Explicit substitution : on the edge of strong normalisation

Eindhoven University of Technology
Department of Mathematics and Computing Science

Explicit Substitution: on the Edge of Strong Normalisation

by

R. Bloo and H. Geuvers

96/10

ISSN 0926-4515

editors: prof.dr. R.C. Backhouse
prof.dr. J.C.M. Baeten

Reports are available at:
http://www.win.tue.nl/win/cs

Computing Science Report 96/10
Eindhoven, April 1996

Explicit Substitution: on the Edge of Strong Normalisation

Roel Bloo* bloo@win.tue.nl
Herman Geuvers herman@win.tue.nl

1 Abstract

We use the Recursive Path Ordering (RPO) technique of semantic labelling to show the *Preservation of Strong Normalisation* (PSN) property for several calculi of explicit substitution. Preservation of Strong Normalisation 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: $\lambda\upsilon$ of Lescanne et al., λs of Kamareddine and Ríos and λx of Rose and Bloo.

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

2 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]) have succeeded in giving calculi of explicit substitutions which have the nice property that every term which is strongly normalising for β-reduction is also strongly normalising in the explicit substitution calculus. We call this property: PSN (Preservation of Strong Normalisation).

In this paper we present a method to prove PSN for explicit substitution calculi based on the recursive path order. Zantema used the recursive path order to show termination of the substitution part of $\lambda\sigma$ [Zantema 94], but the technique he used doesn't apply to show PSN. We use a stronger technique called semantic labelling [Ferreira & Zantema 94] 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 labeled with natural numbers and where variables are represented by just one constant $*$. The recursive path order $>_{\mathrm{rpo}}$ on this labelled calculus is strongly normalising (or: terminating).

Then we take a look at the explicit substitution calculus λx. The β-reduction is here split up into a reduction step $\rightarrow_{\mathrm{Beta}}$ (contracting the β-redex and creating an explicit substitution) and reduction steps $\rightarrow_{\mathrm{x}}$ (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 normalising and confluent.

Now—and this is a crucial point in the proof of PSN—we take a look at the terms in λx of which the *substitution normal form* of all of its subterms is β-SN; we call this set $\lambda\mathrm{x}^{<\infty}$. (The substitution normal form of a term M is obtained by evaluating all the explicit substitutions in M, not contracting any β-redexes. That is, we take the $\rightarrow_{\mathrm{x}}$-normal-form of M.) We then define a translation $\mathcal{T}$ from $\lambda\mathrm{x}^{<\infty}$ into the set of labelled terms. This translation $\mathcal{T}$ is reduction preserving

*address of both authors: Eindhoven University of Technology, P.O.Box 513, NL–5600 MB Eindhoven.

in the sense that, if $M \to_{\mathrm{x}} N$, then $\mathcal{T}(M) >_{\mathrm{rpo}} \mathcal{T}(N)$ or $\mathcal{T}(M) = \mathcal{T}(N)$ and if $M \to_{\mathrm{Beta}} N$, then $\mathcal{T}(M) >_{\mathrm{rpo}} \mathcal{T}(N)$. Hence, using the fact that $\to_{\mathrm{x}}$ is strongly normalizing, we conclude that every $M \in \lambda\mathrm{x}^{<\infty}$ is strongly normalizing. So, λx 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 from $\lambda\mathrm{x}^{<\infty}$ to commutative labelled trees. This translation is also reduction preserving in the same sense as just discussed. This gives a slightly different way of obtaining the PSN property for λx.

To show the flexibility of our proof method we use it for different calculi of explicit substitution. We start off with a calculus where we use 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 slightly more perspicuous. Moreover, it makes it easier to single out the places where the difficulties arise in the calculus of [Abadi et al. 90]. We also apply our proof method to the calculi λv of Lescanne et al. and λs of Kamareddine and Ríos.

3 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.

$$\begin{array}{rcl} x[N/x] & := & N, \\ y[N/x] & := & y, \text{ if } y \not\equiv x, \\ (PQ)[N/x] & := & P[N/x]Q[N/x], \\ (\lambda y.P)[N/x] & := & \lambda y'.P[y'/y][N/x], \text{ if } y' \notin FV(N) \cup \{x\} \cup (FV(P) \setminus \{y\}) \\ (\lambda x.P)[N/x] & := & \lambda x.P. \end{array}$$

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

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

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

In order to get a calculus λx of explicit substitutions, two extensions have to be made. The first is extending the terms with substitutions:

Definition 3.2 *The set of terms λx 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 λx-terms as for λ-terms but with the extra clauses that

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

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

α-equivalence is defined on λx-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 the global substitution in $\rightarrow_\beta$ explicit 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 3.3 *The reduction relations $\rightarrow_{\mathrm{Beta}}$ and $\rightarrow_{\mathrm{x}}$ are defined to be the contextual closures modulo α-conversion of respectively*

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

and

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

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

The reduction $A\langle x{:=}C\rangle \rightarrow_{\mathrm{x}} A$ 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 $y\langle x{:=}C\rangle \rightarrow_{\mathrm{x}} y$ if $x \not\equiv y$ would already be sufficient. The more efficient garbage collection will do no harm however.

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

Lemma 3.4 *$\rightarrow_{\mathrm{x}}$ is strongly normalising, confluent and has unique normalforms.*

Proof: Strong normalisation is shown by defining a map $h : \lambda\mathrm{x} \rightarrow \mathbb{N}$ 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 $A \rightarrow_{\mathrm{x}} B$ then $h(A) > h(B)$.

