Explicit substitution : on the edge of strong normalisation

Citation for published version (APA):

Document status and date:
Published: 01/01/1996

Publisher Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Explicit Substitution: on the Edge of Strong Normalisation

by

R. Bloo and H. Geuvers 96/10

ISSN 0926-4515

All rights reserved
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 $\beta$-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: $\lambda\nu$ of Lescanne et al., $\lambda\nu$ of Kamareddine and Izsak and $\lambda\nu$ 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. Mellies has shown that simply typable terms can have infinite reduction paths in $\lambda\sigma$ ([Mellies 95]). Several people (see [BBLR 95],[Blo & Rose 95],[Bloo 95],[Kamareddine & Izsak 95]) have succeeded in giving calculi of explicit substitutions which have the nice property that every term which is strongly normalising for $\beta$-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 ordering. 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 $\ast$. The recursive path order $\succ_{\text{RPO}}$ on this labelled calculus is strongly normalising (or: terminating).

Then we take a look at the explicit substitution calculus $\lambda\sigma\nu$. The $\beta$-reduction is here split up into a reduction step $\rightarrow_{\beta\text{eta}}$ (contracting the $\beta$-redex and creating an explicit substitution) and reduction steps $\rightarrow_{\nu}$ (moving the explicit substitutions through the term to perform the substitution). It is relatively easy (as usual in these calculi) to observe that $\rightarrow_{\nu}$ 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 $\lambda\sigma\nu$ of which the substitution normal form of all of its subterms is $\beta$-SN; we call this set $\lambda\sigma\nu^{<\infty}$. (The substitution normal form of a term $M$ is obtained by evaluating all the explicit substitutions in $M$, not contracting any $\beta$-redexes. That is, we take the $\rightarrow_{\nu}$-normal-form of $M$.) We then define a translation $T$ from $\lambda\sigma\nu^{<\infty}$ into the set of labelled terms. This translation $T$ is reduction preserving.
in the sense that, if $M \rightarrow_x N$, then $T(M) \triangleright_{po} T(N)$ or $T(M) = T(N)$ and if $M \rightarrow_{Beta} N$, then $T(M) \triangleright_{po} T(N)$. Hence, using the fact that $\rightarrow_\beta$ is strongly normalizing, we conclude that every $M \in \lambda x^{<\infty}$ is strongly normalizing. So, $\lambda x$ has the PSN property (because every $\lambda$-term that is $\beta$-strongly-normalizing is an element of $\lambda x^{<\infty}$).

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 $\lambda x^{<\infty}$ 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 $\lambda x$.

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 $\lambda v$ of Lescanne et al. and $\lambda s$ of Kamareddine and Rios.

### 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.

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

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

$$
x \equiv x,$$

$$
P \equiv N \text{ and } Q \equiv M \Rightarrow PQ \equiv NM,$$

$$
P \equiv Q, y \notin FV(Q) \setminus \{x\} \Rightarrow \lambda z.P \equiv \lambda y.Q[y/x].$$

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 $\lambda x$ of explicit substitutions, two extensions have to be made. The first is extending the terms with substitutions:

**Definition 3.2** The set of terms $\lambda x$ is defined by the following abstract syntax:

$$A ::= x \mid AA \mid \lambda z.A \mid A(x:=A)$$

Where $x$ denotes an arbitrary variable.

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

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

2
\( M(x:=P)(N/x) \equiv M(x:=P[N/x]) \)

\( \alpha \)-equivalence is defined on \( \lambda x \)-terms as for \( \lambda \)-terms but with the extra clause that
\[
M \equiv N, P \equiv Q, y \notin \text{FV}(Q) \setminus \{x\} \Rightarrow P(x:=M) \equiv Q[y/x](y:=N)
\]

A \( \lambda x \) is called pure if \( A \in \lambda \), i.e., \( A \) does not contain any substitution \( (x:=B) \).

The second is refining the notion of \( \beta \)-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. \( \rightarrow_{\text{Beta}} \) is for the creation of a substitution out of a \( \beta \)-redex; \( \rightarrow_\chi \) 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 \( \rightarrow_{\text{Beta}} \) and \( \rightarrow_\chi \) are defined to be the contextual closures modulo \( \alpha \)-conversion of respectively
\[
(\lambda x.A)B \rightarrow_{\text{Beta}} A(x:=B)
\]
and
\[
(AB)(x:=C) \rightarrow_\chi (A(x:=C))(B(x:=C))
\]

\[
(\lambda y.A)(x:=C) \rightarrow_\chi \lambda y.A(x:=C) \text{ if } x \notin y \text{ and } y \notin \text{FV}(C)
\]

\[
z(x:=C) \rightarrow_\chi C
\]

\[
A(x:=C) \rightarrow_\chi A \text{ if } x \notin \text{FV}(C)
\]

The explicit substitution reduction relation \( \rightarrow_{\lambda x} \) is the union of \( \rightarrow_{\text{Beta}} \) and \( \rightarrow_\chi \).

The reduction \( A(x:=C) \rightarrow_\chi A \) if \( x \notin \text{FV}(A) \) is also called garbage collection. Since we consider terms modulo \( \alpha \)-equality, substitutions can always be distributed to variables, hence the rule \( y(x:=C) \rightarrow_\chi y \) if \( x \neq y \) would already be sufficient. The more efficient garbage collection will do no harm however.

The reduction relation \( \rightarrow_\chi \) is called the substitution calculus. It has nice properties:

**Lemma 3.4** \( \rightarrow_\chi \) is strongly normalising, confluent and has unique normalforms.

**Proof:** Strong normalisation is shown by defining a map \( h : \lambda x \rightarrow N \) which decreases on \( x \)-reduction; define
\[
h(x) = 1
\]
\[
h(AB) = h(A) + h(B) + 1
\]
\[
h(\lambda z.A) = h(A) + 1
\]
\[
h(A(x:=B)) = h(A) \cdot (h(B) + 1)
\]
then by induction on the structure of \( A \): if \( A \rightarrow_\chi B \) then \( h(A) > h(B) \).

