Perpetuality for full and safe composition (in a constructive setting)

Delia Kesner

PPS, Université Paris Diderot

Abstract. We study perpetuality in calculi with explicit substitutions having *full composition*. A simple perpetual strategy is used to define strongly normalising terms inductively. This gives a simple argument to show preservation of β-strong normalisation as well as strong normalisation for typed terms. Particularly, the strong normalisation proof is based on implicit substitution rather than explicit substitution, so that it turns out to be modular w.r.t. the well-known proofs for typed lambda-calculus. All the proofs we develop are constructive.

1 Introduction

In calculi with *explicit* substitutions (ES) without composition rules, such as $\lambda\mathtt{x}$ [20, 23], outermost substitutions must be delayed until the total execution of all the innermost substitutions appearing in the same environment. Thus for example, the outermost substitution $[x/v]$ in the term $(zyx)[y/xx][x/v]$ must be delayed until $[y/xx]$ is first executed on zyx. This can be recovered by the use of composition rules which allow to propagate substitutions through (non pure) terms. Thus, $(zyx)[y/xx][x/v]$ can be reduced to $(zyx)[x/v][y/(xx)[x/v]]$, which can be further reduced to $(zyv)[y/vv]$, a term equal to $(zyx)[y/xx]\{x/v\}$, where $\{x/v\}$ denotes the standard *meta/implicit* substitution (on non pure terms) that the *explicit* substitution $[x/v]$ is supposed to implement.

Composition rules for ES first appeared in $\lambda\sigma$ [1]. They are used to get confluence on open terms [10, 11] when implementing higher-order unification [7] or functional abstract machines [19]. They guarantee a property, called *full composition*, that calculi without composition do not enjoy: any term of the form $t[x/u]$ can be reduced to $t\{x/u\}$, *i.e.* explicit substitution implements the implicit one.

Many calculi with ES such as $\lambda\sigma$ [1], $\lambda\sigma_{\Uparrow}$ [10], λ_{sub} [22], $\lambda\mathtt{lxr}$ [14] and $\lambda\mathtt{es}$ [11] enjoy full composition. However, $\lambda\sigma$ and $\lambda\sigma_{\Uparrow}$ do not enjoy neither strong normalisation (SN) for *typed* terms, nor preservation of β-strong normalisation (PSN) for *untyped* terms, a result which is a consequence of Melliès' counter-example [21]. But full composition and normalisation can live together, leading to a notion of *safe* composition; this is for example the case of λ_{sub}, $\lambda\mathtt{es}$ and $\lambda\mathtt{lxr}$. The available SN proofs for calculi with composition are indirect: either one simulates reduction by means of another well-founded relation, or SN is deduced from a sufficient property, as for example PSN. Proofs using the first technique are for example those for λ_{ws} [6] and $\lambda\mathtt{lxr}$ [14], based on the well-foundedness of the reduction relation for multiplicative exponential linear logic

(MELL) proof-nets [9]. An example of SN proof using the second technique is that for $\lambda\mathtt{es}$, where PSN is obtained by two consecutive translations, one from $\lambda\mathtt{es}$ into a calculus with ES and weakenings, the second one from this intermediate calculus into the Church-Klop's Λ_I-calculus [16]. In both cases the proofs are long, but especially not self-contained.

Since nothing indicates that calculi with safe composition could be only understood in terms of MELL proof-nets or the Λ_I-calculus, it will be then significant to provide independent arguments to prove normalisation properties for them. This would be useful, particularly, when integrating them inside other richer frameworks such as type theory.

The aim of this paper is to understand safe composition. For that, we choose to work with a simple calculus, that we call $\lambda\mathtt{ex}$, obtained by extending $\lambda\mathtt{x}$ with one rewriting rule for composition of *dependent* substitutions and one equation for commutation of *independent* substitutions. A similar calculus is studied in [24], where our equation is treated as a non-terminating reduction rule. The $\lambda\mathtt{ex}$-calculus uses *unary* constructors for substitutions but has the same expressive power than calculi with *n-ary* substitutions: thus for example $(xy)[y/x, x/y]$ can be implemented by the α-equivalent term $(wy)[y/x][w/y]$. Indeed, while simultaneous substitutions are specified by *lists* (given by n-ary substitutions) in calculi like $\lambda\sigma$, they are modelled by *sets* (given by commutation of independent unary substitutions) in $\lambda\mathtt{ex}$. The $\lambda\mathtt{ex}$-calculus is conceptually simple, it enjoys full composition and confluence on open-terms.