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

Definition 3.5 *We write $A \in SN_R$ if A is strongly normalizing with respect to the reduction relation R.*

We write $\mathrm{x}(A)$ to denote the x-normalform of A.

For pure terms A, we write $\beta(A)$ to denote the β-normalform of A, if it exists.

We define $\hat{\beta}(A)$ to be the maximal length of a β-reduction path starting with $\mathrm{x}(A)$, if $\mathrm{x}(A) \in SN_\beta$.

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

Proof: We prove by induction on the number of symbols in the sequence $A, B_1, \ldots, B_m$ that $\mathrm{x}(A\langle x_1{:=}B_1\rangle \cdots \langle x_m{:=}B_m\rangle) \equiv \mathrm{x}(A)[\mathrm{x}(B_1)/x_1] \cdots [\mathrm{x}(B_m)/x_m]$.

We distinguish cases according to the structure of A; we only treat some of them:

- $A \equiv x_i$. Then $\mathrm{x}(A\langle x_1{:=}B_1\rangle \cdots \langle x_m{:=}B_m\rangle) \equiv \mathrm{x}(B_i\langle x_{i+1}{:=}B_{i+1}\rangle \cdots \langle x_m{:=}B_m\rangle) \overset{IH}{\equiv}$ $\mathrm{x}(B_i)[\mathrm{x}(B_{i+1})/x_{i+1}] \cdots [\mathrm{x}(B_m)/x_m] \equiv \mathrm{x}(A)[\mathrm{x}(B_1)/x_1] \cdots [\mathrm{x}(B_m)/x_m]$.
- $A \equiv A_1\langle y{:=}A_2\rangle$. Then the number of symbols in the sequence $A, B_1, \ldots, B_m$ is larger than in the sequence $A_1, A_2, B_1, \ldots, B_m$ and the number of symbols in A is bigger than in A_1, A_2 so by the induction hypothesis:
  $\mathrm{x}(A_1\langle y{:=}A_2\rangle\langle x_1{:=}B_1\rangle \cdots \langle x_m{:=}B_m\rangle) \equiv \mathrm{x}(A_1)[\mathrm{x}(A_2)/y][\mathrm{x}(B_1)/x_1] \cdots [\mathrm{x}(B_m)/x_m] \equiv$
  $\mathrm{x}(A_1\langle y{:=}A_2\rangle)[\mathrm{x}(B_1)/x_1] \cdots [\mathrm{x}(B_m)/x_m]$.

□

Lemma 3.7 (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:

1. is immediate and 2. 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 some cases:
   - $A \equiv (\lambda x.A_1)A_2$, $B \equiv A_1\langle x{:=}A_2\rangle$. Then $\mathrm{x}(A) \equiv (\lambda x.\mathrm{x}(A_1))\mathrm{x}(A_2) \to_\beta \mathrm{x}(A_1)[\mathrm{x}(A_2)/x] \overset{3.6}{\equiv} \mathrm{x}(A_1\langle x{:=}A_2\rangle) \equiv \mathrm{x}(B)$.
   - $A \equiv A_1\langle x{:=}A_2\rangle$, $B \equiv A_1\langle x{:=}A_2'\rangle$. Then $\mathrm{x}(A) \overset{3.6}{\equiv} \mathrm{x}(A_1)[\mathrm{x}(A_2)/x] \overset{IH}{\twoheadrightarrow_\beta} \mathrm{x}(A_1)[\mathrm{x}(A_2')/x] \overset{3.6}{\equiv} \mathrm{x}(B)$.

□

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 3.8 (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. We treat the case $A \equiv (\lambda x.A_1)A_2$, $B \equiv A_1[A_2/x]$. Then $A \to_{\mathrm{Beta}} A_1\langle x{:=}A_2\rangle \twoheadrightarrow_{\mathrm{x}} \mathrm{x}(A_1\langle x{:=}A_2\rangle) \overset{\text{Lemma 3.6}}{\equiv} \mathrm{x}(A_1)[\mathrm{x}(A_2)/x] \overset{A \text{ pure}}{\equiv} A_1[A_2/x]$.

□

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

Theorem 3.9 *$\to_{\lambda\mathrm{x}}$ is confluent.*

Proof: If $A \twoheadrightarrow_{\lambda\mathrm{x}} B_1$ and $A \twoheadrightarrow_{\lambda\mathrm{x}} B_2$ then by projection $\mathrm{x}(A) \twoheadrightarrow_\beta \mathrm{x}(B_i)$ so by confluence of $\to_\beta$ there is a pure term C such that $B_i \twoheadrightarrow_\beta C$, now by soundness $\mathrm{x}(B_i) \twoheadrightarrow_{\lambda\mathrm{x}} C$. Then also $B_1 \twoheadrightarrow_{\lambda\mathrm{x}} C$. □

4 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 94] and [Ferreira & Zantema 94].

Definition 4.1 *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

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

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 $i \leq m, n$, $s_1 = t_1, \ldots, s_{i-1} = t_{i-1}, s_i >_{\mathrm{rpo}} t_i$ 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 94], τ is called status function. More complex extensions of $>_{\mathrm{rpo}}$ than multiset or lexicographic are even possible.

Theorem 4.2 (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*

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

Proof: see [Dershowitz 79] or [Ferreira & Zantema 94]. □

5 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; 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 5.1 *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: $A \in \lambda\mathrm{x}^{<\infty}$ if and only if: for all subterms B of A, $\hat{\beta}(B) < \infty$.

Lemma 5.2 $\hat{\beta}((\lambda x.A)B) > \hat{\beta}(A\langle x{:=}B\rangle)$.

