Explicit substitutions for control operators[*]

Gilles Barthe[**], Fairouz Kamareddine[***], and Alejandro Ríos[†]

Abstract. *The $\lambda\Delta$-calculus is a λ-calculus with a local operator closely related to normalisation procedures in classical logic and control operators in functional programming. We introduce $\lambda\Delta\mathsf{exp}$, an explicit substitution calculus for $\lambda\Delta$, show it preserves strong normalisation and that its simply typed version is strongly normalising. Interestingly, $\lambda\Delta\mathsf{exp}$ is the first example for which the decency method of showing preservation of strong normalisation (PSN) works whereas the structure preserving method which is based on the decency method does not. In particular, $\lambda\Delta\mathsf{exp}$ is a very simple calculus yet is not structure preserving. This shows that the structure preserving notion intended to give a general description of calculi of explicit substitution that satisfy PSN, is restrictive. To our knowledge, $\lambda\Delta\mathsf{exp}$ is the first calculus of explicit substitution that is not structure preserving.*[5]

1 Introduction

Explicit substitutions were introduced in [1] as a bridge between λ-calculus and its implementation. The fundamental idea is simple: in order to provide a full account of the computations involved in computing a λ-term, one must describe a method to compute substitutions.

Over the last five years most of the research in the area has focused on one specific problem, the *Preservation of Strong Normalisation* (PSN):

> *is every strongly normalising term of the traditional λ-calculus strongly normalising with respect to a given calculus of explicit substitutions?*

In 1994, Melliès settled in the negative the problem of PSN for the original calculus of explicit substitutions $\lambda\sigma$ [25]. Since then, various calculi with the PSN property have been proposed (see for example [5, 19, 8, 26]). The difficulties in achieving PSN for λ-calculus raise the question of the generality of explicit substitutions:

> *can explicit substitutions provide a bridge between the higher-order rewriting systems used in functional programming and their implementation?*

In this paper, we provide a partial (and preliminary) answer to this question by extending the paradigm of explicit substitutions to λ-calculi with control operators. In a first instance, we consider a specific calculus with control operators, called $\lambda\Delta$ (see [30]), and define its explicit substitution variant $\lambda\Delta\mathsf{exp}$. Then we prove that the $\lambda\Delta\mathsf{exp}$ preserves SN and deduce that the simply typed $\lambda\Delta\mathsf{exp}$ calculus is strongly normalizing.

There are at least three reasons to consider such an extension:

1. control operators play a crucial role in functional programming languages, such as LISP [31], SML [2], Scheme [13], etc. We will only be able to claim a positive answer to the above question if the theory of explicit substitutions can be extended to control operators;

[*] This work is supported by NWO and the British council under UK/Dutch joint scientific research project JRP240 and EPSRC grant GR/K 25014. We are grateful for Roel Bloo for his observation that the RPO method of Bloo and Geuvers, contrary to the structure preserving method, does apply to $\lambda\Delta\mathsf{exp}$.

[**] CWI, P.O. Box 94079, 1090 GB Amsterdam, the Netherlands, email gilles@cwi.nl

[***] University of Glasgow, Department of Computing Science, 17 Lilybank Gardens, Glasgow G12 8QQ, Scotland, UK, email fairouz@dcs.gla.ac.uk

[†] Address as Kamareddine, email rios@dcs.gla.ac.uk

[5] According to Bloo, $\lambda\Delta\mathsf{exp}$ is a calculus he and Rose have been looking for for sometime.

2. control operators and explicit substitutions both have applications in theorem proving and proof theory.[6] The former are used in classical theorem proving and the latter to represent incomplete proofs. By studying explicit substitutions with control operators, we lay the foundations for a classical theorem prover with the ability to handle incomplete proofs and for a classical proof theory based on explicit substitutions.
3. control operators fundamentally differ from λ-calculus in that they are not structure-preserving in the sense of [9]. As a result, the technique developed to prove PSN for explicit substitution calculi using the structure preserving notion (see [9]) cannot be carried over to calculi with control operators. In fact, we will show that the decency method [6] can be adapted to our setting whereas the structure-preserving method [9] cannot despite the fact that it is based on the decency method. This makes $\lambda\Delta\mathsf{exp}$ the first calculus which shows the limitations of the structure preserving notion which was intended to give some genertalisations for proving PSN.

The undertaking of this paper is to provide a representative case study which enables one to see:

1. if the theory of explicit substitutions can be generalised to higher-order rewriting systems;
2. which methods are best suited to carry over such a generalisation.

It should be noted here that the minimal derivation method (see [5, 19]) and the RPO method (see [7]) of showing PSN do apply to $\lambda\Delta$exp. We chose the decency method because it is the one used by Bloo and Rose in their structure preserving method. We include the proof using minimal derivation as an appendix for the interested reader. Applying the RPO technique requires finding the correct labelling and goes through in a straightforward manner as Bloo observed.

Prerequisites and terminology We assume some basic familiarity with λ-calculus [4] and abstract rewriting [21] and use $b \lhd a$ to say that b is a subterm of a. The compatible closure of a notion of reduction R will always be denoted by $\rightarrow_R$. Compatibility itself is defined as usual and the Δ case is treated like the λ case. The reflexive and transitive closure of $\rightarrow_R$ will be denoted by $\twoheadrightarrow_R$. When there is at least one reduction step, we write $\twoheadrightarrow^+_R$. Finally, we let $\mathsf{SN}(R)$ denote the set of strongly normalising terms w.r.t. $\rightarrow_R$.

2 The $\boldsymbol{\lambda\Delta}$-calculus

In this section we describe the syntax of the $\lambda\Delta$-calculus where the reduction rules are closely related to normalisation procedures for classical natural deduction and to reduction rules for control operators (see [30]). We use $.[./.]$ to denote the usual (meta-theoretical) substitution. Free and bound variables are defined as usual and $\mathsf{FV}(a)$ (resp. $\mathsf{BV}(a)$) denotes the free (resp. bound) variables of the term a. In the definitions of $.[./.]$, FV and BV, Δ is treated like λ.

Definition 1

1. The set T of (pure) terms is given by the abstract syntax:

$$T \;=\; V \mid TT \mid \lambda V.T \mid \Delta V.T \qquad \textit{where } V = \{x_n : n \in \mathbb{N}\}$$

2. β-reduction $\rightarrow_\beta$ is defined as the compatible closure of $(\lambda x.a)\, b \quad \rightarrow_\beta \quad a[b/x]$

3. μ-reduction $\rightarrow_\mu$ is defined to be $\rightarrow_{\mu_1} \cup \rightarrow_{\mu_2} \cup \rightarrow_{\mu_3}$ where μ_i-reduction for $1 \leq i \leq 3$ is defined to be the compatible closure of the corresponding i-rule:

$$\begin{array}{lll}
(\Delta x.a)\, b & \rightarrow_{\mu_1} \Delta y.a[\lambda w.y\ (w\ b)/x] & \textit{if } y, w \notin \mathsf{FV}(b), y \neq w \\
\Delta x.x\ a & \rightarrow_{\mu_2} a & \textit{if } x \notin \mathsf{FV}(a) \\
\Delta x.x\ (\Delta y.x\ a) & \rightarrow_{\mu_3} a & \textit{if } x, y \notin \mathsf{FV}(a)
\end{array}$$

4. $\rightarrow_{\beta\mu} = (\rightarrow_\beta \cup \rightarrow_\mu)$.

[6] See [14, 28, 33] for applications of control operators in theorem proving, [27, 24] for applications of explicit substitutions in theorem proving, [3, 11, 15, 23, 28, 29, 30] for applications of control operators in proof theory and [12, 17, 32] for applications of explicit substitutions in proof theory.

We let $x, y, z, w, \ldots$ range over V and $a, b, c, \ldots$ range over T. We consider terms modulo α-conversion (generalised over Δ) and assume the variable convention (VC) of [4] which guarantees that in any context, names of free variables are different from bound ones and that different bound variables are used for different λ's and Δ's. In particular, in $a[b/x]$, we assume that $x \notin \mathsf{FV}(b)$. We use $\mathcal{O}$ to range over $\{\lambda, \Delta\}$.