The technical tools used in this paper are the following. We first define a *perpetual* reduction strategy for $\lambda\mathtt{ex}$: if $t \notin \mathcal{SN}_{\lambda\mathtt{ex}}$ and t reduces to t' by the strategy, then $t' \notin \mathcal{SN}_{\lambda\mathtt{ex}}$. In particular, since the perpetual strategy reduces $t[x/u]$ to $t\{x/u\}$, one has to show that normalisation of **I***mplicit* substitution implies normalisation of **E***xplicit* substitution:

$$(\mathbf{IE})\ u \in \mathcal{SN}_{\lambda\mathtt{ex}}\ \&\ t\{x/u\} \in \mathcal{SN}_{\lambda\mathtt{ex}}\ \text{imply}\ t[x/u] \in \mathcal{SN}_{\lambda\mathtt{ex}}$$

In other words, ES implements implicit substitutions but nothing more than that, otherwise one may get calculi such as $\lambda\sigma$ where $t[x/u]$ does much more than $t\{x/u\}$ since it is able to behave like $t\{x/u\}$ but also to behave differently (for example by looping) before reducing to $t\{x/u\}$. A consequence of **(IE)** is that standard techniques to show SN based on *meta*-substitution can also be applied to calculi with ES, thus considerably simplifying the reasoning. Indeed, the perpetual strategy is used to give an inductive characterisation of the set $\mathcal{SN}_{\lambda\mathtt{ex}}$ by means of just four inference rules. This characterisation is then used to show that untyped terms enjoy PSN and typed terms enjoy SN. In particular, SN is shown by using arithmetical arguments: the proof is the one for simply typed λ-calculus but just adds the new case $t[x/u]$. In that sense we can say that our SN proof is modular w.r.t. the SN proof for typed lambda-terms. All our proofs are constructive in the sense that neither excluded middle nor double negation elimination are used. At the end of the paper we also show how SN of other calculi (with or without) full composition can be obtained from SN of $\lambda\mathtt{ex}$.

Perpetual strategies are studied for the non equational systems $\lambda\mathtt{x}$ in [3, 18, 15], and λ_{ws} in [2]. No abstract use of full composition can be done there. Current investigations carried out in [29] show PSN for different calculi with (full or not) composition. The approach is based on proofs by contradiction which analyse some *minimal* not terminating reduction sequence of the underlying calculus.

The paper is organised as follows. Section 2 introduces syntax and reduction rules. Perpetuality is studied in Section 3 and normalisation proofs are given in Section 4. Section 5 presents the labelling technique to show the **(IE)** property. In Section 6 we explain how to infer SN for other calculi with ES from our result in Section 4. We conclude and give directions for further work in Section 7.

Full details of the proofs in the paper are available on [12].

2 Syntax

The $\lambda\mathtt{ex}$-calculus can be viewed as the $\lambda\mathtt{x}$-calculus together with a safe composition rule for dependent substitutions and a commutativity equation for independent substitutions. The set of $\mathtt{x}$-terms is defined by:

$$\mathcal{T}_{\mathtt{x}} ::= x \mid \mathcal{T}_{\mathtt{x}}\ \mathcal{T}_{\mathtt{x}} \mid \lambda x.\mathcal{T}_{\mathtt{x}} \mid \mathcal{T}_{\mathtt{x}}[x/\mathcal{T}_{\mathtt{x}}]$$

Free and bound variables are defined as usual by assuming the terms $\lambda x.t$ and $t[x/u]$ bind x in t. The congruence generated by renaming of bound variables is called α-conversion. Thus for example $(\lambda y.x)[x/y] =_\alpha (\lambda z.w)[w/y]$. We use the notation $\overline{t_n}$ for a list of terms $t_1, \ldots, t_n$ and $u\overline{t_n}$ for $ut_1 \ldots t_n$ which is an abbreviation of $(\ldots((ut_1)t_2)\ldots\ldots t_n)$.

Meta-substitution on $\mathtt{x}$-terms is defined modulo α-conversion in such a way that capture of variables is avoided:

$$\begin{array}{ll} x\{x/v\} := v & (tu)\{x/v\} := t\{x/v\}u\{x/v\} \\ y\{x/v\} := y \text{ if } y \neq x & (\lambda y.t)\{x/v\} := \lambda y.t\{x/v\} \\ & t[y/u]\{x/v\} := t\{x/v\}[y/u\{x/v\}] \end{array}$$

Thus for example $(\lambda y.x)\{x/y\} = \lambda z.y$. Note that $t\{x/u\} = t$ if $x \notin \mathtt{fv}(t)$.