Proof: Every $\rightarrow_\beta$-reduction path of length n of $\mathrm{x}(A\langle x{:=}B\rangle)$ can be extended to a reduction path of length $n+1$ of $\mathrm{x}((\lambda x.A)B)$ by prefixing it by $\mathrm{x}((\lambda x.A)B) \equiv (\lambda x.\mathrm{x}(A))\mathrm{x}(B) \rightarrow_\beta \mathrm{x}(A)[\mathrm{x}(B)/x] \equiv \mathrm{x}(A\langle x{:=}B\rangle)$. □

Lemma 5.3 *If $A \in \lambda x^{<\infty}$ and $A \to_{\lambda x} A'$ then $A' \in \lambda x^{<\infty}$.*

Proof: Induction on the structure of A. □

Definition 5.4 *We define the set of labelled terms λ^l by the following abstract syntax:*

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

where n ranges over the natural numbers.

We define by induction on the structure of terms a translation $\mathcal{T} : \lambda x^{<\infty} \to \Lambda^l$:

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

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

Definition 5.5 *We define a TRS λ^l using as terms the set Λ^l and having the reduction relation $\to_l$ defined by*

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

Note that $\to_l$ is not confluent; for our purposes this is no problem since $\to_l$ is only designed to be useful for proving strong normalisation. The last two rules are called Decr in [Zantema 94] and are necessary to decrease the labels of applications and substitutions if inside of them a $\to_{\text{Beta}}$-reduction is performed. Note that in the presence of the Decr rules we could also have $(\lambda A) \cdot_{n+1} B \to_l A\langle B\rangle_n$ for all n instead of $(\lambda A) \cdot_m B \to_l A\langle B\rangle_n$ for all $m > n$.

Lemma 5.6 *The relation $\to_l$ is a subrelation of some recursive path order. (That is, there is a precedence relation $\rhd$ such that for all $A, B \in \lambda^l$, if $A \to_l B$, then $A >_{\text{rpo}} B$, where $>_{\text{rpo}}$ is the rpo ordering induced by $\rhd$.)*

Proof: Define the precedence $\rhd$ by

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

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

Corollary 5.7 *$\to_l$ is SN.*

Proof: By Theorem 4.2, $>_{\text{rpo}}$ of Lemma 5.6 is strongly normalising, hence by Lemma 5.6 $\to_l$ is strongly normalising. □

Lemma 5.8 *If $A \in \lambda x^{<\infty}$ and $A \to_{\lambda x} A'$ then $\mathcal{T}(A) \twoheadrightarrow_l^+ \mathcal{T}(A')$.*

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

- $A \equiv (\lambda x.A_1)A_2 \to_{\text{Beta}} A_1\langle x:=A_2\rangle \equiv A'$.

  Then $\mathcal{T}(A) \equiv (\lambda\mathcal{T}(A_1)) \cdot_m \mathcal{T}(A_2) \to_l \mathcal{T}(A_1)\langle x:=T(A_2)\rangle_n \equiv \mathcal{T}(A')$ where $m = \hat{\beta}(A)$; $n = \hat{\beta}(A')$; note that $m > n$ by Lemma 5.2.

- $A \equiv (A_1A_2)\langle x:=A_3\rangle \to_{\mathrm{x}} (A_1\langle x:=A_3\rangle)(A_2\langle x:=A_3\rangle) \equiv A'$.

  Then $\mathcal{T}(A) \equiv (\mathcal{T}(A_1) \cdot_m \mathcal{T}(A_2))\langle\mathcal{T}(A_3)\rangle_n \to_l (\mathcal{T}(A_1)\langle\mathcal{T}(A_3)\rangle_p \cdot_n (\mathcal{T}(A_2)\langle\mathcal{T}(A_3)\rangle_q \equiv \mathcal{T}(A')$, 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 $n \geq q$.

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

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

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

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

□

Corrollary 5.9 (PSN) *1.* $A \in SN_{\lambda\mathrm{x}} \iff A \in \lambda\mathrm{x}^{<\infty}$

2. $\lambda\mathrm{x}$ *preserves strong normalisation*

6 $\lambda\upsilon$, $\lambda\mathrm{s}$ 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 $\lambda\mathrm{s}$ 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.

6.1 The calculi $\lambda\upsilon$ and $\lambda\mathrm{s}$

Definition 6.1 *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 the set* $\{1,2,3,4,\ldots\}$.

The reduction relation $\to_{\lambda\upsilon}$ *is the union of* $\to_{\upsilon\text{Beta}}$ *and* $\to_\upsilon$ *which are defined by*

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

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 substituting 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.)

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

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

Definition 6.2 *Terms and substitutions of* λs *are defined by the following abstract syntaxes:*

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

where n, i *range over the set* $\{1,2,3,4,\ldots\}$ *and* j *ranges over* $\{0,1,2,3,\ldots\}$.