To prove confluence, it is now sufficient to show weak confluence which is easy. \( \Box \)

**Definition 3.5** We write \( A \in \text{SN}_R \) if \( A \) is strongly normalizing with respect to the reduction relation \( R \).

We write \( x(A) \) to denote the \( x \)-normalform of \( A \).

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

We define \( \hat{\beta}(A) \) to be the maximal length of a \( \beta \)-reduction path starting with \( x(A) \), if \( x(A) \in \text{SN}_\beta \).
Note that for $A \in \lambda x, x(A)$ is pure.

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

**Lemma 3.6 (substitution)** For all terms $A, B$: $x(A(x:=B)) \equiv x(A)[x(B)/x]$.

**Proof:** We prove by induction on the number of symbols in the sequence $A, B, A_1, \ldots, B_m$ that $x(A(x_1:=B_1) \cdots (x_m:=B_m)) \equiv x(A)[x(B_1)/x_1] \cdots [x(B_m)/x_m]$.

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

- $A \equiv x_i$. Then $x(A(x_1:=B_1) \cdots (x_m:=B_m)) \equiv x(B_i[x(x_{i+1}:=B_{i+1}) \cdots (x_m:=B_m)])$.

- $A \equiv A_1(y:=A_2)$. 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:

  $x(A_1(y:=A_2)) \equiv x(A_1)[x(A_2)/y]$.

**Lemma 3.7 (projection)** For all terms $A, B$:

1. if $A \rightarrow_x B$ then $x(A) \equiv x(B)$
2. if $A \rightarrow_{\beta}$ $B$ then $x(A) \rightarrow_{\beta} x(B)$

**Proof:**

1. is immediate and 2. is by induction on the structure of $A$. Note that if $N \rightarrow_{\beta} N'$ then $M[N/z] \rightarrow_{\beta} M[N'/z]$. We treat some cases:

   - $A \equiv (\lambda x, A_1)A_2$, $B \equiv A_1(x:=A_2)$. Then $x(A) \equiv (\lambda x, x(A_1))x(A_2) \rightarrow_{\beta} x(A_1)[x(A_2)/x]$.

   - $A \equiv A_1(x:=A_2)$, $B \equiv A_1(x:=A_2)$. Then $x(A) \rightarrow_{\beta} x(A_1)[x(A_2)/x]$.

The projection lemma is not strong enough to give us PSN. The problem is that if $A \rightarrow_{\beta} B$ then sometimes $x(A) \equiv x(B)$, as in $x(y:=(\lambda x,C)D) \rightarrow_{\beta} x(y:C(z:=D))$. 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 $A \rightarrow_{\beta} B$ then $A \rightarrow_{\lambda x} B$.

**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 $A \rightarrow_{\beta} A_1(x:=A_2) \rightarrow_{\lambda x} x(A_1(x:=A_2)) \rightarrow_{\beta} x(A_1)[x(A_2)/x]$.

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

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

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

4
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 $F$ be a set of function symbols, $X$ a set of variables such that $F \cap X = \emptyset$, let $T(F, X)$ be the set of (open) terms over $F$ and $X$. Let $\triangleright$ be a partial order on $F$. Let $\tau$ be a map assigning to every function symbol $f \in F$ one of the words mult or lex.

The recursive path order $\triangleright_{rpo}$ on $T(F, X)$ induced by $\triangleright$ and $\tau$ is defined by

$$f(s_1, \ldots, s_m) \triangleright_{rpo} g(t_1, \ldots, t_n)$$

iff

$$\exists i \leq m \land \forall j \left( f(s_1, \ldots, s_m) \triangleright_{rpo} t_j \right) \lor \left( f = g \land \forall j \left( f(s_1, \ldots, s_m) \triangleright_{rpo} t_j \right) \land \left( s_1, \ldots, s_m \triangleright_{rpo} \tau(f(s_1, \ldots, s_m)) \right) \right)$$

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

- $\langle s_1, \ldots, s_m \rangle \triangleright_{lex} \langle t_1, \ldots, t_n \rangle$ iff there is some $i \leq m, n$, $s_i = t_i$ and $s_{i-1} = t_{i-1}$, $s_i \triangleright_{rpo} t_i$ or $s_i = t_i$, $s_{m} = t_{m}$, $m > n$.

- $\langle s_1, \ldots, s_m \rangle \triangleright_{mult} \langle t_1, \ldots, t_n \rangle$ 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 $s \triangleright_{rpo} t$ one or more times.'

In [Ferreira & Zantema 94], $\tau$ is called status function. More complex extensions of $\triangleright_{rpo}$ than multiset or lexicographic are even possible.

Theorem 4.2 (Dershowitz) Let $\triangleright$ be a partial order and $\tau$ a status function on a set of function symbols $F$, let $\triangleright_{rpo}$ be the induced recursive path order. Then

$$\triangleright_{rpo} \text{ is well-founded} \iff \triangleright \text{ is well-founded}$$

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

5 PSN for $\lambda x$

In this section we use the recursive path order to show that $\lambda x$ has PSN. Since the recursive path order is about first order term rewrite systems, we need to translate terms of $\lambda x$ 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 $\lambda x$-terms.

Definition 5.1 We define the set $\lambda x^{<\infty} \subseteq \lambda x$ by

$$\lambda x^{<\infty} = \{ A \in \lambda x \mid \text{for all subterms } B \text{ of } A, x(B) \in \text{SN}_\beta \}$$

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

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

Proof: Every $\rightarrow_\beta$-reduction path of length $n$ of $x(A(x:=B))$ can be extended to a reduction path of length $n+1$ of $x(\lambda x.A.B)$ by prefixing it by $x(\lambda x.A.B) \equiv (\lambda x.B) \equiv x(x(B)) \rightarrow_\beta x(A[x(B)/x] \equiv x(A[x:=B]))$. □
Lemma 5.3 If \( A \in \lambda x^{<\infty} \) and \( A \rightarrow_{\lambda x} A' \) then \( A' \in \lambda x^{<\infty} \).