Besides α-conversion, we consider the following equations and rules.

Equations :			
$t[x/u][y/v]$	$=_{\mathtt{C}}$	$t[y/v][x/u]$	if $y \notin \mathtt{fv}(u)$ & $x \notin \mathtt{fv}(v)$
Rules :			
$(\lambda x.t)\ u$	$\to_{\mathtt{B}}$	$t[x/u]$	
$x[x/u]$	$\to_{\mathtt{Var}}$	u	
$t[x/u]$	$\to_{\mathtt{Gc}}$	t	if $x \notin \mathtt{fv}(t)$
$(tu)[x/v]$	$\to_{\mathtt{App}}$	$t[x/v]\ u[x/v]$	
$(\lambda y.t)[x/v]$	$\to_{\mathtt{Lamb}}$	$\lambda y.t[x/v]$	
$t[x/u][y/v]$	$\to_{\mathtt{Comp}}$	$t[y/v][x/u[y/v]]$	if $y \in \mathtt{fv}(u)$

The *rewritingrelation* generated by all the previous rules except $\mathtt{B}$ is denoted by $\mathtt{x}$. We write $\mathtt{Bx}$ for $\mathtt{B}\cup\mathtt{x}$. The *equivalence relation* generated by the conversions α and $\mathtt{C}$ is written $\mathtt{e}$. The *reduction relation* generated by the *rewriting relations modulo* $\mathtt{e}$ specify rewriting of $\mathtt{e}$-equivalence classes:

$$t \to_{\mathtt{ex}} t' \quad \text{iff} \quad \exists\ s, s' \text{ s.t. } t =_{\mathtt{e}} s \to_{\mathtt{x}} s' =_{\mathtt{e}} t'$$
$$t \to_{\lambda\mathtt{ex}} t' \quad \text{iff} \quad \exists\ s, s' \text{ s.t. } t =_{\mathtt{e}} s \to_{\mathtt{Bx}} s' =_{\mathtt{e}} t'$$

Note that all the equations and rules are assumed to avoid capture of variables by α-conversion. Thus for example we have $y \neq x$ and $y \notin \mathtt{fv}(v)$ in rule $\mathtt{Lamb}$. Same kind of assumptions are done for $\mathtt{Comp}$ and $\mathtt{C}$.

The notation $\to^*_{\lambda\mathtt{ex}}$ (resp. $\to^+_{\lambda\mathtt{ex}}$) is used for the reflexive and transitive (resp. transitive) closure of $\to_{\lambda\mathtt{ex}}$. Thus, if $t \to^*_{\lambda\mathtt{ex}} t'$ in 0 reduction steps, then $t =_{\mathtt{e}} t'$.

A term t is said to be in $\lambda\mathtt{ex}$-normal form, written $t \in \mathcal{NF}_{\lambda\mathtt{ex}}$, if there is no s such that $t \to_{\lambda\mathtt{ex}} s$. A term t is said to be $\lambda\mathtt{ex}$-strongly normalising, written $t \in \mathcal{SN}_{\lambda\mathtt{ex}}$, if there is no infinite $\lambda\mathtt{ex}$-reduction sequence starting at t, in which case $\eta_{\lambda\mathtt{ex}}(t)$ denotes the maximal length of a $\lambda\mathtt{ex}$-reduction sequence starting at t. A standard inductive definition of $\mathcal{SN}_{\lambda\mathtt{ex}}$ can be given by:

$$t \in \mathcal{SN}_{\lambda\mathtt{ex}} \text{ iff } \forall s\ (t \to_{\lambda\mathtt{ex}} s \text{ implies } s \in \mathcal{SN}_{\lambda\mathtt{ex}})$$

The following basic properties of $\lambda\mathtt{ex}$-reduction can be shown by a straightforward induction on the $\lambda\mathtt{ex}$-reduction relation.

Lemma 1 (Basic Properties).