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

$$\begin{aligned}
(\lambda a)b &\to_{\mathrm{sBeta}} a\sigma^1 b \\
(\lambda a)\sigma^i b &\to_{\mathrm{s}} \lambda(a\sigma^{i+1} b) \\
(a_1 a_2)\sigma^i b &\to_{\mathrm{s}} (a_1\sigma^i b)(a_1\sigma^i b) \\
\underline{n}\sigma^i b &\to_{\mathrm{s}} \begin{cases} \underline{n-1} & \text{if} \quad n > i \\ \phi^i_0(b) & \text{if} \quad n = i \\ \underline{n} & \text{if} \quad n < i \end{cases} \\
\phi^i_k(\lambda a) &\to_{\mathrm{s}} \lambda(\phi^i_{k+1} a) \\
\phi^i_k(a_1 a_2) &\to_{\mathrm{s}} (\phi^i_k a_1)(\phi^i_k a_2) \\
\phi^i_k \underline{n} &\to_{\mathrm{s}} \begin{cases} \underline{n+i-1} & \text{if} \quad n > k \\ \underline{n} & \text{if} \quad n \le k \end{cases}
\end{aligned}$$

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 $\lambda\upsilon$.

$$\begin{aligned}
(\lambda(\lambda(\underline{12})))(\underline{11}) &\to_{\mathrm{sBeta}} (\lambda(\underline{12}))\sigma^1(\underline{11}) \\
&\to_{\mathrm{s}} \lambda((\underline{12})\sigma^2(\underline{11})) \\
&\to_{\mathrm{s}} \lambda(\underline{1}\sigma^2(\underline{11}))(\underline{2}\sigma^2(\underline{11}))) \\
&\to_{\mathrm{s}} \lambda(\underline{1}(\underline{2}\sigma^2(\underline{11}))) \\
&\to_{\mathrm{s}} \lambda(\underline{1}\phi^2_0(\underline{11})) \\
&\to_{\mathrm{s}} \lambda(\underline{1}(\phi^2_0(\underline{1})\phi^2_0(\underline{1}))) \\
&\to\!\!\to_{\mathrm{s}} \lambda(\underline{1}(\underline{22}))
\end{aligned}$$

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 updatefunction generated whereas in $\underline{n+1}[\Uparrow(s)] \to_\upsilon \underline{n}[s][\uparrow]$ 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 $\lambda\upsilon$ has PSN by contradicting the existence of a minimal infinite $\lambda\upsilon$-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 $\lambda\upsilon$ 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 $\lambda\upsilon$ and λs we have the usual properties such as SN, CR, UN for $\to_\upsilon$ respectively $\to_s$, substitution lemma, projection lemma, soundness lemma and confluence for $\to_{\lambda\upsilon}$ respectively $\to_{\lambda s}$. We denote the $\to_\upsilon$-normal form respectively $\to_s$-normal form of a term b by $\upsilon(b)$ respectively $s(b)$. Note that a substitution of $\lambda\upsilon$ is of the form $\Uparrow^n(b/)$ or $\Uparrow^n(\uparrow)$ for some n.

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

Definition 6.3

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

Lemma 6.4 *1. $\lambda\upsilon^{<\infty}$ is closed under $\to_{\lambda\upsilon}$-reduction*

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

Definition 6.5 *1. $\mathcal{T}_\upsilon : \lambda\upsilon^{<\infty} \longrightarrow \Lambda^l$ is defined by*

$$\begin{aligned} \mathcal{T}_\upsilon(\underline{n}) &= * \\ \mathcal{T}_\upsilon(ab) &= \mathcal{T}_\upsilon(a) \cdot_p \mathcal{T}_\upsilon(b) && \text{where } p = \hat{\beta}(ab) \\ \mathcal{T}_\upsilon(\lambda a) &= \lambda\mathcal{T}_\upsilon(a) \\ \mathcal{T}_\upsilon(a[\Uparrow^n(b/)]) &= \mathcal{T}_\upsilon(a)\langle\mathcal{T}_\upsilon(b)\rangle_p && \text{where } p = \hat{\beta}(a[\Uparrow^n(b/)]) \\ \mathcal{T}_\upsilon(a[\Uparrow^n(\uparrow)]) &= \mathcal{T}_\upsilon(a) \end{aligned}$$

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

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

Lemma 6.6 *1. If $a \in \lambda\upsilon^{<\infty}$ and $a \to_\upsilon b$ then $\mathcal{T}_\upsilon(a) \twoheadrightarrow_l \mathcal{T}_\upsilon(b)$*

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

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 $a \to_{s\text{Beta}} b$ then $\mathcal{T}_s(a) \twoheadrightarrow_l^+ \mathcal{T}_s(b)$

Proof: Induction on the structure of a. □

Theorem 6.7 *1. $a \in SN_{\lambda\upsilon} \iff a \in \lambda\upsilon^{<\infty}$*

2. $\rightarrow_{\lambda v}$ *has PSN*

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

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

Proof:

1. $\Rightarrow$ by projection; $\Leftarrow$: since $\rightarrow_v$ is SN, any infinite $\rightarrow_{\lambda v}$-reduction must contain infinitely many $\rightarrow_{v\text{Beta}}$-steps. Therefore an infinite reduction of a pure term which is SN for $\rightarrow_\beta$ translates by $\mathcal{T}_v$ into an infinite $\rightarrow_I$-reduction which is impossible by 5.7.

2. follows from 1.

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

□

6.2 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:

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

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 needed) doesn't have PSN. We give an infinite derivation starting from the term $(\lambda y.(\lambda y'.x)((\lambda y.a)b))((\lambda y.a)b)$. Note that this term is even simpler than the term used in [Melliès 95].