Proof: Induction on the structure of \( A \). \( \square \)

Definition 5.4 We define the set of labelled terms \( \lambda^l \) by the following abstract syntax:

\[
A ::= * \mid A \cdot n \mid \lambda A \mid A(A)_n
\]

where \( n \) ranges over the natural numbers.

We define by induction on the structure of terms a translation \( T : \lambda x^{<\infty} \rightarrow \lambda^l \):

\[
T(x) = *
\]

\[
T(AB) = T(A) \cdot n T(B) \quad \text{where} \quad n = \hat{\beta}(AB)
\]

\[
T(\lambda x.A) = \lambda T(A)
\]

\[
T(A(x:=B)) = T(A)(T(B))_n \quad \text{where} \quad n = \hat{\beta}(A(x:=B))
\]

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

Definition 5.5 We define a TRS \( \lambda^l \) using as terms the set \( \lambda^l \) and having the reduction relation \( \rightarrow_1 \) defined by

\[
(\lambda A)_m B \rightarrow_1 A(B)_n \quad \text{if} \quad m > n
\]

\[
(A \cdot m B)(C)_n \rightarrow_1 A(C)_p \cdot q B(C)_r \quad \text{if} \quad n \geq p, q, r
\]

\[
(\lambda A)(C)_n \rightarrow_1 \lambda A(C)_n
\]

\[
A(C) \rightarrow_1 C
\]

\[
A(C) \rightarrow_1 A
\]

\[
A \cdot m B \rightarrow_1 A \cdot n B \quad \text{if} \quad m > n
\]

\[
A(B)_m \rightarrow_1 A(B)_n \quad \text{if} \quad m > n
\]

Note that \( \rightarrow_1 \) is not confluent; for our purposes this is no problem since \( \rightarrow_1 \) is only designed to be useful for proving strong normalisation. The last two rules are called Deer in [Zantema 94] and are necessary to decrease the labels of applications and substitutions if inside of them a \( \rightarrow_{\beta\eta} \)-reduction is performed. Note that in the presence of the Deer rules we could also have \( (\lambda A) \cdot n + 1 B \rightarrow_1 A(B)_n \) for all \( n \) instead of \( (\lambda A) \cdot m B \rightarrow_1 A(B)_n \) for all \( m > n \).

Lemma 5.6 The relation \( \rightarrow_1 \) is a subrelation of some recursive path order. (That is, there is a precedence relation \( \triangleright \) such that for all \( A, B \in \lambda^l \), if \( A \rightarrow_1 B \), then \( A \triangleright_{\text{rpo}} B \), where \( \triangleright_{\text{rpo}} \) is the rpo ordering induced by \( \triangleright \).)

Proof: Define the precedence \( \triangleright \) by

\[
\cdot n \triangleright \cdot \triangleright \cdot n \triangleright \cdot \triangleright \lambda, *
\]

and the status function \( \tau \) by \( \tau(\cdot n) = \tau(\lambda) = \tau(\cdot) = \text{lex} \). Then \( \rightarrow_1 \) is a subrelation of the induced recursive path order \( \triangleright_{\text{rpo}} \). \( \square \)

Corollary 5.7 \( \rightarrow_1 \) is SN.

Proof: By Theorem 4.2, \( \triangleright_{\text{rpo}} \) of Lemma 5.6 is strongly normalising, hence by Lemma 5.6 \( \rightarrow_1 \) is strongly normalising. \( \square \)

Lemma 5.8 If \( A \in \lambda x^{<\infty} \) and \( A \rightarrow_{\lambda x} A' \) then \( T(A) \rightarrow_1 T(A') \).

Proof: Induction on the structure of \( A \); we treat some of the more interesting cases.
Then $T(A) = (\lambda T(A_1)) \cdot m T(A_3) \rightarrow_i T(A_1)(x:=T(A_2)) \cdot n T(A_3)$ where $m = \hat{\beta}(A)$; $n = \hat{\beta}(A')$; note that $m > n$ by Lemma 5.2.

- $A \equiv (A_1 A_2)(x:=A_3) \rightarrow_x (A_1(x:=A_3)) \cdot (A_2(x:=A_3)) \equiv A'$.
  
  Then $T(A) = (\lambda T(A_1))(T(A_2))(T(A_3)) \rightarrow_i (T(A_1)(T(A_2))(T(A_3))) \cdot n T(A_3) \equiv T(A')$, where $m = \hat{\beta}(A_1 A_2)$, $n = \hat{\beta}(A)$, $p = \hat{\beta}(A_1(x:=A_3))$, $q = \hat{\beta}(A_2(x:=A_3))$; note that $n \geq p$ and $n \geq q$.

- $A \equiv x(x:=A_1) \rightarrow_x A_1 \equiv A'$. Then $T(A) = * T(A_1) \rightarrow_i T(A_1) \equiv T(A')$ where $m = \hat{\beta}(A)$.

- $A \equiv A_1(x:=A_2) \rightarrow_x \gamma c A_1 \equiv A'$. Then $T(A) = T(A_1) \rightarrow_i T(A_1) \equiv T(A')$ where $m = \hat{\beta}(A)$.

- $A \equiv (\lambda y.A_1) (x:=A_2) \rightarrow_x \lambda y. (A_1(x:=A_2)) \equiv A'$. Then $T(A) = (\lambda T(A_1))(T(A_2)) \rightarrow_i (\lambda (T(A_1))(T(A_2)) \cdot m T(A_1) \equiv T(A')$ where $m = \hat{\beta}(A) = \hat{\beta}(A')$.