- *If* $t \to_{\lambda\mathtt{ex}} t'$, *then* $\mathtt{fv}(t') \subseteq \mathtt{fv}(t)$.
- *For* $\mathcal{R} \in \{\mathtt{ex}, \lambda\mathtt{ex}\}$, *if* $t \to_{\mathcal{R}} t'$, *then* $u\{x/t\} \to^*_{\mathcal{R}} u\{x/t'\}$ *and* $t\{x/u\} \to_{\mathcal{R}} t'\{x/u\}$. *Thus in particular* $t\{x/u\} \in \mathcal{SN}_{\mathcal{R}}$ *implies* $t \in \mathcal{SN}_{\mathcal{R}}$.

The rule $\mathtt{Comp}$ and the equation $\mathtt{C}$ guarantee the following property:

Lemma 2 (Full composition). $t[x/u] \to^+_{\mathtt{ex}} t\{x/u\}$.

Proof. By induction on t. The interesting case is $t = s[y/v]$. If $x \in \mathtt{fv}(v)$, then $s[y/v][x/u] \to_{\mathtt{Comp}} s[x/u][y/v[x/u]] \to^*_{\mathtt{ex}\ (i.h.)} s\{x/u\}[y/v\{x/u\}] = t\{x/u\}$. If $x \notin \mathtt{fv}(v)$, then $s[y/v][x/u] =_{\mathtt{C}} s[x/u][y/v] \to^*_{\mathtt{ex}\ (i.h.)} s\{x/u\}[y/v] = t\{x/u\}$.

Lemma 3 (Confluence). *The reduction relation is confluent on open terms.*

Proof. This can be proved by the Tait and Martin Löf technique. The proof proceeds similarly to that of the $\lambda\mathtt{es}$-calculus given in [11].

3 Perpetuality

A *perpetual* strategy gives an *infinite* reduction sequence for a term, if one exists, otherwise, it gives a finite reduction sequence leading to some normal form. Perpetual strategies can be seen as antonyms of normalising strategies, they are

particularly used to obtain normalisation results. For a survey about perpetual strategies we refer the reader to [30].

In contrast to *one-step* strategies for ES given for example in [18, 3], we now define a *many-step* strategy for $\mathtt{x}$-terms which preserves $\lambda\mathtt{ex}$-normal forms and gives a $\to^+_{\lambda\mathtt{ex}}$-reduct for any $t \notin \mathcal{NF}_{\lambda\mathtt{ex}}$.

This is done according to the following cases. If $t = xt_1 \dots t_n$, rewrite the *left-most* t_i which is reducible. If $t = \lambda x.u$, rewrite u. If $t = (\lambda x.s)u\overline{v_n}$, rewrite the head redex. If $t = s[x/u]\overline{v_n}$ and $u \notin \mathcal{SN}_{\lambda\mathtt{ex}}$, rewrite u. If $t = s[x/u]\overline{v_n}$ and $u \in \mathcal{SN}_{\lambda\mathtt{ex}}$, rewrite the head redex using full composition. Formally,

Definition 1. *The* strategy $\rightsquigarrow$ *on* $\mathtt{x}$*-terms is given by an inductive definition.*

$$\frac{\overline{u_n} \in \mathcal{NF}_{\lambda\mathtt{ex}} \quad t \rightsquigarrow t'}{x\overline{u_n}t\overline{v_m} \rightsquigarrow x\overline{u_n}t'\overline{v_m}}\ \mathtt{(p\text{-}var)} \qquad \frac{t \rightsquigarrow t'}{\lambda x.t \rightsquigarrow \lambda x.t'}\ \mathtt{(p\text{-}abs)} \qquad \frac{}{(\lambda x.t)u\overline{u_n} \rightsquigarrow t[x/u]\overline{u_n}}\ \mathtt{(p\text{-}B)}$$

$$\frac{u \in \mathcal{SN}_{\lambda\mathtt{ex}}}{t[x/u]\overline{u_n} \rightsquigarrow t\{x/u\}\overline{u_n}}\ \mathtt{(p\text{-}subs1)} \qquad \frac{u \notin \mathcal{SN}_{\lambda\mathtt{ex}} \quad u \rightsquigarrow u'}{t[x/u]\overline{u_n} \rightsquigarrow t[x/u']\overline{u_n}}\ \mathtt{(p\text{-}subs2)}$$

The strategy is deterministic so that $t \rightsquigarrow u$ and $t \rightsquigarrow v$ implies $u = v$. Moreover, the strategy is not necessarily leftmost-outermost or left-to-right because of the `(p-subs1)` rule: substitution propagation can be performed in any order. Note also that the strategy is not effective since it is based on an undecidable predicate. The strategy is perpetual: if $t \notin \mathcal{SN}_{\lambda\mathtt{ex}}$ and $t \rightsquigarrow t'$, then $t' \notin \mathcal{SN}_{\lambda\mathtt{ex}}$. This will be used later to give an inductive characterisation of the set $\mathcal{SN}_{\lambda\mathtt{ex}}$.