First we define substitutions α_i for $m \in \mathbb{N}$ by

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

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.)

$$\begin{aligned} (\lambda y.(\lambda y'.x)((\lambda y.a)b))((\lambda y.a)b) &\twoheadrightarrow x\langle y' := (\lambda y.a)b \rangle \langle y := (\lambda y.a)b \rangle \\ &\rightarrow x\langle y' := ((\lambda y.a)b)\langle y := (\lambda y.a)b \rangle \rangle \\ &\twoheadrightarrow x\langle y' := (\lambda y.a\langle y := (\lambda y.a)b \rangle)(b\langle y := (\lambda y.a)b \rangle) \rangle \\ &\equiv x\langle y' := (\lambda y.a\alpha_0)(b\alpha_0) \rangle \\ &\rightarrow x\langle y' := a\alpha_0\langle y := b\alpha_0 \rangle \rangle \\ &\equiv x\langle y' := a\alpha_0\alpha_1 \rangle \end{aligned}$$

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

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

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

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

Recall that we proved PSN by showing that every term of which the x-normal form of any of its subterms is in SN_β, is also SN for $\to_{\mathrm{Beta}} \cup \to_{\mathrm{x}}$. 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-normalform x (and is also SN for λx-reduction), but it has Ω as a subterm of a reduct if composition is allowed.

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:

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

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 $\to_{\lambda\mathrm{x}}$-reduct. Hence allowing to create $b\langle y:=c\rangle$ at this point will not spoil PSN. We strongly believe that adding this reduction rule does not spoil SN (i.e. λx extended with the rule $\to_{\mathrm{x}'}$ has PSN), but we have not been able to prove it.

7 Proof of PSN using labelled trees

In this section we outline a proof of the Preservation of Strong Normalization property, 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 $\mathsf{Tree}^\star$, 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 $\mathsf{Tree}^\star$), a reduction relation $\Rightarrow$ is defined.

Definition 7.1 *The relation* $\Rightarrow$ *on* $\mathsf{Tree}^\star$ *is defined as follows.*

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

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

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

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 issue of PSN for explicit substituion.

Theorem 7.2 ([Klop 92],[Dershowitz 79]) *The relation $\Rightarrow\!\!\rhd^+$ is well-founded on* Tree *(the set of trees without markers).*

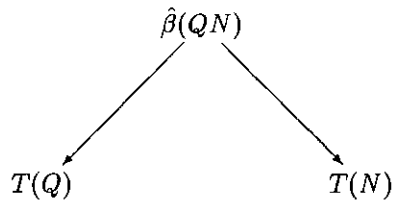
To prove PSN for the calculus λx, we now proceed by defining a reduction preserving mapping T from $\lambda x^{<\infty}$ to Tree: if $M \to_x N$, then $M \Rightarrow\!\!\rhd^* N$ and if $M \to_{\text{Beta}} N$, then $M \Rightarrow\!\!\rhd^+ N$. Hence, using the fact that $\to_x$ is strongly normalizing, we can again conclude that every $M \in \lambda x^{<\infty}$ is strongly normalizing 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 7.3 *For $M \in \lambda x^{<\infty}$, we define the tree $T(M)$ by induction on the length of M as follows.*

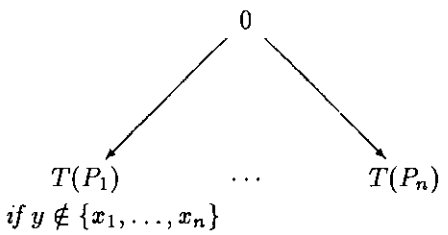
$$T(x) = 0,$$

$$T(QN) =$$



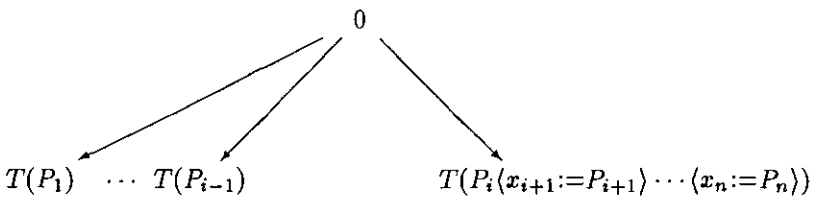
$$T(\lambda y.N) = T(N)$$

$$T(y\overline{\langle x{:=}P\rangle}) =$$

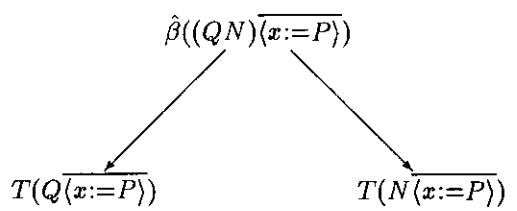


if $y \notin \{x_1, \ldots, x_n\}$

$$T(x_i\overline{\langle x{:=}P\rangle}) =$$



$$T((QN)\overline{\langle x{:=}P\rangle}) =$$



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

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 on which the induction should be done). We just outline the proofs.

Lemma 7.4 *For $M \in \lambda x^{<\infty}$, if $M \to_x N$, then $T(M) \Rightarrow^* T(N)$.*

Proof: By induction on the length of M, distinguishing subcases according to the structure of M. Note that we need Lemma 5.3 to make sure that $N \in \lambda x^{<\infty}$ and hence that $T(N)$ is well-defined. □

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

Lemma 7.5 *For $N\overline{\langle x{:=}P\rangle} \in \lambda x^{<\infty}$, $T(N\overline{\langle x{:=}P\rangle}) \Rightarrow^* T(N)$.*

Proof: By induction on the length of N. □

Lemma 7.6 *For $((\lambda y.N)Q)\overline{\langle x{:=}P\rangle} \in \lambda x^{<\infty}$, $T(((\lambda y.N)Q)\overline{\langle x{:=}P\rangle}) \Rightarrow^+ T(N\langle y{:=}Q\rangle\overline{\langle x{:=}P\rangle})$.*

Proof: By induction on the length of N, using Lemma 7.5. 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 7.7 *The calculus λx has the PSN property.*

8 Conclusions

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

- determine a suitable set contained in the set of strongly normalising 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 normalising 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 combinator 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.