Proposition 2 *(see [30]) $\rightarrow_{\beta\mu}$ is confluent (CR).*

We define the $\beta\mu$-norm $\beta\mu(a)$ of a pure term a as the maximal number of $\beta\mu$-reduction steps in a reduction starting from a. This will be used in the proof of PSN using the decency method.

Lemma 3 *If $(\mathcal{O}x.b)c$ is a subterm of a pure term a and $\beta\mu(a) < \infty$ then $\beta\mu(c) < \beta\mu(a)$.*

Proof: By induction on the structure of a. □

Lemma 4 *The following holds:*

1. *If $x \neq y$ and $x \notin \mathsf{FV}(c)$ then $a[b/x][c/y] = a[c/y][b[c/y]/x]$.*
2. *If $a \rightarrow_{\beta\mu} a'$ then $a[b/x] \rightarrow_{\beta\mu} a'[b/x]$.*
3. *If $b \rightarrow_{\beta\mu} b'$ then $a[b/x] \twoheadrightarrow_{\beta\mu} a[b'/x]$; if $x \in \mathsf{FV}(a)$ then $a[b/x] \twoheadrightarrow^{+}_{\beta\mu} a[b'/x]$.*
4. *If $a \twoheadrightarrow_{\beta\mu} a'$ and $b \twoheadrightarrow_{\beta\mu} b'$ then $a[b/x] \twoheadrightarrow_{\beta\mu} a'[b'/x]$.*
5. *$\beta\mu(a) \leq \beta\mu(a[b/x])$.*

Proof: 1, 2 and 3 are by induction on the structure of a. 4 is a corollary of 2 and 3. 5: A $\beta\mu$-reduction path starting with a gives by 2 a path of the same length starting with $a[b/x]$. □

3 The $\lambda\Delta$exp-calculus

The $\lambda\Delta\mathsf{exp}$-calculus is obtained from $\lambda\Delta$ by giving an explicit treatment of substitutions.

Definition 5

1. *The set T^e of terms of the $\lambda\Delta\mathsf{exp}$-calculus is given by the abstract syntax:*

   $T^e \ = \ V \mid T^e T^e \mid \lambda V.T^e \mid \Delta V.T^e \mid T^e[V := T^e]$ *where $V = \{x_n : n \in \mathbb{N}.\}$ Note that $T \subseteq T^e$*

2. *$\underline{\beta}$-reduction $\rightarrow_{\underline{\beta}}$ is defined as the compatible closure of $(\lambda x.a)\ b \quad \rightarrow_{\underline{\beta}} \quad a[x := b]$*
3. *$\underline{\mu}$-reduction $\rightarrow_{\underline{\mu}}$ is defined to be $\rightarrow_{\underline{\mu_1}} \cup \rightarrow_{\underline{\mu_2}} \cup \rightarrow_{\underline{\mu_3}}$ where $\underline{\mu_i}$-reduction for $1 \leq i \leq 3$ is defined to be the compatible closure of the corresponding $\underline{i}$-rule:*

$$\begin{array}{lll} (\Delta x.a)\ b & \rightarrow_{\underline{\mu_1}} \Delta y.a[x := \lambda w.y\ (w\ b)] & y, w \notin \mathsf{FV}(b), y \neq w \\ \Delta x.x\ a & \rightarrow_{\underline{\mu_2}} a & \textit{if } x \notin \mathsf{FV}(a) \\ \Delta x.x\ (\Delta y.x\ a) & \rightarrow_{\underline{\mu_3}} a & \textit{if } x, y \notin \mathsf{FV}(a) \end{array}$$

4. *σ-reduction $\rightarrow_{\sigma}$ is defined as the compatible closure of*

$$\begin{array}{lll} x[x := b] & \rightarrow_{\sigma} b & \\ y[x := b] & \rightarrow_{\sigma} y & \textit{if } x \neq y \\ (a\ a')[x := b] & \rightarrow_{\sigma} (a[x := b])\ (a'[x := b]) & \\ (\mathcal{O}y.a)[x := b] & \rightarrow_{\sigma} \mathcal{O}y.(a[x := b]) & \textit{if } y \notin \mathsf{FV}(b) \end{array}$$

5. *$\rightarrow_{\underline{\beta\mu}\sigma} = (\rightarrow_{\underline{\beta}} \cup \rightarrow_{\underline{\mu}} \cup \rightarrow_{\sigma})$.*

Again we let $a, b, c, \ldots$ range over T^e and generalise α-conversion by taking $a[x := b]$ to be α-equal to $a'[y := b']$ if b' is α-equal to b and a' is α-equal to $a[y/x]$ for $y \notin \mathsf{FV}(a) \setminus \{x\}$. $\mathsf{FV}(a)$, $\mathsf{BV}(a)$ and VC over T^e are generalised by treating $a[x := b]$ as we would treat $\lambda x : b.a$ in typed λ-calculi. In particular, $\mathsf{FV}(a[x := b]) = \mathsf{FV}(b) \cup (\mathsf{FV}(a) \setminus \{x\})$, and when we write $(\Delta y.a)[x := b]$, it is assumed that $x \neq y$ and $x, y \notin \mathsf{FV}(b)$. Meta-substitution is generalised over T^e by adding the clause: $a[x := b][c/y] = a[c/y][x := b[c/y]]$.

If $x \notin \mathsf{FV}(a)$, the free variables of b are not substitutable free variables of $a[x := b]$. For this reason, we follow [8] and give the following definition:

Definition 6 *For any term a, we define the set of* substitutable free variables *of a, denoted $\sigma\mathsf{FV}(a)$ by the rules:*

$$\sigma\mathsf{FV}(x) = \{x\} \qquad \sigma\mathsf{FV}(ab) = \sigma\mathsf{FV}(a) \cup \sigma\mathsf{FV}(b) \qquad \sigma\mathsf{FV}(\mathcal{O}x.a) = \sigma\mathsf{FV}(a) \setminus \{x\}$$
$$\sigma\mathsf{FV}(a[x := b]) = \begin{cases} \sigma\mathsf{FV}(a) & \text{if } x \notin \mathsf{FV}(a) \\ (\sigma\mathsf{FV}(a) \setminus \{x\}) \cup \sigma\mathsf{FV}(b) & \text{if } x \in \mathsf{FV}(a) \end{cases}$$

Hence $\sigma\mathsf{FV}(x[y := z]) = \{x\}$ whereas $\mathsf{FV}(x[y := z]) = \{x, z\}$ for distinct x, y, z.

The following shows amongst other things that $\rightarrow_\sigma$ is SN and CR and that $\beta\mu$-reduction is sound in $\lambda\Delta\mathsf{exp}$. This will be used to show the simulation of $\beta\mu$-reduction and the CR of $\lambda\Delta\mathsf{exp}$.

Lemma 7 *Let $a, b \in T^e$. The following holds:*

1. *$\rightarrow_\sigma$ is SN and CR. Hence, every term $c \in T^e$ has a unique σ-normal form, denoted $\sigma(c)$.*
2. *$\sigma(ab) = \sigma(a)\sigma(b)$, $\sigma(\lambda x.a) = \lambda x.\sigma(a)$, $\sigma(\Delta x.a) = \Delta x.\sigma(a)$.*
3. *$\sigma(a[x := b]) = \sigma(a)[\sigma(b)/x]$*
4. *If $a \rightarrow_\sigma b$ then $\sigma(a) = \sigma(b)$ and $\sigma\mathsf{FV}(a) = \sigma\mathsf{FV}(b)$.*
5. *If $a \rightarrow_{\underline{\beta}} b$ then $\sigma(a) \twoheadrightarrow_\beta \sigma(b)$*
6. *If $a \rightarrow_{\underline{\mu}} b$ then $\sigma(a) \twoheadrightarrow_\mu \sigma(b)$*
7. *If $a \twoheadrightarrow_{\underline{\beta\mu}\sigma} b$ then $\sigma(a) \twoheadrightarrow_{\beta\mu} \sigma(b)$ and $\beta\mu(\sigma(b)) \leq \beta\mu(\sigma(a))$.*
8. *Let $y, w \notin \mathsf{FV}(b)$ and $y \neq w$. It holds that $\beta\mu(\sigma(\lambda w.y(wb'))) = \beta\mu(\sigma(b'))$ and that $b' \in \mathsf{SN}(\underline{\beta\mu}\sigma)$ iff $\lambda w.y(wb') \in \mathsf{SN}(\underline{\beta\mu}\sigma)$.*