- $A \equiv A_1 A_2 \rightarrow_{\lambda x} A_1 A_2 \equiv A'$. Then $T(A) = T(A_1) \cdot m T(A_2) \rightarrow_i T(A_1) \cdot m T(A_2) \rightarrow_i T(A_1) \cdot n T(A_2)$ where $m = \hat{\beta}(A) \geq n = \hat{\beta}(A')$.

\[\square\]

**Corollary 5.9 (PSN)**

1. $A \in S N_{\lambda x} \iff A \in \lambda x^{<\infty}$

2. $\lambda x$ preserves strong normalisation

6 \ $\lambda u$, $\lambda s$ and extensions

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

6.1 The calculi $\lambda u$ and $\lambda s$

**Definition 6.1** Terms and substitutions of $\lambda u$ are defined by the following abstract syntaxes:

\[
\begin{align*}
\lambda u &::= \text{n} \mid (\lambda u) \mid (\lambda a) \mid (a[s]) \\
\lambda s &::= a \mid \gamma (s) \mid \top
\end{align*}
\]

where $n$ ranges over the set $\{1, 2, 3, 4, \ldots\}$.

The reduction relation $\rightarrow_{\lambda u}$ is the union of $\rightarrow_{u_{\beta}}$ and $\rightarrow_{v}$ which are defined by

\[
\begin{align*}
(\lambda a)b &\rightarrow_{u_{\beta}} a[b/]
\end{align*}
\]

\[
\begin{align*}
(ab)[s] &\rightarrow_v a[s][s] \\
(\lambda a)[s] &\rightarrow_v \lambda (a[\gamma (s)]) \\
1[a/ ] &\rightarrow_v a \\
2[a/ ] &\rightarrow_v n \\
1[\gamma (s)] &\rightarrow_v \bot \\
n + 1[\gamma (s)] &\rightarrow_v n[s][1] \\
n[1] &\rightarrow_v n + 1
\end{align*}
\]
Some initial intuition to motivate the reduction rules of $\lambda v$: $a[b/]$ stands for ‘substitute $b$ for $1$ in $a$’, $\{s\}$ 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 $\{l\}$ 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.)

\[
\begin{align*}
(\lambda(\lambda(12)))(11) & \rightarrow_{\text{Beta}} (\lambda(12))[11/]
\rightarrow_{v} \lambda((12)[1(11)])
\rightarrow_{v} \lambda(1[1(11)]2[1(11)])
\rightarrow_{v} \lambda(12[1(11)])
\rightarrow_{v} \lambda(1[1(11)]1[1])
\rightarrow_{v} \lambda(1(11)[1])
\rightarrow_{v} \lambda(1(11)[1])
\rightarrow_{v} (\lambda(22))
\end{align*}
\]

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

**Definition 6.2** Terms and substitutions of $\lambda v$ are defined by the following abstract syntaxes:

\[
a ::= n | (aa) | (\lambda a) | (\phi_i^j a) | (a\sigma^i a)
\]

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

The reduction relation $\rightarrow_{\lambda v}$ is the union of $\rightarrow_{\text{Beta}}$ and $\rightarrow_{s}$ which are defined by

\[
\begin{align*}
(\lambda a) b & \rightarrow_{\text{Beta}} \lambda (a\sigma^i b)
(\lambda a)\sigma^i b & \rightarrow_{s} \lambda (a\sigma^{i+1} b)
(a_1 a_2)\sigma^i b & \rightarrow_{s} (a_1 \sigma^i b)(a_1 \sigma^i b)
\end{align*}
\]

\[
\begin{align*}
n\sigma^i b & \rightarrow_{s} \begin{cases} n-1 & \text{if $n > i$} \\
\phi_i^j(b) & \text{if $n = i$} \\
\phi_i^j(b) & \text{if $n < i$}
\end{cases}
\end{align*}
\]

\[
\begin{align*}
\phi_i^j(\lambda a) & \rightarrow_{s} \lambda (\phi_i^{j+1} a)
\phi_i^j(a_1 a_2) & \rightarrow_{s} (\phi_i^j a)(\phi_i^j a)
\phi_i^j n & \rightarrow_{s} \begin{cases} n+i-1 & \text{if $n > k$} \\
n & \text{if $n \leq k$}
\end{cases}
\end{align*}
\]

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 $i$, $\phi_i^j(a)$ 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 $\lambda v$.

\[
\begin{align*}
(\lambda(\lambda(12)))(11) & \rightarrow_{\text{Beta}} (\lambda(12))\sigma^1(11)
\rightarrow_{s} \lambda((12)\sigma^2(11))
\rightarrow_{s} \lambda(1(2\sigma^2(11)))
\rightarrow_{s} \lambda(1(2\sigma^2(11)))
\rightarrow_{s} \lambda(2\sigma^2(11))
\rightarrow_{s} \lambda(1)\phi_i^j(1)
\rightarrow_{s} \lambda(1)\phi_i^j(1)
\rightarrow_{s} \lambda(1)\phi_i^j(1)
\rightarrow_{s} \lambda(1)\phi_i^j(1)
\rightarrow_{s} \lambda(1)\phi_i^j(1)
\rightarrow_{s} \lambda(1)\phi_i^j(1)
\rightarrow_{s} \lambda(22)
\end{align*}
\]
The calculus $\lambda u$ is very similar to $\lambda v$. The difference is mainly in the moment of updating: in $\lambda v$ every step $n + 1[\tilde{v}(s)] \rightarrow_v a[s][1]$ creates an update substitution $[1]$ whereas in $\lambda u$ the update functionsymbol $\phi_b^s$ is only created at the actual moment of substitution in $n \sigma^n a \rightarrow_\phi a$. Also, in the reductions $\mu \sigma^n b \rightarrow_\mu n - 1 (n > i)$ and $\mu \sigma^n b \rightarrow_\mu n (n < i)$, there is no update functionsymbol generated whereas in $n + 1[\tilde{v}(s)] \rightarrow_\mu a[s][1]$ an update substitution is created regardless of whether the substitution $[1]$ is binding or is void.

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

We show that $\lambda v$ and $\lambda u$ are PSN by using the labelled calculus $\lambda^l$. The proof is very similar to the proof of PSN for $\lambda^l$ that we gave in the previous section.