9 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.

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

[Bloo & Rose 95] Bloo, R., and Rose, K. H., Preservation of Strong Normalisation 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 Normalisation, to appear in: *Proceedings of RTA'96.*

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

[Ferreira & Zantema 94] Ferreira, M.C.F. and Zantema, H., Well-foundedness of Term Orderings, Technical Report UU-CS-1994-46, Utrecht University.

[BBLR 95] Benaissa, Z.E.A., Briaud, D., Lescanne, P. and Rouyer-Degli, J., $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation, 1995, accepted for publication in *Journal of Functional Programming.*

[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.

[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 94] Zantema, H., Termination of Term Rewriting by Semantic Labelling, *Fundamenta Informaticae*, Vol. **24** (1,2), pp. 89–106, 1995

Computing Science Reports

Department of Mathematics and Computing Science
Eindhoven University of Technology

In this series appeared:

93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.
93/05	J.C.M. Baeten C. Verhoef	A congruence theorem for structured operational semantics with predicates, p. 18.
93/06	J.P. Veltkamp	On the unavoidability of metastable behaviour, p. 29
93/07	P.D. Moerland	Exercises in Multiprogramming, p. 97
93/08	J. Verhoosel	A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
93/09	K.M. van Hee	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
93/10	K.M. van Hee	Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/11	K.M. van Hee	Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach Part V: Specification Language, p. 89.
93/14	J.C.M. Baeten J.A. Bergstra	On Sequential Composition, Action Prefixes and Process Prefix, p. 21.
93/15	J.C.M. Baeten J.A. Bergstra R.N. Bol	A Real-Time Process Logic, p. 31.
93/16	H. Schepers J. Hooman	A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27
93/17	D. Alstein P. van der Stok	Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.
93/18	C. Verhoef	A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.
93/19	G-J. Houben	The Design of an Online Help Facility for ExSpect, p.21.
93/20	F.S. de Boer	A Process Algebra of Concurrent Constraint Programming, p. 15.
93/21	M. Codish D. Dams G. Filé M. Bruynooghe	Freeness Analysis for Logic Programs - And Correctness, p. 24
93/22	E. Poll	A Typechecker for Bijective Pure Type Systems, p. 28.
93/23	E. de Kogel	Relational Algebra and Equational Proofs, p. 23.
93/24	E. Poll and Paula Severi	Pure Type Systems with Definitions, p. 38.
93/25	H. Schepers and R. Gerth	A Compositional Proof Theory for Fault Tolerant Real-Time Distributed Systems, p. 31.
93/26	W.M.P. van der Aalst	Multi-dimensional Petri nets, p. 25.
93/27	T. Kloks and D. Kratsch	Finding all minimal separators of a graph, p. 11.
93/28	F. Kamareddine and R. Nederpelt	A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.
93/29	R. Post and P. De Bra	GOLD, a Graph Oriented Language for Databases, p. 42.
93/30	J. Deogun T. Kloks D. Kratsch H. Müller	On Vertex Ranking for Permutation and Other Graphs, p. 11.

93/31	W. Körver	Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.
93/32	H. ten Eikelder and H. van Geldrop	On the Correctness of some Algorithms to generate Finite Automata for Regular Expressions, p. 17.
93/33	L. Loyens and J. Moonen	ILIAS, a sequential language for parallel matrix computations, p. 20.
93/34	J.C.M. Baeten and J.A. Bergstra	Real Time Process Algebra with Infinitesimals, p.39.
93/35	W. Ferrer and P. Severi	Abstract Reduction and Topology, p. 28.
93/36	J.C.M. Baeten and J.A. Bergstra	Non Interleaving Process Algebra, p. 17.
93/37	J. Brunekreef J-P. Katoen R. Koymans S. Mauw	Design and Analysis of Dynamic Leader Election Protocols in Broadcast Networks, p. 73.
93/38	C. Verhoef	A general conservative extension theorem in process algebra, p. 17.
93/39	W.P.M. Nuijten E.H.L. Aarts D.A.A. van Erp Taalman Kip K.M. van Hee	Job Shop Scheduling by Constraint Satisfaction, p. 22.
93/40	P.D.V. van der Stok M.M.M.P.J. Claessen D. Alstein	A Hierarchical Membership Protocol for Synchronous Distributed Systems, p. 43.
93/41	A. Bijlsma	Temporal operators viewed as predicate transformers, p. 11.
93/42	P.M.P. Rambags	Automatic Verification of Regular Protocols in P/T Nets, p. 23.
93/43	B.W. Watson	A taxomomy of finite automata construction algorithms, p. 87.
93/44	B.W. Watson	A taxonomy of finite automata minimization algorithms, p. 23.
93/45	E.J. Luit J.M.M. Martin	A precise clock synchronization protocol,p.
93/46	T. Kloks D. Kratsch J. Spinrad	Treewidth and Patwidth of Cocomparability graphs of Bounded Dimension, p. 14.
93/47	W. v.d. Aalst P. De Bra G.J. Houben Y. Kornatzky	Browsing Semantics in the "Tower" Model, p. 19.
93/48	R. Gerth	Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.
94/01	P. America M. van der Kammen R.P. Nederpelt O.S. van Roosmalen H.C.M. de Swart	The object-oriented paradigm, p. 28.
94/02	F. Kamareddine R.P. Nederpelt	Canonical typing and Π-conversion, p. 51.
94/03	L.B. Hartman K.M. van Hee	Application of Marcov Decision Processe to Search Problems, p. 21.
94/04	J.C.M. Baeten J.A. Bergstra	Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18.
94/05	P. Zhou J. Hooman	Formal Specification and Compositional Verification of an Atomic Broadcast Protocol, p. 22.
94/06	T. Basten T. Kunz J. Black M. Coffin D. Taylor	Time and the Order of Abstract Events in Distributed Computations, p. 29.
94/07	K.R. Apt R. Bol	Logic Programming and Negation: A Survey, p. 62.
94/08	O.S. van Roosmalen	A Hierarchical Diagrammatic Representation of Class Structure, p. 22.
94/09	J.C.M. Baeten J.A. Bergstra	Process Algebra with Partial Choice, p. 16.