Proof: The proofs of 1-5 are analogous to the proofs of the corresponding results for $\rightarrow_x$ (see [7]), whose rules are our σ-rules except the Δ-transition. We just remark that the function used to prove SN should be here extended with $h(\Delta x.a) = h(a) + 1$. 6 is by induction on a. The first half of 7 is a corollary of 4, 5 and 6 and implies the second half. 8 is straightforward. □

The following lemma shows that $\lambda\Delta\mathsf{exp}$ is powerful enough to simulate $\beta\mu$-reduction:

Lemma 8 (Simulation of $\beta\mu$-reduction) *For pure terms a, b: if $a \rightarrow_{\beta\mu} b$ then $a \twoheadrightarrow^+_{\underline{\beta\mu}\sigma} b$.*

Proof: By induction on a using Lemma 7.2 and .3. □

Theorem 9 *The $\lambda\Delta\mathsf{exp}$-calculus is confluent.*

Proof: We use the interpretation method [10, 16]. If $a \twoheadrightarrow_{\underline{\beta\mu}\sigma} b_1$ and $a \twoheadrightarrow_{\underline{\beta\mu}\sigma} b_2$ then by Lemma 7.7, $\sigma(a) \twoheadrightarrow_{\beta\mu} \sigma(b_i)$, for $i \in \{1, 2\}$, and by CR of $\lambda\Delta$, $\exists c$ such that $\sigma(b_i) \twoheadrightarrow_{\beta\mu} c$, and by Lemma 8 $\sigma(b_i) \twoheadrightarrow_{\underline{\beta\mu}\sigma} c$. Hence, $b_i \twoheadrightarrow_{\underline{\beta\mu}\sigma} c$. □

Finally, the following is the converse of the generalised PSN result we are aiming for:

Lemma 10 *Let $a \in T^e$. $a \in \mathsf{SN}(\underline{\beta\mu}\sigma) \Longrightarrow$ for all subterms b of a, $\sigma(b) \in \mathsf{SN}(\beta\mu)$,*

Proof: Assume $\exists b \lhd a$ where $\sigma(b) \notin \mathsf{SN}(\beta\mu)$ and let $\sigma(b) \rightarrow_{\beta\mu} b_1 \rightarrow_{\beta\mu} b_2 \rightarrow_{\beta\mu} \ldots$ be an infinite derivation. As $b \twoheadrightarrow_\sigma \sigma(b)$, then by Lemma 8, the derivation $b \twoheadrightarrow_{\underline{\beta\mu}\sigma} \sigma(b) \twoheadrightarrow^+_{\underline{\beta\mu}\sigma} b_1 \twoheadrightarrow^+_{\underline{\beta\mu}\sigma} b_2 \twoheadrightarrow^+_{\underline{\beta\mu}\sigma} \ldots$ is infinite. Absurd as $b \lhd a$ and $a \in \mathsf{SN}(\underline{\beta\mu}\sigma)$. □

Corollary 11 *If a is a pure term such that $a \in \mathsf{SN}(\underline{\beta\mu}\sigma)$, then $a \in \mathsf{SN}(\beta\mu)$.*

4 Preservation of Strong Normalisation

The question arises whether every term $a \in T$ which is in $\mathsf{SN}(\beta\mu)$ is also in $\mathsf{SN}(\underline{\beta\mu}\sigma)$ (i.e. whether $\lambda\Delta\mathsf{exp}$ preserves $\beta\mu$-strong normalisation). We start by defining two notions:

Definition 12

1. *A term $a \in T$ obeys the preservation of strong normalisation (PSN) property if:*
   $a \in \mathsf{SN}(\beta\mu) \Longrightarrow a \in \mathsf{SN}(\underline{\beta\mu}\sigma)$.

2. *A term $a \in T^e$ obeys the generalised preservation of strong normalisation (GPSN) property if:* $(\forall b \lhd a.\sigma(b) \in \mathsf{SN}(\beta\mu)) \Longrightarrow a \in \mathsf{SN}(\underline{\beta\mu}\sigma)$.

The GPSN property is a mild generalization of the PSN property. In our view, the GPSN property is more fundamental than the PSN property for two reasons:

1. the GPSN property applies to all terms, not only the pure ones;
2. for most typed λ-calculi with explicit substitutions, strong normalisation is an immediate consequence of the GPSN property and of strong normalisation of the standard calculus without explicit substitutions.

Lemma 13 *Let $a \in T$. a obeys GPSN $\Longleftrightarrow$ a obeys PSN.*

Proof: $\Longrightarrow$) Assume a obeys GPSN and $a \in \mathsf{SN}(\beta\mu)$. If $a \notin \mathsf{SN}(\underline{\beta\mu}\sigma)$ then, as a obeys GPSN, there exists $b \lhd a$ of a such that $\sigma(b) \notin \mathsf{SN}(\beta\mu)$. Since a is pure, b is pure and hence $b \notin \mathsf{SN}(\beta\mu)$. Therefore $a \notin \mathsf{SN}(\beta\mu)$. Absurd.
$\Longleftarrow$) Assume a obeys PSN and $\forall b \lhd a.\sigma(b) \in \mathsf{SN}(\beta\mu)$. As $a \lhd a$ and a is pure, we get $a \in \mathsf{SN}(\beta\mu)$. Hence, as a obeys PSN, $a \in \mathsf{SN}(\underline{\beta\mu}\sigma)$ and so a obeys GPSN. □

4.1 Structure-preserving calculi

In a recent paper [9], Bloo and Rose describe how to construct an explicit substitution CRS from an arbitrary CRS.[7] Moreover they show that PSN holds for a restricted class of CRSs, which they call structure-preserving.

Unfortunately, PSN for the $\lambda\Delta\mathsf{exp}$-calculus cannot be derived from [9]. Indeed, the first μ-rewrite rule is written in the CRS framework as $(\mu x.X(x))\ Y \to \mu y.X(\lambda w.y\ (w\ Y))$. The condition of structure-preserving requires the argument $\lambda w.y\ (w\ Y)$ of the meta-application in the right-hand side to be a subterm of the left-hand side. Obviously this is not the case.

It is interesting to notice that it is decidable whether a CRS is structure-preserving: one only needs to look at the rules and ensure that certain syntactic conditions are satisfied. In the case of the $\lambda\Delta\mathsf{exp}$ calculus, one must know the behavior of the whole reduction relation to prove PSN. In particular one must know that every redex in an instance $\lambda w.y\ (w\ a)$ of $\lambda w.y\ (w\ Y)$ can be traced back and thus must occur in a.

4.2 Proving PSN with the decency method

We shall prove that the $\lambda\Delta\mathsf{exp}$-calculus preserves strong normalization using the technique of decency introduced in [6] to prove that $\lambda\mathsf{exp}$ preserves β-SN. First, we define the following notion:

Definition 14 *We say that $[x := b]$ is* superfluous *in a if x is not a substitutable free variable of the term in the scope of $[x := b]$ in a (recall that we treat, from the point of view of binding, $c[x := b]$ as we would treat $\lambda x : b.c$). A reduction $a \to_{\underline{\beta\mu}\sigma} b$ is called* superfluous *if the contracted redex in a is part of d for $[x := d]$ superfluous in a.*

A superfluous reduction, similarly to the internal reduction of Definition 28, concentrates on reduction inside b for $[x := b]$ being a substitution item. A reduction is superfluous if the contracted redex is internal and the substitution item $[x := b]$ in which it occurs does not bind any substitution occurrence of x. The following lemma is similar to Lemma 29. Note that the use of $\twoheadrightarrow^+_{\beta\mu}$ rather than $\to_{\beta\mu}$ is due to the fact that a non superfluous redex is either external, or internal inside an $[x := d]$ where there are occurrences of x in a that are within the scope of $[x := d]$.