For $\lambda v$ and $\lambda u$ we have the usual properties such as SN, CR, UN for $\rightarrow_\nu$ respectively $\rightarrow_\sigma$, substitution lemma, projection lemma, soundness lemma and confluence for $\rightarrow_\lambda$ respectively $\rightarrow_\lambda$. We denote the $\rightarrow_\nu$-normal form respectively $\rightarrow_\sigma$-normal form of a term $b$ by $\nu(b)$ respectively $\sigma(b)$. Note that a substitution of $\lambda v$ is of the form $\sigma^n (b/)$ or $\sigma^n (1)$ for some $n$.

We denote $\beta$-reduction on $\lambda v$-terms as well as on $\lambda u$-terms by $\rightarrow_\beta$; for a $\lambda v$- respectively $\lambda u$-term $a$ we write $\beta(a)$ to denote the maximal number of $\beta$-reduction steps starting from $\nu(a)$ respectively $\sigma(a)$, if this number exists.

**Definition 6.3**

$$\lambda v^{<\infty} := \{ a \in \lambda v \mid \forall b \subseteq a[\nu(b) \in SN_\beta] \}$$

$$\lambda u^{<\infty} := \{ a \in \lambda u \mid \forall b \subseteq a[\sigma(b) \in SN_\beta] \}$$

**Lemma 6.4**

1. $\lambda v^{<\infty}$ is closed under $\rightarrow_\lambda v$-reduction

2. $\lambda u^{<\infty}$ is closed under $\rightarrow_\lambda u$-reduction

**Definition 6.5**

1. $\mathcal{T}_v : \lambda v^{<\infty} \rightarrow \Lambda^l$ is defined by

$$\mathcal{T}_v(\text{nil}) = *$$

$$\mathcal{T}_v(ab) = \mathcal{T}_v(a) \cdot \mathcal{T}_v(b) \quad \text{where } p = \beta(ab)$$

$$\mathcal{T}_v(\lambda a) = \lambda T_v(a)$$

$$\mathcal{T}_v(a[\nu^n (b/)]) = \mathcal{T}_v(a)[T_v(b)]_p \quad \text{where } p = \beta(a[\nu^n (b/)])$$

$$\mathcal{T}_v(a[\nu^n (1)]) = T_v(a)$$

2. $\mathcal{T}_u : \lambda u^{<\infty} \rightarrow \Lambda^l$ is defined by

$$\mathcal{T}_u(\text{nil}) = *$$

$$\mathcal{T}_u(ab) = \mathcal{T}_u(a) \cdot \mathcal{T}_u(b) \quad \text{where } p = \beta(ab)$$

$$\mathcal{T}_u(\lambda a) = \lambda T_u(a)$$

$$\mathcal{T}_u(a[\nu^i (b)]) = \mathcal{T}_u(a)[T_u(b)]_p \quad \text{where } p = \beta(a[\nu^i (b)])$$

$$\mathcal{T}_u(\phi_b^1 a) = T_u(a)$$

**Lemma 6.6**

1. If $a \in \lambda u^{<\infty}$ and $a \rightarrow_\nu b$ then $T_v(a) \rightarrow_\nu T_v(b)$

2. If $a \in \lambda v^{<\infty}$ and $a \rightarrow_{\nu Beta} b$ then $T_v(a) \rightarrow^+_\nu T_v(b)$

3. If $a \in \lambda u^{<\infty}$ and $a \rightarrow_\sigma b$ then $T_u(a) \rightarrow_\sigma T_u(b)$

4. If $a \in \lambda u^{<\infty}$ and $a \rightarrow_{\sigma Beta} b$ then $T_u(a) \rightarrow^+_\sigma T_u(b)$

**Proof:** Induction on the structure of $a$. $\square$

**Theorem 6.7**

1. $a \in SN_{\lambda u} \iff a \in \lambda v^{<\infty}$
2. →_λμ has PSN
3. a ∈ SN_λμ ⇔ a ∈ λμ<∞
4. →_λμ has PSN

Proof:
1. ⇒ by projection; ⇐: since →_σ is SN, any infinite →_λμ-reduction must contain infinitely many →_cBeta-steps. Therefore an infinite reduction of a pure term which is SN for →_σ translates by T_σ into an infinite →_F-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 λx of [Abadi et al. 90] was designed to be able to compose substitutions. The price however is not having PSN (cf. [Mellies 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 λσ. For the precise definition of λσ, the reader is referred to [Abadi et al. 90]. The composition of substitutions in λσ is mainly performed by two rules, Comp and Map. The first glues two substitutions together: a[s][t] Comp a[sot], while Map allows the distribution of the second substitution over the first: (b c d')ot Mapp b[t] (((c d')ot) Map b[t] c[t] (d'ot)).

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

\[ t ::= x | t t | \lambda x . t | t(x := i) \]

where \( x := i \) is shorthand for \( (x_1, \ldots, x_m := t_1, \ldots, t_m) \); reductions are similar as for λx plus the composition rule

\[ a(x := b)(y := c) \quad \longrightarrow \quad a(x := b(y := c)) \]

In this calculus one can imitate the counterexample to PSN of λσ (cf [Mellies 95]). In fact, even the calculus λx extended with the rule

\[ a(x := b)(y := c) \quad \longrightarrow \quad a(x := b(y := c)) \quad \text{if} \quad y \notin FV(a) \]

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

First we define substitutions \( a_m \) for \( m \in \mathbb{N} \) by

\[ a_0 = (x := (\lambda y . a)b) \]
\[ a_{m+1} = (y := b a_m) \]

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.)

\[
(\lambda y . (\lambda y' . x)((\lambda y . a)b)((\lambda y . a)b)) \quad \rightarrow \quad x(y' := (\lambda y . a)b)(y := (\lambda y . a)b)
\]
\[
\rightarrow x(y' := ((\lambda y . a)b)(y := (\lambda y . a)b))
\]
\[
\rightarrow x(y' := (\lambda y . a)(y := (\lambda y . a)b))(b(y := (\lambda y . a)b))
\]
\[
\equiv x(y' := (\lambda y . a)b a_0)(b a_0)
\]
\[
\rightarrow x(y' := a a_0(y := b a_0))
\]
\[
\equiv x(y' := a a_0 a_1)
\]
Recall that we proved PSN by showing that every term of which the x-normal form of any of its subterms is in SN, is also SN for \( \rightarrow_{\text{beta}} \cup \rightarrow_{x} \). With the extra composition reduction defined above, there is an easy counterexample to that: the term \( x(y:=z z)(Z:=A W . W W) \) has x-normalform \( x \) (and is also SN for \( \lambda x \)-reduction), but it has \( \Omega \) 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 \( \lambda \sigma: T(x(y:=z z)(z:=\lambda w . w w)) \equiv \ast((\ast \ast)\lambda(\ast \ast)\ast)\ast \) whereas after composition of the two substitutions, the label of the innermost substitution does not exist: \( T(x(y:=z z)(z:=\lambda w . w w)) \equiv \ast((\ast \ast)\lambda(\ast \ast)\ast)\ast \). So reduction in \( \lambda \sigma \) does not always decrease \( T \)-images.