94/10	T. verhoeff	The testing Paradigm Applied to Network Structure. p. 31.
94/11	J. Peleska C. Huizing C. Petersohn	A Comparison of Ward & Mellor's Transformation Schema with State- & Activitycharts, p. 30.
94/12	T. Kloks D. Kratsch H. Müller	Dominoes, p. 14.
94/13	R. Seljée	A New Method for Integrity Constraint checking in Deductive Databases, p. 34.
94/14	W. Peremans	Ups and Downs of Type Theory, p. 9.
94/15	R.J.M. Vaessens E.H.L. Aarts J.K. Lenstra	Job Shop Scheduling by Local Search, p. 21.
94/16	R.C. Backhouse H. Doornbos	Mathematical Induction Made Calculational, p. 36.
94/17	S. Mauw M.A. Reniers	An Algebraic Semantics of Basic Message Sequence Charts, p. 9.
94/18	F. Kamareddine R. Nederpelt	Refining Reduction in the Lambda Calculus, p. 15.
94/19	B.W. Watson	The performance of single-keyword and multiple-keyword pattern matching algorithms, p. 46.
94/20	R. Bloo F. Kamareddine R. Nederpelt	Beyond β-Reduction in Church's $\lambda\rightarrow$, p. 22.
94/21	B.W. Watson	An introduction to the Fire engine: A C++ toolkit for Finite automata and Regular Expressions.
94/22	B.W. Watson	The design and implementation of the FIRE engine: A C++ toolkit for Finite automata and regular Expressions.
94/23	S. Mauw and M.A. Reniers	An algebraic semantics of Message Sequence Charts, p. 43.
94/24	D. Dams O. Grumberg R. Gerth	Abstract Interpretation of Reactive Systems: Abstractions Preserving $\forall$CTL*, $\exists$CTL* and CTL*, p. 28.
94/25	T. Kloks	$K_{1,3}$-free and W_4-free graphs, p. 10.
94/26	R.R. Hoogerwoord	On the foundations of functional programming: a programmer's point of view, p. 54.
94/27	S. Mauw and H. Mulder	Regularity of BPA-Systems is Decidable, p. 14.
94/28	C.W.A.M. van Overveld M. Verhoeven	Stars or Stripes: a comparative study of finite and transfinite techniques for surface modelling, p. 20.
94/29	J. Hooman	Correctness of Real Time Systems by Construction, p. 22.
94/30	J.C.M. Baeten J.A. Bergstra Gh. Ştefanescu	Process Algebra with Feedback, p. 22.
94/31	B.W. Watson R.E. Watson	A Boyer-Moore type algorithm for regular expression pattern matching, p. 22.
94/32	J.J. Vereijken	Fischer's Protocol in Timed Process Algebra, p. 38.
94/33	T. Laan	A formalization of the Ramified Type Theory, p.40.
94/34	R. Bloo F. Kamareddine R. Nederpelt	The Barendregt Cube with Definitions and Generalised Reduction, p. 37.
94/35	J.C.M. Baeten S. Mauw	Delayed choice: an operator for joining Message Sequence Charts, p. 15.
94/36	F. Kamareddine R. Nederpelt	Canonical typing and Π-conversion in the Barendregt Cube, p. 19.
94/37	T. Basten R. Bol M. Voorhoeve	Simulating and Analyzing Railway Interlockings in ExSpect, p. 30.
94/38	A. Bijlsma C.S. Scholten	Point-free substitution, p. 10.

94/39	A. Blokhuis T. Kloks		On the equivalence covering number of splitgraphs, p. 4.
94/40	D. Alstein		Distributed Consensus and Hard Real-Time Systems, p. 34.
94/41	T. Kloks D. Kratsch		Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph, p. 6.
94/42	J. Engelfriet J.J. Vereijken		Concatenation of Graphs, p. 7.
94/43	R.C. Backhouse M. Bijsterveld		Category Theory as Coherently Constructive Lattice Theory: An Illustration, p. 35.
94/44	E. Brinksma R. Gerth W. Janssen S. Katz M. Poel C. Rump	J. Davies S. Graf B. Jonsson G. Lowe A. Pnueli J. Zwiers	Verifying Sequentially Consistent Memory, p. 160
94/45	G.J. Houben		Tutorial voor de ExSpect-bibliotheek voor "Administratieve Logistiek", p. 43.
94/46	R. Bloo F. Kamareddine R. Nederpelt		The λ-cube with classes of terms modulo conversion, p. 16.
94/47	R. Bloo F. Kamareddine R. Nederpelt		On Π-conversion in Type Theory, p. 12.
94/48	Mathematics of Program Construction Group		Fixed-Point Calculus, p. 11.
94/49	J.C.M. Baeten J.A. Bergstra		Process Algebra with Propositional Signals, p. 25.
94/50	H. Geuvers		A short and flexible proof of Strong Normalazation for the Calculus of Constructions, p. 27.
94/51	T. Kloks D. Kratsch H. Müller		Listing simplicial vertices and recognizing diamond-free graphs, p. 4.
94/52	W. Penczek R. Kuiper		Traces and Logic, p. 81
94/53	R. Gerth R. Kuiper D. Peled W. Penczek		A Partial Order Approach to Branching Time Logic Model Checking, p. 20.
95/01	J.J. Lukkien		The Construction of a small CommunicationLibrary, p.16.
95/02	M. Bezem R. Bol J.F. Groote		Formalizing Process Algebraic Verifications in the Calculus of Constructions, p.49.
95/03	J.C.M. Baeten C. Verhoef		Concrete process algebra, p. 134.
95/04	J. Hidders		An Isotopic Invariant for Planar Drawings of Connected Planar Graphs, p. 9.
95/05	P. Severi		A Type Inference Algorithm for Pure Type Systems, p.20.
95/06	T.W.M. Vossen M.G.A. Verhoeven H.M.M. ten Eikelder E.H.L. Aarts		A Quantitative Analysis of Iterated Local Search, p.23.
95/07	G.A.M. de Bruyn O.S. van Roosmalen		Drawing Execution Graphs by Parsing, p. 10.
95/08	R. Bloo		Preservation of Strong Normalisation for Explicit Substitution, p. 12.
95/09	J.C.M. Baeten J.A. Bergstra		Discrete Time Process Algebra, p. 20
95/10	R.C. Backhouse R. Verhoeven O. Weber		MathJpad: A System for On-Line Prepararation of Mathematical Documents, p. 15