Lemma 15 *If $a \to_{\underline{\beta\mu}} b$ where a substitution $[x := d]$ is generated and the reduction $a \to_{\underline{\beta\mu_1}} b$ is not superfluous, then $\sigma(a) \twoheadrightarrow^+_{\beta\mu} \sigma(b)$.*

[7] The theory of Combinatory Reduction Systems was developed by J-W. Klop (see [20, 22]).

Proof: By induction on the structure of a using Lemmas 4 and 7. $\square$

The following is crucial to the GPSN proof. It imposes the condition that in *decent* terms, for any $[x := b]$, either $b \in \mathsf{SN}(\underline{\beta\mu}\sigma)$ or all $\beta\mu$-derivations starting at $\sigma(b)$ are finite.

Definition 16

- *A term a is called* decent *if for every $[x := b]$ in a, $b \in \mathsf{SN}(\underline{\beta\mu}\sigma)$.*
- *A term a is called* decent of order n *if for every $[x := b]$ in a, $b \in \mathsf{SN}(\underline{\beta\mu}\sigma)$ or $\beta\mu(\sigma(b)) < n$.*

Note that if a is decent and $\beta\mu(\sigma(a)) < \infty$, then a is decent of order $\beta\mu(\sigma(a)) + 1$.

Finally, the following notion of *ancestor*, aims to achieve similar conditions to those shown in Lemmas 26 and 27. It is related to what is referred to as "backtracking" in the minimal derivation method. Note that we use ")a" to denote an application item. For example, in $(\lambda x.a)b$ the application item is $)b$.[8]

Definition 17 *For a reduction $a \twoheadrightarrow_{\underline{\beta\mu}\sigma} a'$, we define the notion of* the ancestor *of a substitution item $[x := d]$ in a' as follows:*

- *If $a \rightarrow_{\underline{\beta\mu}\sigma} a'$ and $b = b'$ or if $b \rightarrow_{\underline{\beta\mu}\sigma} b'$ and $a = a'$ then $[x := b']$ in $a'[x := b']$ has ancestor $[x := b]$ in $a[x := b]$.*
- *In the following reductions, the first underlined item (which may be an application written ")."") is ancestor of the second underlined item:*

$$\begin{array}{lll}
(bc)\underline{[x := a]} & \rightarrow_{\underline{\beta\mu}\sigma} & (b\underline{[x := a]})c[x := a] \\
(bc)\underline{[x := a]} & \rightarrow_{\underline{\beta\mu}\sigma} & (b[x := a])c\underline{[x := a]} \\
(\mathcal{O}y.b)\underline{[x := a]} & \rightarrow_{\underline{\beta\mu}\sigma} & \mathcal{O}y.b\underline{[x := a]} \\
((\lambda x.b\underline{)a}) & \rightarrow_{\underline{\beta\mu}\sigma} & b\underline{[x := a]} \\
((\Delta x.a\underline{)b}) & \rightarrow_{\underline{\beta\mu}\sigma} & \Delta y.a\underline{[x := \lambda w.y(wb)]}
\end{array}$$

- *The ancestor relation behaves as expected in the confrontation with σ-reductions; i.e., if $\xi[x := a]$ is a context in which $[x := a]$ appears, then:*

$$\begin{array}{lll}
(\lambda y.b)\xi\underline{[x := a]} & \rightarrow_{\underline{\beta\mu}\sigma} & b[y := \xi\underline{[x := a]}] \\
(\Delta y.b)\xi\underline{[x := a]} & \rightarrow_{\underline{\beta\mu}\sigma} & \Delta z.b[y := \lambda w.z(w\xi\underline{[x := a]})] \\
(\lambda y.\xi\underline{[x := a]})b & \rightarrow_{\underline{\beta\mu}\sigma} & \xi\underline{[x := a]}[y := b] \\
(\Delta y.\xi\underline{[x := a]})b & \rightarrow_{\underline{\beta\mu}\sigma} & \Delta z.\xi\underline{[x := a]}[y := \lambda w.z(wb)] \\
(\mathcal{O}y.\xi\underline{[x := a]})[z := b] & \rightarrow_{\underline{\beta\mu}\sigma} & \mathcal{O}y.\xi\underline{[x := a]}[z := b] \\
(\mathcal{O}y.b)[z := \xi\underline{[x := a]}] & \rightarrow_{\underline{\beta\mu}\sigma} & \mathcal{O}y.b[z := \xi\underline{[x := a]}] \\
(bc)[z := \xi\underline{[x := a]}] & \rightarrow_{\underline{\beta\mu}\sigma} & b[z := \xi\underline{[x := a]}]c[z := \xi\underline{[x := a]}] \\
(b\xi\underline{[x := a]})[y := c] & \rightarrow_{\underline{\beta\mu}\sigma} & b[y := c]\xi\underline{[x := a]}[y := c] \\
(\xi\underline{[x := a]}b)[y := c] & \rightarrow_{\underline{\beta\mu}\sigma} & \xi\underline{[x := a]}[y := c]b[y := c]
\end{array}$$

- *The ancestor relation is compatible; e.g.: if $a \rightarrow_{\underline{\beta\mu}\sigma} a'$ where $[x := b']$ in a' has ancestor $[x := b]$ resp., $)b$ in a, and if $c \rightarrow_{\underline{\beta\mu}\sigma} c'$ then $[x := b']$ in $a'c'$ has ancestor $[x := b]$ resp., $)b$ in ac.*

The ancestor notion gives a full characterisation of how a substitution item might have been generated. It achieves the same aims of Lemmas 26 and 27 but in an alternative way.

Lemma 18 *If $a \rightarrow_{\underline{\beta\mu}\sigma} a'$ and $[x := b']$ is in a', then one of the following holds:*

- *Exactly one $[x := b]$ in a is an ancestor of $[x := b']$ in a' and $b \twoheadrightarrow_{\underline{\beta\mu}\sigma} b'$.*
- *$[x := b']$ has an application item $)b$ as ancestor with $b = b'$ or $b' = \lambda w.y(wb)$ for some $y, w \notin \mathsf{FV}(b)$ and $y \neq w$.*

[8] One can even go further as in [18] by calling λx the λ item but this is not needed here.

Proof: By induction on the structure of a. □

The following is informative about the subterms b of a term a that are not part of substitution items $[y := d]$ in a. It says that for any such b, performing some meta-substitutions on $\sigma(b)$ results in a subterm of $\sigma(a)$. Moreover, if $b = (\mathcal{O}x.b')c$ and if $\beta\mu(\sigma(a)) < \infty$, then $\beta\mu(\sigma(c)) < \beta\mu(\sigma(a))$.

Lemma 19

1. *If b is a subterm of a, b is not part of d for $[y := d]$ in a, then $\exists m, x_1, \dots x_m,\ c_1, \dots c_m$ such that $\sigma(b)[c_1/x_1][c_2/x_2]\dots[c_m/x_m]$ is a subterm of $\sigma(a)$.*
2. *If $(\mathcal{O}x.b)c$ is a subterm of a which is not part of d for any $[y := d]$ in a, and if $\beta\mu(\sigma(a)) < \infty$ then $\beta\mu(\sigma(c)) < \beta\mu(\sigma(a))$.*

Proof: 1: By induction on the structure of a. 2: $(\mathcal{O}x.\sigma(b))\sigma(c)[c_1/x_1]\dots[c_m/x_m]$ is a subterm of $\sigma(a)$ for some $c_i, x_i, 1 \leq i \leq m$, by 1 and Lemma 7. Hence, using Lemma 4.5, $\beta\mu(((\mathcal{O}x.\sigma(b))\sigma(c))) \leq \beta\mu(\sigma(a))$. Now, by Lemma 3, as $(\mathcal{O}x.\sigma(b))\sigma(c)$ is pure, $\beta\mu(\sigma(c)) < \beta\mu(\sigma(a))$. □

The following lemma is the key to proving GPSN. It says that any $\underline{\beta\mu}\sigma$ reduct a' of a decent term a whose σ-normal form has no infinite $\beta\mu$-derivations, is itself decent and its σ-normal form has no infinite $\beta\mu$-derivations.