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

\[
(a(x:=b)(y:=c)) \rightarrow_{x} a(x:=b(y:=c)) \text{ if } y \notin FV(a), x \in FV(x(a))
\]

The idea behind this rule is that, if \( x \in FV(x(a)) \), then \( b(y:=c) \) will occur as a subterm of some \( \rightarrow_{x} \)-reduct. Hence allowing to create \( b(y:=c) \) at this point will not spoil PSN. We strongly believe that adding this reduction rule does not spoil SN (i.e. \( \lambda x \) extended with the rule \( \rightarrow_{x} \) 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 \( \text{Tree} \) (i.e. trees are identified up to 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 \( \text{Tree}^{*} \), where some nodes in a tree may have a marker \( * \).

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 \( n^{*}(t_{1}, \ldots, t_{p}) \). In the following, we abbreviate \( t_{1}, \ldots, t_{p} \) to \( t \). On these commutative labelled trees with markers (the set \( \text{Tree}^{*} \)), a reduction relation \( \Rightarrow \) is defined.

**Definition 7.1** The relation \( \Rightarrow \) on \( \text{Tree}^{*} \) is defined as follows.

\[
\begin{align*}
n(i) & \Rightarrow n^{*}(i), \\
n^{*}(i) & \Rightarrow m(n^{*}(i), \ldots, n^{*}(i)), \\
n^{*}(s, i) & \Rightarrow n(s^{*}, \ldots, s^{*}, i), \\
n^{*}(i) & \Rightarrow t_{i}, \\
1 \leq i \leq p.
\end{align*}
\]
Furthermore, the relation $\Rightarrow$ is compatible with the tree-forming operations, that is, if $t_i \Rightarrow t'_i$, then $n(t_1, \ldots, t_i, \ldots, t_p) \Rightarrow n(t_1, \ldots, t'_i, \ldots, t_p)$.

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

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 substitution.

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

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

For notational convenience, we abbreviate the sequence of definitions $(x_1:=P_1) \cdots (x_n:=P_n)$ to $(x:=P)$.

**Definition 7.3** For $M \in \lambda x^{<\infty}$, we define the tree $T(M)$ by induction on the length of $M$ as follows.

\[
T(x) = 0,
\]

\[
T(Q N) = \begin{cases} 
\beta(Q N) \\
T(Q) \\
T(N) 
\end{cases}
\]

\[
T(\lambda y. N) = T(N)
\]

\[
T(y(x:=P)) = \begin{cases} 
0 \\
T(P_1) \\
\cdots \\
T(P_n) \\
if \ y \notin \{x_1, \ldots, x_n\}
\end{cases}
\]

\[
T(x_i(x:=P)) = \begin{cases} 
T(P_1) \\
\cdots \\
T(P_i-1) \\
T(P_i(x_{i+1}:=P_{i+1}) \cdots (x_n:=P_n)) \\
\beta((Q N)(x:=P))
\end{cases}
\]

\[
T((Q N)(x:=P)) = \begin{cases} 
T(Q(x:=P)) \\
T(N(x:=P))
\end{cases}
\]

\[
T((\lambda y. N)(x:=P)) = T(N(x:=P))
\]
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 $M \rightarrow_{\chi} N$, 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 $\rightarrow_{\text{Beta}}$ reduction by $T$.

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

**Proof:** By induction on the length of $N$. □

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

**Proof:** By induction on the length of $N$, using Lemma 7.5. First write $N$ as $R(y:=Q)$, with $R$ not a term that ends with a substitution item. (So, the sequence $(y:=Q)$ should be taken as long as possible.) Then distinguish cases according to the structure of $R$. □

**Corrollary 7.7** The calculus $\lambda x$ 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 $\beta$-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 Brujin 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 $\lambda x$-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 benefited from discussions with the following people: Hans Zantema, Gilles Barthe, Daniel Briaud, Twan Laan, Pierre Lescanne, Rob Nederpelt and Kristoffer Rose.