Lemma 4. *If* $t \rightsquigarrow t'$*, then* $t \to^+_{\lambda\mathtt{ex}} t'$*.*

Proof. By induction on the strategy $\rightsquigarrow$ using Lemma 2.

Theorem 1 (Perpetuality). *If* $t \rightsquigarrow t'$ *and* $t' \in \mathcal{SN}_{\lambda\mathtt{ex}}$*, then* $t \in \mathcal{SN}_{\lambda\mathtt{ex}}$*.*

Proof. By induction on the strategy $\rightsquigarrow$. We only treat the non trivial cases.

`(p-B)` $t = (\lambda x.s)u\overline{u_n} \rightsquigarrow s[x/u]\overline{u_n} = t'$. If $s[x/u]\overline{u_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$, then $s, u, \overline{u_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$. We thus show by induction on $\eta_{\lambda\mathtt{ex}}(s) + \eta_{\lambda\mathtt{ex}}(u) + \Sigma_i \eta_{\lambda\mathtt{ex}}(u_i)$ that every $\lambda\mathtt{ex}$-reduct of $(\lambda x.s)u\overline{u_n}$ is in $\mathcal{SN}_{\lambda\mathtt{ex}}$. Conclude $(\lambda x.s)u\overline{u_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$.

`(p-subs2)` $t = s[x/u]\overline{u_n} \rightsquigarrow s[x/u']\overline{u_n} = t'$, $u \notin \mathcal{SN}_{\lambda\mathtt{ex}}$ and $u \rightsquigarrow u'$. If $s[x/u']\overline{u_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$ then in particular $u' \in \mathcal{SN}_{\lambda\mathtt{ex}}$, thus $u \in \mathcal{SN}_{\lambda\mathtt{ex}}$ by the i.h. From $u \notin \mathcal{SN}_{\mathtt{ex}}$ and $u \in \mathcal{SN}_{\lambda\mathtt{ex}}$ we get (constructively) any proposition, so in particular $t \in \mathcal{SN}_{\lambda\mathtt{ex}}$.

`(p-subs1)` $t = s[x/u]\overline{u_n} \rightsquigarrow s\{x/u\}\overline{u_n} = t'$ and $u \in \mathcal{SN}_{\lambda\mathtt{ex}}$. Then the **(IE)** property (Lemma 8) allows to conclude.

4 Normalisation Properties

To show that untyped x-terms enjoy PSN and typed x-terms are $\lambda\mathtt{ex}$-strongly normalising we proceed in two different steps. We first define an inductive set $\mathcal{ISN}$ which turns out to be equal to $\mathcal{SN}_{\lambda\mathtt{ex}}$. PSN can then be easily proved by using the inductive definition of $\mathcal{ISN}$. To show SN, we can then choose at least two different ways to proceed. We include in Section 4.1 the shortest one which is based on simple arithmetical arguments [27], and we refer the reader to [12] for the second one which uses standard reducibility technology [26, 8].

Inductive characterisations of SN terms are useful, for instance, in constructive SN proofs. An inductive definition of SN terms for the λ-calculus is given for example in [28]. It was then extended in [3, 18] for calculi with ES, but using many different inference rules to characterise SN terms of the form $t[x/u]$. We just give here one inference rule for each possible x-term.

Definition 2. *The* inductive set $\mathcal{ISN}$ *is defined as follows:*

$$\frac{t_1,\dots,t_n \in \mathcal{ISN} \quad n \geq 0}{xt_1\dots t_n \in \mathcal{ISN}}\ (\mathtt{var}) \qquad \frac{u[x/v]t_1\dots t_n \in \mathcal{ISN} \quad n \geq 0}{(\lambda x.u)vt_1\dots t_n \in \mathcal{ISN}}\ (\mathtt{app})$$

$$\frac{u\{x/v\}t_1\dots t_n \in \mathcal{ISN} \quad v \in \mathcal{ISN} \quad n \geq 0}{u[x/v]t_1\dots t_n \in \mathcal{ISN}}\ (\mathtt{subs}) \qquad \frac{u \in \mathcal{ISN}}{\lambda x.u \in \mathcal{ISN}}\ (\mathtt{abs})$$

Proposition 1. $\mathcal{SN}_{\lambda\mathtt{ex}} = \mathcal{ISN}$.