Lemma 20 *If a is a term such that $\beta(\sigma(a)) < \infty$, a is decent, then for any $\underline{\beta\mu}\sigma$-reduct a' of a, a' is decent of order $\beta\mu(\sigma(a))$.*

Proof: By induction on the number of reduction steps in $a \twoheadrightarrow_{\underline{\beta\mu}\sigma} a'$. If $a \twoheadrightarrow_{\underline{\beta\mu}\sigma} a$ then as a is decent, a is decent of order $\beta\mu(\sigma(a))$.

Assume $a \twoheadrightarrow_{\underline{\beta\mu}\sigma} a'' \rightarrow_{\underline{\beta\mu}\sigma} a'$ where a'' is decent of order $\beta\mu(\sigma(a))$.

Let $[x := b]$ in a'. We must show that $b \in \mathsf{SN}(\underline{\beta\mu}\sigma)$ or $\beta\mu(\sigma(b)) < \beta\mu(\sigma(a))$.

The ancestor of $[x := b]$ in a'' is either:

1. $[x := b']$ in a'' where $b' \twoheadrightarrow_{\underline{\beta\mu}\sigma} b$
2. $)b$ in a'' and $(\lambda x.c)b \rightarrow_{\underline{\beta\mu}\sigma} c[x := b]$ is the contracted redex in $a'' \rightarrow_{\underline{\beta\mu}\sigma} a'$.
3. $)b'$ in a'' where $(\Delta x.c)b' \rightarrow_{\underline{\beta\mu}\sigma} c[x := \lambda w.y(wb')]$ is the contracted redex in $a'' \rightarrow_{\underline{\beta\mu}\sigma} a'$ and $b = \lambda w.y(wb')$.

In the first case, as a'' is decent of order $\beta\mu(\sigma(a))$, then either $b' \in \mathsf{SN}(\underline{\beta\mu}\sigma)$ or $\beta\mu(\sigma(b')) < \beta\mu(\sigma(a))$. Hence, $b \in \mathsf{SN}(\underline{\beta\mu}\sigma)$ or $\beta\mu(\sigma(b)) \leq \beta\mu(\sigma(b')) < \beta\mu(\sigma(a))$ using Lemma 7.

In the second case, if $)b$ is not part of d for some $[y := d]$ in a'', then by Lemma 19, as $\beta\mu(\sigma(a'')) < \infty$, $\beta\mu(\sigma(b)) < \beta\mu(\sigma(a'')) \leq \beta\mu(\sigma(a))$ by Lemma 7. If $)b$ is part of d for some $[y := d]$ in a'', then we may assume that there is no $[z := e]$ such that $)b$ is part of e and $[z := e]$ is part of d. Then as a'' is decent, either $d \in \mathsf{SN}(\underline{\beta\mu}\sigma)$ or $\beta\mu(\sigma(d)) < \beta\mu(\sigma(a''))$. If $d \in \mathsf{SN}(\underline{\beta\mu}\sigma)$ then $b \in \mathsf{SN}(\underline{\beta\mu}\sigma)$. If $\beta\mu(\sigma(d)) < \beta\mu(\sigma(a'')) \leq \beta\mu(\sigma(a))$ then as $(\lambda x.c)b$ is not part of some $[z := e]$ in d, we get by Lemma 19 that $\beta\mu(\mu\sigma(b)) < \beta\mu(\mu\sigma(d))$. Hence, $\beta\mu(\mu\sigma(b)) < \beta\mu(\mu\sigma(a))$.

The third case is similar to the second but note that $\beta\mu(\sigma(\lambda w.y(wb'))) = \beta\mu(\mu\sigma(b'))$ by Lemma 7, and $b' \in \mathsf{SN}(\underline{\beta\mu}\sigma)$ iff $\lambda w.y(wb') \in \mathsf{SN}(\underline{\beta\mu}\sigma)$. □

Finally, any decent term whose σ-normal form does not have an infinite $\beta\mu$-derivation, is itself $\underline{\beta\mu}\sigma$-strongly normalising:

Theorem 21 *If a is a term such that $\beta\mu(\sigma(a)) < \infty$ and a is decent, then $a \in \mathsf{SN}(\underline{\beta\mu}\sigma)$.*

Proof: By induction on $\beta\mu(\sigma(a)) < \infty$. We only treat the inductive case. Take a decent a such that $\beta\mu(\sigma(a)) < \infty$ and $\forall$ decent a' where $\beta\mu(\sigma(a')) < \beta\mu(\sigma(a))$ we have $a' \in \mathsf{SN}(\underline{\beta\mu}\sigma)$.

By Lemma 20, all $\rightarrow_{\underline{\beta\mu}\sigma}$-reducts of a are decent of order $\beta\mu(\sigma(a))$. Let us show that $a \in \mathsf{SN}(\underline{\beta\mu}\sigma)$. Assume the contrary and take an infinite derivation $a \rightarrow_{\underline{\beta\mu}\sigma} a_1 \rightarrow_{\underline{\beta\mu}\sigma} a_2 \dots$.
As σ is SN (Lemma 7), this derivation can be written as $a \twoheadrightarrow_\sigma b_1 \rightarrow_{\underline{\beta\mu}} c_1 \twoheadrightarrow_\sigma b_2 \rightarrow_{\underline{\beta\mu}} c_2 \dots$.
Again by Lemma 7, $\sigma(a) = \sigma(b_1) \twoheadrightarrow_{\beta\mu} \sigma(c_1) \twoheadrightarrow_{\beta\mu} \sigma(c_2) \twoheadrightarrow_{\beta\mu} \dots$.
By Lemma 15 and the fact that $\beta\mu(\sigma(a)) < \infty$, only finitely many of the reductions $b_m \rightarrow_{\underline{\beta\mu}} c_m$ are not superfluous. Otherwise, we will have an infinite $\beta\mu$-derivation starting at $\sigma(a)$ which is

impossible since $\beta\mu(\sigma(a)) < \infty$. Let $b_M \rightarrow_{\underline{\beta\mu}} c_M$ be the last non-superfluous $\rightarrow_{\underline{\beta\mu}}$-reduction and define h_2 as follows:

$$\begin{array}{llll} h_2(x) & = 1 & h_2(ab) & = h_2(a) + h_2(b) + 1 \\ h_2(\mathcal{O}x.b) & = h_2(b) + 1 & h_2(a[x := b]) & = \begin{cases} h_2(a).(h_2(b) + 2) & \text{if } x \in \sigma\mathsf{FV}(b) \\ 2h_2(a) & \text{otherwise} \end{cases} \end{array}$$

It is easy to prove by induction on the structure of terms that:

– If $a \rightarrow_{\underline{\beta\mu}\sigma} b$ is superfluous then $h_2(a) = h_2(b)$
– If $a \rightarrow_\sigma b$ is not superfluous then $h_2(a) > h_2(b)$.

Now, $\exists N > M$ such that $\forall n \geq N$, $h_2(c_n) = h_2(c_N)$, as $\forall n > M$, $b_n \rightarrow_{\underline{\beta\mu_1}} c_n$ is superfluous. Hence, $h_2(b_n) = h_2(c_n)$. Moreover, $h_2(d) < \infty$ for any term d.

Now, look at the part of the derivation: $c_N \twoheadrightarrow_\sigma b_{N+1} \rightarrow_{\underline{\beta\mu}} c_{N+1} \twoheadrightarrow_\sigma \ldots$.
We know that in this derivation, all $\underline{\beta\mu}$-reduction steps are superfluous. As $\forall n \geq N$, $h_2(c_n) = h_2(c_N) = h_2(b_n) = h_2(b_{n+1})$, it must be also the case that $c_n \twoheadrightarrow_\sigma b_{n+1}$ is superfluous for all $n \geq N$, otherwise, $h_2(c_n) > h_2(b_{n+1})$, contradiction.