References


### Computing Science Reports

**In this series appeared:**

<table>
<thead>
<tr>
<th>Issue</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>93/01</td>
<td>R. van Geldrop</td>
<td>Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.</td>
</tr>
<tr>
<td>93/02</td>
<td>T. Verhoeff</td>
<td>A continuous version of the Prisoner's Dilemma, p. 17</td>
</tr>
<tr>
<td>93/03</td>
<td>T. Verhoeff</td>
<td>Quicksort for linked lists, p. 8</td>
</tr>
<tr>
<td>93/05</td>
<td>J.C.M. Baeten, C. Verhoef</td>
<td>A congruence theorem for structured operational semantics with predicates, p. 18.</td>
</tr>
<tr>
<td>93/06</td>
<td>J.P. Veltkamp</td>
<td>On the unavoidability of metastable behaviour, p. 29</td>
</tr>
<tr>
<td>93/07</td>
<td>P.D. Moerland</td>
<td>Exercises in Multiprogramming, p. 97</td>
</tr>
<tr>
<td>93/08</td>
<td>J. Verhoosel</td>
<td>A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.</td>
</tr>
<tr>
<td>93/10</td>
<td>K.M. van Hee</td>
<td>Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.</td>
</tr>
<tr>
<td>93/16</td>
<td>H. Schepers, J. Hooman</td>
<td>A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27.</td>
</tr>
<tr>
<td>93/17</td>
<td>D. Aistein, P. van der Stok</td>
<td>Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.</td>
</tr>
<tr>
<td>93/18</td>
<td>C. Verhoef</td>
<td>A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.</td>
</tr>
<tr>
<td>93/19</td>
<td>G.J. Houben</td>
<td>The Design of an Online Help Facility for ExSpect, p. 21.</td>
</tr>
<tr>
<td>93/22</td>
<td>E. Poll</td>
<td>A Typechecker for Bijective Pure Type Systems, p. 28.</td>
</tr>
<tr>
<td>93/23</td>
<td>E. de Kogel</td>
<td>Relational Algebra and Equational Proofs, p. 23.</td>
</tr>
<tr>
<td>93/24</td>
<td>E. Poll, Paula Severi</td>
<td>Pure Type Systems with Definitions, p. 38.</td>
</tr>
<tr>
<td>93/26</td>
<td>W.M.P. van der Aalst</td>
<td>Multi-dimensional Petri nets, p. 25.</td>
</tr>
<tr>
<td>93/27</td>
<td>T. Kloks and D. Kratsch</td>
<td>Finding all minimal separators of a graph, p. 11.</td>
</tr>
<tr>
<td>93/28</td>
<td>F. Kamareddine and R. Nederpelt</td>
<td>A Semantics for a finite λ-calculus with de Bruijn indices, p. 49.</td>
</tr>
<tr>
<td>93/29</td>
<td>R. Post and P. De Bra</td>
<td>GOLD, a Graph Oriented Language for Databases, p. 42.</td>
</tr>
<tr>
<td>93/30</td>
<td>J. Deogun, T. Kloks, D. Kratsch, H. Müller</td>
<td>On Vertex Ranking for Permutation and Other Graphs, p. 11.</td>
</tr>
</tbody>
</table>
93/31 W. Körver
93/32 H. ten Eikelder and H. van Geldrop
93/33 L. Loyens and J. Moonen
93/34 J.C.M. Baeten and J.A. Bergstra
93/35 W. Ferrer and P. Severi
93/36 J.C.M. Baeten and J.A. Bergstra
93/37 J. Bruantkreef J-P. Katoen R. Koymans S. Mauw
93/38 C. Verheoef
93/39 W.P.M. Nuijen E.H.L. Aarts D.A.A. van Erp Teulman Kip K.M. van Hee
93/40 P.D.V. van der Stok M.M.M.P.J. Claessen D. Alstein
93/41 A. Bijlsma
93/42 P.M.P. Rambags
93/43 B.W. Watson
93/44 B.W. Watson
93/45 E.J. Luit J.M.M. Martin
93/46 T. Kloks D. Kratsch J. Spinrad
93/47 W. v.d. Aalst P. De Bra G.J. Houben Y. Kornatzky
93/48 R. Gerth
94/01 P. America M. van der Kamen M.R.K. Nederpelt O.S. van Roomelen H.C.M. de Swart
94/02 F. Kamareddine R.P. Nederpelt
94/03 I.B. Hamman K.M. van Hoe
94/04 J.C.M. Baeten J.A. Bergstra
94/05 P. Zhou J. Hoeman
94/06 T. Basten T. Kuz J. Black M. Coffin D. Taylor
94/07 K.R. Apt R. Bol
94/08 O.S. van Roomelen
94/09 J.C.M. Baeten J.A. Bergstra

Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.


ILIAS, a sequential language for parallel matrix computations, p. 20.

Real Time Process Algebra with Infinitesimals, p.39.

Abstract Reduction and Topology, p. 28.

Non Interleaving Process Algebra, p. 17.

Design and Analysis of Dynamic Leader Election Protocols in Broadcast Networks, p. 73.

A general conservative extension theorem in process algebra, p. 17.

Job Shop Scheduling by Constraint Satisfaction, p. 22.


Temporal operators viewed as predicate transformers, p. 11.

Automatic Verification of Regular Protocols in P/T Nets, p. 23.

A taxonomy of finite automata construction algorithms, p. 87.

A taxonomy of finite automata minimization algorithms, p. 23.

A precise clock synchronization protocol, p.


Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.

The object-oriented paradigm, p. 28.

Canonical typing and Π-conversion, p. 51.


Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18.


Time and the Order of Abstract Events in Distributed Computations, p. 29.


A Hierarchical Diagrammatic Representation of Class Structure, p. 22.

Process Algebra with Partial Choice, p. 16.
94/10  T. Verhoeven
T. Verhoeff
The testing Paradigm Applied to Network Structure. p. 31.

94/11  J. Peleska
C. Hanzag
C. Petersohn
A Comparison of Ward & Mellor’s Transformation

94/12  T. Kloks
D. Kratsch
H. Müller

94/13  R. Seljé
E.H.L. Aarts
J.K. Leastra
A New Method for Integrity Constraint checking in Deductive Databases, p. 34.

94/14  W. Peremans
R.J.M. Vaessens
Job Shop Scheduling by Local Search, p. 21.

94/15  R. Seljé
E.H.L. Aarts
J.K. Leastra
Ups and Downs of Type Theory, p. 9.

94/16  R.C. Backhouse
B.W. Watson
H. Doornbos
Mathematical Induction Made Calculational, p. 36.

94/17  S. Mauw
F. Kamareddine
M.A. Reniers
R. Nederpelt
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
Beyond β-Reduction in Church’s λ →, p. 22.

94/20  R. Bloo
F. Kamareddine
R. Nederpelt
The performance of single-keyword and multiple-keyword pattern matching
algorithms. p. 46.

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
D. Dams
O. Grumberg
R. Gerth
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 ∀CTL*, ∃CTL* and CTL*, p. 28.

94/25  T. Kloks
Kₜ₀-free and W₄-free graphs, p. 10.

94/26  R.R. Hoogerwoord
R.R. Hoogerwoorden
On the foundations of functional programming: a programmer’s point of view, p. 56.

