.

[Bloo & Rose 96] Bloo, R., and Rose, K. H., Combinatory Reduction Systems with Explicit Substitution that Preserve Strong Normalization, *Lecture Notes in Computer Science*, Vol. 1103, Rewrite Techniques and Applications '96, pages 169–183 , Springer-Verlag, 1996.

[Dershowitz 79] Dershowitz, N., A note on simplification orderings, *Inf. Proc. Letters* **9** (5): 212–215, 1979

[Ferreira & Zantema 95] Ferreira, M.C.F., and Zantema, H., Well-foundedness of Term Orderings, proceedings of CTRS-94, Springer, 1995, LNCS vol. 968, pp. 106–123.

[FKP 97] Ferreira, Kesner and Puel, λ-calculi with explicit substitutions and composition which preserve β-strong normalization (Extended Abstract), in: Hanus, M., and Rodríguez-Artalejo, M. (eds.), *Proceedings of Algebraic and Logic Programming '96*, Lecture Notes in Computer Science, Vol. 1139, pages 284–298, Springer-Verlag, 1996.

[Kamareddine & Nederpelt 93] Kamareddine, F., and Nederpelt, R.P., On stepwise explicit substitution, *International Journal of Foundations of Computer Science 4 (3)*, 197–240, 1993.

[Kamareddine & Rios 95] Kamareddine, F., and Rios, A., λ-calculus à la de Bruijn & explicit substitution, *Lecture Notes in Computer Science*, Vol. 982, 7th international symposium on Programming Languages: Implementations, Logics and Programs, PLILP '95, pages 45–62 , Springer-Verlag, 1995.

[Klop 92] Klop, J. W., Term rewrite systems, in: [Abramsky et al. 1992].

[Melliès 95] Melliès, P.-A., Typed λ-calculi with explicit substitutions may not terminate, in: *Proceedings of TLCA'95* , Lecture Notes in Computer Science, Vol. 902, eds. M. Dezani-Ciancaglini and G. Plotkin.

[Munoz 96] Muñoz, C., Confluence and Preservation of Strong Normalization in an Explicit Substitutions Calculus, in: *Proceedings of LICS '96*, IEEE Computer Society Press, 1996.

[Rose 95] Rose, K.H., Combinator Reduction Systems with Explicit Substitution, in: *Proceedings of HOA'95*, (Second International Workshop on Higher-Order Algebra, Logic and Term Rewriting), Paderborn, Germany, 1995, also available as technical report via WWW; URL: ftp://ftp.diku.dk/diku/semantics/papers/D-247.ps.

[Zantema 95] Zantema, H., Termination of Term Rewriting by Semantic Labelling, *Fundamenta Informaticae*, Vol. **24** (1,2), pp. 89–105, 1995