Hence, one $[x := d]$ in c_N has an infinite $\underline{\beta\mu}\sigma$-derivation. Otherwise, there wouldn't be an infinite $\underline{\beta\mu}\sigma$-derivation starting at c_N, contradicting infinity of $c_N \twoheadrightarrow_\sigma b_{N+1} \rightarrow_{\underline{\beta\mu}} c_{N+1} \ldots$.
Now, take one innermost $[x := d]$ in c_N which has an infinite $\underline{\beta\mu}\sigma$-derivation. Then d is decent. As c_N is a $\underline{\beta\mu}\sigma$-reduct of a, then c_N is decent of order $\beta\mu(\sigma(a))$ by Lemma 20. Moreover, $\beta\mu(\sigma(d)) < \beta\mu(\sigma(a))$.

Hence, by IH, as $\beta\mu(\sigma(d)) < \beta\mu(\sigma(a))$ and d is decent, we get that $d \in \mathsf{SN}(\underline{\beta\mu}\sigma)$. Absurd. □

Now, the proof of GPSN is immediate:

Theorem 22 (Generalised Preservation of Strong Normalisation)
Let $a \in \mathcal{T}^e$, if every subterm b of a satisfies $\sigma(b) \in \mathsf{SN}(\beta\mu)$, then $a \in \mathsf{SN}(\underline{\beta\mu}\sigma)$.

Proof: By induction on the structure of a. As a is a subterm of a, then $\sigma(a) \in \mathsf{SN}(\beta\mu)$ and so $\beta\mu(\sigma(a)) < \infty$. Let $[x := b]$ in a where IH holds for b. Then $b \in \mathsf{SN}(\underline{\beta\mu}\sigma)$ and hence a is decent. So by Theorem 21, $a \in \mathsf{SN}(\underline{\beta\mu}\sigma)$. □

5 A type-assignment for $\lambda\Delta$exp

In [30], a classical type-assignment system for $\lambda\Delta$ is presented. The type-assignment system is simply typed, with a specific type $\perp$ standing for absurdity. Δ is typed with double negation.

Definition 23

1. *The set of types is given by the abstract syntax: $\mathcal{T} = \perp \mid \mathcal{T} \rightarrow \mathcal{T}$*
2. *A variable declaration is a pair $x : A$ where $x \in V$ and $A \in \mathcal{T}$.*
3. *A context is a finite list of declarations $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ such that $i \neq j \Rightarrow x_i \neq x_j$. If $\Gamma = x_1 : A_1, \ldots, x_n : A_n$ is a context, $B \in \mathcal{T}$ and x does not occur in Γ, then $\Gamma, x : B$ is used to denote the context $x_1 : A_1, \ldots, x_n : A_n, x : B$.*
4. *The set of contexts is denoted by $\mathcal{C}$.*
5. *The derivability relation $\vdash_{\beta\mu} \subseteq \mathcal{C} \times T \times \mathcal{T}$ is defined as follows (using the standard notation):*

$$(var)\quad \frac{}{\Gamma \vdash_{\beta\mu} x : A}\ \textit{if}\ (x : A) \in \Gamma \qquad\qquad (\lambda)\quad \frac{\Gamma, x : A \vdash_{\beta\mu} a : B}{\Gamma \vdash_{\beta\mu} \lambda x.a : A \rightarrow B}$$

$$(ap)\quad \frac{\Gamma \vdash_{\beta\mu} a : A \rightarrow B \quad \Gamma \vdash_{\beta\mu} b : A}{\Gamma \vdash_{\beta\mu} a\ b : B} \qquad\qquad (\Delta)\quad \frac{\Gamma, x : A \rightarrow \perp \vdash_{\beta\mu} a : \perp}{\Gamma \vdash_{\beta\mu} \Delta x.a : A}$$

6. *The derivability relation $\vdash_{\underline{\beta\mu}\sigma} \subseteq \mathcal{C} \times T^e \times \mathcal{T}$ is defined by the above rules and the new rule:*

$$(subst)\quad \frac{\Gamma, x : A \vdash_{\underline{\beta\mu}\sigma} a : B \quad \Gamma \vdash_{\underline{\beta\mu}\sigma} b : A}{\Gamma \vdash_{\underline{\beta\mu}\sigma} a[x := b] : B}$$

The following lemma establishes three basic properties:

Lemma 24

1. *Subject Reduction: if* $\Gamma \vdash_{\underline{\beta\mu}\sigma} a : A$ *and* $a \rightarrow_{\underline{\beta\mu}\sigma} b$*, then* $\Gamma \vdash_{\underline{\beta\mu}\sigma} b : A$.
2. *Conservativity: if* $\Gamma \vdash_{\underline{\beta\mu}\sigma} a : A$ *then* $\Gamma \vdash_{\beta\mu} \sigma(a) : A$.
3. *Closure under subterms: every subterm of a well-typed term is well-typed.*

Proof: By an easy induction on the derivation of $\Gamma \vdash_{\underline{\beta\mu}\sigma} a : A$. □

The following proposition establishes that the simply typed version of $\lambda\Delta$exp is SN. Its proof is simple thanks to the generalised PSN.

Proposition 25

1. *If* $\Gamma \vdash_{\beta\mu} a : A$*, then* $a \in \mathsf{SN}(\beta\mu)$.
2. *If* $\Gamma \vdash_{\underline{\beta\mu}\sigma} a : A$*, then* $a \in \mathsf{SN}(\underline{\beta\mu}\sigma)$.

Proof: 1. is proved in [30]. 2: assume a is a term of minimal length such that $\Gamma \vdash_{\underline{\beta\mu}\sigma} a : A$ and $a \notin \mathsf{SN}(\underline{\beta\mu}\sigma)$. By Lemma 24.2 and 1 above, $\sigma(a) \in \mathsf{SN}(\beta\mu)$. By GPSN (Theorem 35), a must therefore contain a strict subterm b such that $\sigma(b) \notin \mathsf{SN}(\beta\mu)$. By Lemma 8, $\rightarrow_{\beta\mu} \subseteq \twoheadrightarrow_{\underline{\beta\mu}\sigma}$, hence it follows that $\sigma(b) \notin \mathsf{SN}(\underline{\beta\mu}\sigma)$ and so $b \notin \mathsf{SN}(\underline{\beta\mu}\sigma)$. By Lemma 24.3, b is a well-typed term. This contradicts the minimality of a. □

6 Conclusion

We have introduced a calculus of explicit substitutions $\lambda\Delta$exp for the calculus $\lambda\Delta$ and proved that PSN holds. Moreover we have shown that the typed $\lambda\Delta$exp-calculus is strongly normalizing using a new method based on a mild generalization of PSN.

To our knowledge, $\lambda\Delta$exp is the first calculus with explicit substitutions which is not structure-preserving. Its study has revealed two importants points:

1. one may be able to prove PSN for a class of CRSs substantially bigger than the class of structure-preserving CRSs.
2. not all approaches to prove PSN for λ-calculi with explicit substitutions are generalisable.

We are currently investigating whether the minimal derivation technique could prove useful in generalizing the results of [9]. It would be interesting to provide some general conditions for a CRS to have PSN. Ideally one would be able to provide some very weak (probably undecidable) conditions equivalent to PSN. The condition of structure-preserving will then appear as a specialization of these conditions.

Another area which remains open is explicit substitutions for non-local control operators: the Δ-operator considered in this paper is compatible (reduction rules apply in all contexts) and local (does not refer to contexts). Some other control operators are non-local and manipulate contexts. It remains a challenge to determine whether such calculi with non-local control operators have the PSN property. Interestingly, such calculi will require an explicit handling of contexts so their explicit variants will probably be equational term-rewriting systems (the equational part taking care of contexts). This subject is left for future work.

References

1. M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit Substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.
2. A. W. Appel. *Compiling with Continuations.* Cambridge University Press, 1992.
3. F. Barbanera and S. Berardi. Continuations and simple types: A strong normalization result. In *ACM SIGPLAN Workshop on Continuations*, 1992.
4. H. Barendregt. *The Lambda Calculus : Its Syntax and Semantics.* North Holland, 1984.
5. Z. Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. $\lambda\upsilon$, a calculus of explicit substitutions which preserves strong normalisation. *Journal of Functional Programming*, 1995.