94/27  S. Mauw and H. Mulder
C.W.A.M. van Overveld
M. Verhoeven
S. Mauw and H. Mulder

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
J. Hoornman
Correctness of Real Time Systems by Construction, p. 22.

94/30  J.C.M. Baeten
J.A. Bergstra
Gh. Stefanescu
Process Algebra with Feedback, p. 22.

94/31  J.C.M. Baeten
J.A. Bergstra
Gh. Stefanescu
A Boyer-Moore type algorithm for regular expression
pattern matching, p. 22.

94/32  B.W. Watson
R.E. Watson
F. Kamareddine
R. Nederpelt

94/33  J.J. Vereijken
T. Laan
A formalization of the Ramified Type Theory, p.40.

94/34  J.J. Vereijken
T. Laan
The Barendregt Cube with Definitions and Generalised
Reduction, p. 37.

94/35  J.C.M. Baeten
J.A. Bergstra
Gh. Stefanescu
S. Mauw
Delayed choice: an operator for joining Message
Sequence Charts, p. 15.

94/36  S. Mauw
F. Kamareddine
R. Nederpelt
Canonical typing and II-conversion in the Barendregt
Cube, p. 19.

94/37  J.C.M. Baeten
J.A. Bergstra
Gh. Stefanescu
S. Mauw
Simulating and Analyzing Railway Interlockings in
ExSpect, p. 30.

94/38  T. Basten
R. Bol
M. Voorhoeve
A. Bijlsma
C.S. Scholten
Point-free substitution, p. 10.
On the equivalence covering number of splitgraphs, p. 4.

Distributed Consensus and Hard Real-Time Systems, p. 34.

Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph, p. 6.

Concatenation of Graphs, p. 7.

Category Theory as Coherently Constructive Lattice Theory: An Illustration, p. 35.

Verifying Sequentially Consistent Memory, p. 160

Tutorial voor de ExSpect-bibliotheek voor “Administratieve Logistiek”, p. 43.

The A-cube with classes of terms modulo conversion, p. 16.

On Λ-conversion in Type Theory, p. 12.

Fixed-Point Calculus, p. 11.

Process Algebra with Propositional Signals, p. 25

A short and flexible proof of Strong Normalization for the Calculus of Constructions, p. 27.

Listing simplicial vertices and recognizing diamond-free graphs, p. 4.

Traces and Logic, p. 81

A Partial Order Approach to Branching Time Logic Model Checking, p. 20.

The Construction of a small Communication Library, p. 16.

Formalizing Process Algebraic Verifications in the Calculus of Constructions, p. 49.

Concrete process algebra, p. 134.

An Isotopic Invariant for Planar Drawings of Connected Planar Graphs, p. 9.

A Type Inference Algorithm for Pure Type Systems, p. 20.

A Quantitative Analysis of Iterated Local Search, p. 23.

Drawing Execution Graphs by Parsing, p. 10.

Preservation of Strong Normalisation for Explicit Substitution, p. 12.

Discrete Time Process Algebra, p. 20

MathJpad: A System for On-Line Preparation of Mathematical Documents, p. 15
R. Selje
Deductive Database Systems and integrity constraint checking, p. 36.

S. Mauw and M. Reniers
Empty Interworkings and Refinement Semantics of Interworkings Revised, p. 19.

B.W. Watson and G. Zwaan

A. Peine, C. Verhoef, S.F.M. Vlijmen (eds.)
De proceedings: ACP'95, p.

P. Niebert and W. Penaczk
On the Connection of Partial Order Logics and Partial Order Reduction Methods, p. 12.

D. Dam, O. Grumborg, R. Gerth
Abstract Interpretation of Reactive Systems: Preservation of CTL*, p. 27.

S. Mauw and E.A. van der Meulen
Specification of tools for Message Sequence Charts, p. 36.

F. Kamareddine and T. Laan
A Reflection on Russell’s Ramified Types and Kripke’s Hierarchy of Truths, p. 14.

J.C.M. Baeten and J.A. Bergstra
Discrete Time Process Algebra with Abstraction, p. 15.

F. van Raamsdonk and P. Severi
On Normalisation, p. 33.

A. van Deursen
Axiomatizing Early and Late Input by Variable Elimination, p. 44.

B. Arnold, A. v. Deursen, M. Res

W.M.P. van der Aalst
Petri net based scheduling, p. 20.

P.P.M. Dignum, W.P.M. Nuijten, L.M.A. Janssen

P. Feijs
Synchronous Sequence Charts In Action, p. 36.

W.M.P. van der Aalst
A Class of Petri nets for modeling and analyzing business processes, p. 24.

P.D.V. van der Stok, J. van der Wal

W. Fokkink, C. Verhoef
A Conservative Look at term Deduction Systems with Variable Binding, p. 29.

H. Jurjus
On Nesting of a Nonmonotonic Conditional, p. 14

J. Hidders, C. Hoskins, J. Paredaens

P. Kelb, D. Dams and R. Gerth
Practical Symbolic Model Checking of the full μ-calculus using Compositional Abstractions, p. 17.

W.M.P. van der Aalst

J. Engelfriet and JI. Versteijnen
Context-Free Graph Grammars and Concatenation of Graphs, p. 35.

J. Zwanenburg
Record concatenation with intersection types, p. 46.

T. Basten and M. Voorhoeve
An algebraic semantics for hierarchical P/T Nets, p. 32.

M. Voorhoeve and T. Basten
Process Algebra with Autonomous Actions, p. 12.

P. de Bra and A. Aerts
Multi-User Publishing in the Web: DreSS, A Document Repository Service Station, p. 12

W.M.P. van der Aalst

S. Mauw
Example specifications in phi-SDL.

T. Basten and W.M.P. v.d. Aalst

W.M.P. van der Aalst and T. Baston
Life-Cycle Inheritance A Petri-Net-Based Approach, p. 18.

M. Voorhoeve
Structural Petri Net Equivalence, p. 16.

A.T.M. Aerts, P.M.E. De Bra, J.T. de Munk

F. Dignum, H. Weigand, E. Verharen