95/11	R. Seljée	Deductive Database Systems and integrity constraint checking, p. 36.
95/12	S. Mauw and M. Reniers	Empty Interworkings and Refinement Semantics of Interworkings Revised, p. 19.
95/13	B.W. Watson and G. Zwaan	A taxonomy of sublinear multiple keyword pattern matching algorithms, p. 26.
95/14	A. Ponse, C. Verhoef, S.F.M. Vlijmen (eds.)	De proceedings: ACP'95, p.
95/15	P. Niebert and W. Penczek	On the Connection of Partial Order Logics and Partial Order Reduction Methods, p. 12.
95/16	D. Dams, O. Grumberg, R. Gerth	Abstract Interpretation of Reactive Systems: Preservation of CTL*, p. 27.
95/17	S. Mauw and E.A. van der Meulen	Specification of tools for Message Sequence Charts, p. 36.
95/18	F. Kamareddine and T. Laan	A Reflection on Russell's Ramified Types and Kripke's Hierarchy of Truths, p. 14.
95/19	J.C.M. Baeten and J.A. Bergstra	Discrete Time Process Algebra with Abstraction, p. 15.
95/20	F. van Raamsdonk and P. Severi	On Normalisation, p. 33.
95/21	A. van Deursen	Axiomatizing Early and Late Input by Variable Elimination, p. 44.
95/22	B. Arnold, A. v. Deursen, M. Res	An Algebraic Specification of a Language for Describing Financial Products, p. 11.
95/23	W.M.P. van der Aalst	Petri net based scheduling, p. 20.
95/24	F.P.M. Dignum, W.P.M. Nuijten, L.M.A. Janssen	Solving a Time Tabling Problem by Constraint Satisfaction, p. 14.
95/25	L. Feijs	Synchronous Sequence Charts In Action, p. 36.
95/26	W.M.P. van der Aalst	A Class of Petri nets for modeling and analyzing business processes, p. 24.
95/27	P.D.V. van der Stok, J. van der Wal	Proceedings of the Real-Time Database Workshop, p. 106.
95/28	W. Fokkink, C. Verhoef	A Conservative Look at term Deduction Systems with Variable Binding, p. 29.
95/29	H. Jurjus	On Nesting of a Nonmonotonic Conditional, p. 14
95/30	J. Hidders, C. Hoskens, J. Paredaens	The Formal Model of a Pattern Browsing Technique, p.24.
95/31	P. Kelb, D. Dams and R. Gerth	Practical Symbolic Model Checking of the full μ-calculus using Compositional Abstractions, p. 17.
95/32	W.M.P. van der Aalst	Handboek simulatie, p. 51.
95/33	J. Engelfriet and JJ. Vereijken	Context-Free Graph Grammars and Concatenation of Graphs, p. 35.
95/34	J. Zwanenburg	Record concatenation with intersection types, p. 46.
95/35	T. Basten and M. Voorhoeve	An algebraic semantics for hierarchical P/T Nets, p. 32.
96/01	M. Voorhoeve and T. Basten	Process Algebra with Autonomous Actions, p. 12.
96/02	P. de Bra and A. Aerts	Multi-User Publishing in the Web: DreSS, A Document Repository Service Station, p. 12
96/03	W.M.P. van der Aalst	Parallel Computation of Reachable Dead States in a Free-choice Petri Net, p. 26.
96/04	S. Mauw	Example specifications in phi-SDL.
96/05	T. Basten and W.M.P. v.d. Aalst	A Process-Algebraic Approach to Life-Cycle Inheritance Inheritance = Encapsulation + Abstraction, p. 15.
96/06	W.M.P. van der Aalst and T. Basten	Life-Cycle Inheritance A Petri-Net-Based Approach, p. 18.
96/07	M. Voorhoeve	Structural Petri Net Equivalence, p. 16.
96/08	A.T.M. Aerts, P.M.E. De Bra, J.T. de Munk	OODB Support for WWW Applications: Disclosing the internal structure of Hyperdocuments, p. 14.
96/09	F. Dignum, H. Weigand, E. Verharen	A Formal Specification of Deadlines using Dynamic Deontic Logic, p. 18.