6. R. Bloo. Preservation of Strong Normalisation for Explicit Substitution. Technical Report CS-95-08, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1995.
7. R. Bloo and H. Geuvers. Explicit substitution: On the edge of strong normalisation. Technical Report CS-96-10, Department of Mathematics and Computing Science, Eindhoven University of Technology, 1996.
8. R. Bloo and K. Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. *Computer Science in the Netherlands*, 1995.
9. R. Bloo and K. Rose. Combinatory reduction systems with explicit substitutions that preserve strong normalisation. *RTA '96*, 1996. To appear.
10. P.-L. Curien, T. Hardin, and J.-J. Lévy. Confluence properties of weak and strong calculi of explicit substitutions. Technical Report RR 1617, INRIA, Rocquencourt, 1992. To appear in the JACM.
11. P. de Groote. On the relation between the $\lambda\mu$-calculus and the syntactic theory of sequential control. In *Logic Programming and Automated Reasoning*, volume 822 of *Lecture Notes in Computer Science*, pages 31–43. Springer-Verlag, 1994.
12. G. Dowek, T. Hardin, and C. Kirchner. Higher-order unification via explicit substitutions. In *Proceedings of the Tenth Annual Symposium on Logic in Computer Science*, pages 366–374. IEEE Computer Society Press, 1995.
13. R.K. Dybvig. *The Scheme Programming Language.* Prentice-Hall, 1987.
14. R. Constable et al. *Implementing Mathematics with the NUPRL Development System.* Prentice-Hall, 1986.
15. T.G. Griffin. A formulae-as-types notion of control. In *Principles of Programming Languages*, pages 47–58. ACM Press, 1990.
16. T. Hardin. Confluence Results for the Pure Strong Categorical Logic CCL : λ-calculi as Subsystems of CCL. *Theoretical Computer Science*, 65(2):291–342, 1989.
17. H. Herbelin. *Elimination des coupures dans les sequents qu'on calcule.* PhD thesis, Université de Paris 7, 1994.
18. F. Kamareddine and R. P. Nederpelt. A useful λ-notation. *Theoretical Computer Science*, 155:85–109, 1996.
19. F. Kamareddine and A. Ríos. A λ-calculus à la de Bruijn with explicit substitutions. Proceedings of PLILP'95. *Lecture Notes in Computer Science*, 982:45–62, 1995.
20. J.-W. Klop. Combinatory Reduction Systems. *Mathematical Center Tracts*, 27, 1980.
21. J.-W. Klop. Term rewriting systems. *Handbook of Logic in Computer Science*, II, 1992.
22. J.-W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: Introduction and survey. *Theoretical Computer Science*, 121:279–308, 1993.
23. J.-L. Krivine. Classical logic, storage operators, and second-order λ-calculus. *Annals of Pure and Applied Logic*, 68:53–78, 1994.
24. L. Magnusson. *The implementation of ALF: a proof editor based on Martin-Löf's monomorphic type theory with explicit substitution.* PhD thesis, Department of Computer Science, Chalmers University, 1994.
25. P.-A. Melliès. Typed λ-calculi with explicit substitutions may not terminate, in *proceedings of tlca'95. Lecture Notes in Computer Science*, 902, 1995.
26. C. Muñoz. Confluence and preservation of strong normalisation in an explicit substitutions calculus. Technical report, INRIA, Rocquencourt, 1995. To appear in LICS '96.
27. C. Muñoz. Proof representation in type theory: State of the art. Accepted to be presented in the XXII Latinamerican Conference of Informatics CLEI Panel 96, June 3–7, 1996, Santafé de Bogotá, Colombia, April 1996.
28. C. Murthy. *Extracting Constructive Contents from Classical Proofs.* PhD thesis, Cornell University, 1990.
29. M. Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In *International Conference on Logic Programming and Automated Reasoning*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer-Verlag, 1992.
30. N.J. Rehof and M.H. Sørensen. The λ_Δ calculus. In M. Hagiya and J. Mitchell, editors, *Theoretical Aspects of Computer Software*, volume 789 of *Lecture Notes in Computer Science*, pages 516–542. Springer-Verlag, 1994.
31. G. L. Steele. *Common Lisp: The Language.* Digital Press, Bedford, MA, 1984.
32. A. Tasistro. Formulation of Martin-Löf's theory of types with explicit substitutions. Master's thesis, Chalmers University, 1993.
33. J. Underwood. *Aspects of the computational content of proofs.* PhD thesis, Cornell University, 1994.

A Proving PSN with the minimal derivation technique

We shall prove that $\lambda\Delta\mathsf{exp}$ preserves strong normalization using the technique of minimal derivations used in [5] to prove that $\lambda\upsilon$ preserves β-SN and in [19] to prove that λs preserves β-SN. First, we backtrack the substitutions operators in the derivations via the following two lemmas.

Lemma 26 *Let $a, c, d \in T^e$ and C a context, such that $a \to_{\underline{\beta\mu}\sigma} C\{c[x := d]\}$, then one of the following must hold:*

1. *$a = C\{(\lambda x.c)d\}$*
2. *$\exists b \in T^e$ and a context C' such that $a = C'\{(\Delta x.c)b\}$, $d = \lambda w.y(wb)$ and $C'\{\Delta y.\{\ \}\} = C$.*
3. *$\exists c', d' \in T^e$ and a context C' such that $a = C'\{c'[x := d']\}$ with $d' = d$ or $d' \to_{\underline{\beta\mu}\sigma} d$.*

Proof: By induction on a. □

Lemma 27 *Let $a_1 \to_{\underline{\beta\mu}\sigma} \ldots \to_{\underline{\beta\mu}\sigma} a_n \to_{\underline{\beta\mu}\sigma} a_{n+1} = C\{c[x := d]\}$, then there exists $c', d' \in T^e$ and a context C' such that one of the following must hold:*

1. *$a_1 = C'\{c'[x := d']\}$ with $d' \twoheadrightarrow_{\underline{\beta\mu}\sigma} d$.*
2. *$\exists k \le n$ such that $a_k = C'\{(\lambda x.c')d'\}$ and $a_{k+1} = C'\{c'[x := d']\}$ with $d' \twoheadrightarrow_{\underline{\beta\mu}\sigma} d$.*
3. *$\exists k \le n$ with $a_k = C'\{(\Delta x.c')d'\}$, $a_{k+1} = C'\{\Delta y.c'[x := \lambda w.y(wd')]\}$ and $\lambda w.y(wd') \twoheadrightarrow_{\underline{\beta\mu}\sigma} d$.*

Proof: Induction on the length of the derivation using the previous lemma. □

In order to apply the minimal derivation technique we define internal and external reductions. This has been done either by defining first internal and external positions as in [5] or by giving a direct inductive definition as in [19]. We choose here another equivalent but simpler presentation:

Definition 28 *A reduction $\longrightarrow$ over T^e is* internal*, denoted $a \xrightarrow{\mathtt{int}} b$, if there exists a context C such that $a = C\{c[x := d]\}$, $d \longrightarrow d'$ and $b = C\{c[x := d']\}$.*

A reduction over T^e is external*, denoted $a \xrightarrow{\mathtt{ext}} b$, when it is not internal.*

The following is a slight but essential variation of Lemma 7 cases 5 and 6. A step of *external* $\underline{\beta}$ and $\underline{\mu}$ is studied and the lemma ensures that we *have exactly one step of β- or μ-reduction* between the corresponding σ-normal forms.

Lemma 29 *Let a, $b \in T^e$.*

1. *If $a \xrightarrow{\mathtt{ext}}_{\underline{\beta}} b$ then $\sigma(a) \to_\beta \sigma(b)$.*
2. *If $a \xrightarrow{\mathtt{ext}}_{\underline{\mu}} b$ then $\sigma(a) \to_\mu \sigma(b)$.*

Proof: Induction on a in a similar fashion to cases 5 and 6 of Lemma 7. Now, the point is that in the case $a = c[x := d]$, the reduction cannot take place within d because it is external, and this is the only case that forced us to consider the reflexive-transitive closure in Lemma 7. □

The following lemma plays a fundamental role in Lemma 31 and hence in the Generalised Preservation theorem. Its proof follows the lines of the Commutation Lemma given in [19].

Lemma 30 (Commutation Lemma) *Let a, $b \in T^e$ such that $\sigma(a) \in \mathsf{SN}(\beta\mu)$ and $\sigma(a) = \sigma(b)$. If $a \xrightarrow{\mathtt{int}}_{\underline{\beta\mu}\sigma} \cdot \xrightarrow{\mathtt{ext}}_{\sigma} b$ then $a \xrightarrow{\mathtt{ext}}{}^{+}_{\sigma} \cdot \overset{\mathtt{int}}{\twoheadrightarrow}_{\underline{\beta\mu}\sigma} b$.*

Proof: By a careful induction on a, analysing the positions of the redexes. □

Lemma 31 *Let $a \in T^e$ such that for every subterm b of a, $\sigma(b) \in \mathsf{SN}(\beta\mu)$. For every infinite $\underline{\beta\mu}\sigma$-derivation $a \to_{\underline{\beta\mu}\sigma} b_1 \to_{\underline{\beta\mu}\sigma} \cdots \to_{\underline{\beta\mu}\sigma} b_n \to_{\underline{\beta\mu}\sigma} \cdots$, there exists N such that for $i \ge N$ all the reductions $b_i \to_{\underline{\beta\mu}} b_{i+1}$ are internal.*

Proof: The proof follows the lines of Lemma 16 in [19] and requires the Commutation Lemma. $\square$

In order to prove the Generalised Preservation Theorem we need two definitions.

Definition 32 *An infinite $\underline{\beta\mu}\sigma$-derivation $\mathcal{D}$: $a_1 \rightarrow_{\underline{\beta\mu}\sigma} \cdots \rightarrow_{\underline{\beta\mu}\sigma} a_n \rightarrow_{\underline{\beta\mu}\sigma} \cdots$ is* minimal *if for every step of reduction $a_i \rightarrow_{\underline{\beta\mu}\sigma} a_{i+1}$, every other derivation which contracts a redex that is a proper subterm of the redex contracted in $\mathcal{D}$ is finite.*

The intuitive idea of a minimal derivation is that if one rewrites at least one of its steps within a subterm of the actual redex, then an infinite derivation is impossible.

Definition 33 Skeletons *are defined by the following syntax:*

$$\textbf{Skeletons}\ K ::= V \mid K\ K \mid \lambda V.K \mid \Delta V.K \mid K[V := \square] \quad \textit{where } \square \textit{ is a fresh symbol}$$

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

$$\begin{array}{lll} Sk(x) = x & Sk(a\,b) = Sk(a)Sk(b) & Sk(a[x := b]) = Sk(a)[x := \square] \\ & Sk(\lambda x.a) = \lambda x.Sk(a) & Sk(\Delta x.a) = \Delta x.Sk(a) \end{array}$$

Remark 34 *If $a \xrightarrow{\texttt{int}}_{\underline{\beta\mu}\sigma} b$ then $Sk(a) = Sk(b)$.*

Theorem 35 (Generalised Preservation of strong normalisation)
Let $a \in T^e$, if every subterm b of a satisfies $\sigma(b) \in \mathsf{SN}(\beta\mu)$, then $a \in \mathsf{SN}(\underline{\beta\mu}\sigma)$.

Proof: Let us assume the existence of $a \in T^e$ with minimal length such that for every subterm b of a, $\sigma(b) \in \mathsf{SN}(\beta\mu)$ and $a \notin \mathsf{SN}(\underline{\beta\mu}\sigma)$. Let us consider a minimal infinite derivation $\mathcal{D}$: $a \rightarrow_{\underline{\beta\mu}\sigma} a_1 \rightarrow_{\underline{\beta\mu}\sigma} \cdots \rightarrow_{\underline{\beta\mu}\sigma} a_n \rightarrow_{\underline{\beta\mu}\sigma} \cdots$. By lemma 31, there exists N, such that for $i \geq N$, $a_i \rightarrow_{\underline{\beta\mu}\sigma} a_{i+1}$ is internal. Therefore, by the previous remark, $Sk(a_i) = Sk(a_{i+1})$ for $i \geq N$. As there are only a finite number of closures in $Sk(a_N)$ and as the reductions within these closures are independent, an infinite subderivation of $\mathcal{D}$ must take place within the same and unique closure in $Sk(a_N)$ and, evidently, this subderivation is also minimal. Let us call it $\mathcal{D}'$ and let C be the context such that $a_N = C\{c[x := d]\}$ and $c[x := d]$ is the closure where $\mathcal{D}'$ takes place. Therefore we have:

$$\mathcal{D}' \ :\ a_N = C\{c[x := d]\} \xrightarrow{\texttt{int}}_{\underline{\beta\mu}\sigma} C\{c[x := d_1]\} \xrightarrow{\texttt{int}}\!\!\!\!\rightarrow_{\underline{\beta\mu}\sigma} C\{c[x := d_n]\} \xrightarrow{\texttt{int}}_{\underline{\beta\mu}\sigma} \cdots$$

Lemma 27 gives rise to three possibilities:

1. $a = C'\{c'[x := d']\}$ with $d' \twoheadrightarrow_{\underline{\beta\mu}\sigma} d$.
2. $\exists I \leq N$ such that $a_I = C'\{(\lambda x.c')d'\} \rightarrow_{\underline{\beta\mu}\sigma} a_{I+1} = C'\{c'[x := d']\}$ and $d' \twoheadrightarrow_{\underline{\beta\mu}\sigma} d$.
3. $\exists I \leq N$: $a_I = C'\{(\Delta x.c')d'\} \rightarrow_{\underline{\beta\mu}\sigma} a_{I+1} = C'\{\Delta y.c'[x := \lambda w.y(wd')]\}$ and $\lambda w.y(wd') \twoheadrightarrow_{\underline{\beta\mu}\sigma} d$.

In the first case, we have an infinite derivation $d' \twoheadrightarrow_{\underline{\beta\mu}\sigma} d \rightarrow_{\underline{\beta\mu}\sigma} d_1 \twoheadrightarrow_{\underline{\beta\mu}\sigma} \ldots$, and since d' is a subterm of a, for every subterm c of d', $\sigma(c) \in \mathsf{SN}(\beta\mu)$ as well. This contradicts the fact that a was chosen with minimal length.
In the second case, let us consider the following derivation $\mathcal{D}''$:
$a \twoheadrightarrow_{\underline{\beta\mu}\sigma} a_I = C'\{(\lambda x.c')d'\} \twoheadrightarrow_{\underline{\beta\mu}\sigma} C'\{(\lambda x.c')d\} \rightarrow_{\underline{\beta\mu}\sigma} C'[(\lambda x.c')d_1] \twoheadrightarrow_{\underline{\beta\mu}\sigma} C'\{(\lambda x.c')d_n\} \rightarrow \cdots$
In the third case, since $\lambda w.y(wd') \twoheadrightarrow_{\underline{\beta\mu}\sigma} d$, necessarily $d = \lambda w.y(wb)$ with $d' \twoheadrightarrow_{\underline{\beta\mu}\sigma} b$. Furthermore, since $d \twoheadrightarrow^{+}_{\underline{\beta\mu}\sigma} d_i$, necessarily $d_i = \lambda w.y(wb_i)$ with $b \twoheadrightarrow^{+}_{\underline{\beta\mu}\sigma} b_i$. Consider $\mathcal{D}'''$:
$a \twoheadrightarrow_{\underline{\beta\mu}\sigma} a_I = C'\{(\Delta x.c')d'\} \twoheadrightarrow_{\underline{\beta\mu}\sigma} C'\{(\Delta x.c')b\} \rightarrow_{\underline{\beta\mu}\sigma} C'[(\Delta x.c')b_1] \twoheadrightarrow_{\underline{\beta\mu}\sigma} C'\{(\Delta x.c')b_n\} \rightarrow \cdots$
In $\mathcal{D}''$ (resp. $\mathcal{D}'''$) the redex in a_I is within d', a proper subterm of $(\lambda x.c')d'$ (resp. $(\Delta x.c')d'$), whereas in $\mathcal{D}$ the redex in a_I is $(\lambda x.c')d'$ (resp. $(\Delta x.c')d'$). This contradicts minimality of $\mathcal{D}$. $\square$

This article was processed using the LaTeX macro package with LLNCS style