Proof. If $t \in \mathcal{SN}_{\lambda\mathtt{ex}}, t \in \mathcal{ISN}$ is proved by induction on the pair $\langle \eta_{\lambda\mathtt{ex}}(t), \mathtt{size}(t)\rangle$. If $t \in \mathcal{ISN}$, $t \in \mathcal{SN}_{\lambda\mathtt{ex}}$ is proved by induction on $t \in \mathcal{ISN}$ using Theorem 1.

Theorem 2 (PSN for λ-terms). *If $t \in \mathcal{SN}_\beta$, then $t \in \mathcal{SN}_{\lambda\mathtt{ex}}$.*

Proof. By induction on the definition of $\mathcal{SN}_\beta$ [28] using Prop. 1. If $t = x\overline{t_n}$ with $t_i \in \mathcal{SN}_\beta$, then $t_i \in \mathcal{SN}_{\lambda\mathtt{ex}}$ by the i.h. so that the (var) rule allows to conclude. The case $t = \lambda x.u$ is similar. If $t = (\lambda x.u)v\overline{t_n}$, with $u\{x/v\}\overline{t_n} \in \mathcal{SN}_\beta$ and $v \in \mathcal{SN}_\beta$, then both terms are in $\mathcal{SN}_{\lambda\mathtt{ex}}$ by the i.h. so that the (subs) rule gives $u[x/v]\overline{t_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$ and the (app) rule gives $(\lambda x.u)v\overline{t_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$.

We now give a type system for x-terms. Richer type systems with intersection types could also be given to characterise the set $\mathcal{SN}_{\lambda\mathtt{ex}}$ in terms of typed terms (see [13, 18] for details).

Types are built over a set of atomic types and the $\to$ constructor. An *environment* is a finite set of pairs $x : A$. A *sequent* $\Gamma \vdash t : A$ is formed by an environment Γ, a term t and a type A. *Derivations* of sequents are obtained by application of the following typing rules.

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash tu : B}$$

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B} \qquad \frac{\Gamma \vdash u : B \quad \Gamma, x : B \vdash t : A}{\Gamma \vdash t[x/u] : A}$$

A term t *of type* A, written t^A, is a term s.t. $\Gamma \vdash t : A$ is derivable for some Γ. A *typed* term t is a term of type A for some type A.

Induction on type derivations together with weakening/strengthening allow us to show the following stability properties.

Lemma 5 (Stability of Typed Terms).

(by substitution) *If* $\Gamma \vdash u : B$ & $\Gamma, x : B \vdash t : A$, *then* $\Gamma \vdash t\{x/u\} : A$.
(by reduction) *If* $\Gamma \vdash t : A$ & $t \to_{\lambda\mathtt{ex}} t'$, *then* $\Gamma \vdash t' : A$.

4.1 The arithmetical technique

This technique is based on van Daalen's strong normalisation proof for the typed lambda-calculus [27], and is extremely short.

Lemma 6. *If* $t^A, u^B \in \mathcal{SN}_{\lambda\mathtt{ex}}$, *then* $t\{x^B/u^B\} \in \mathcal{SN}_{\lambda\mathtt{ex}}$.

Proof. By induction on $\langle B, \eta_{\lambda\mathtt{ex}}(t), \mathtt{size}(t)\rangle$.

- The cases $t = x$, $t = \lambda y.v$ and $y\overline{v_n}$ are straightforward.
- $t = xv\overline{v_n}$. The i.h. gives $V = v\{x/u\}$ and $V_i = v_i\{x/u\}$ in $\mathcal{SN}_{\lambda\mathtt{ex}}$. To show $t\{x/u\} = uV\overline{V_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$ we show that all its reducts are in $\mathcal{SN}_{\lambda\mathtt{ex}}$. We reason by induction on $\eta_{\lambda\mathtt{ex}}(u) + \eta_{\lambda\mathtt{ex}}(V) + \Sigma_{i\in 1\ldots n}\ \eta_{\lambda\mathtt{ex}}(V_i)$.
  If reduction takes place in a subterm of $uV\overline{V_n}$, we conclude by the i.h.
  Suppose $u = \lambda y.U$ and $(\lambda y.U)V\overline{V_n} \to U[y/V]\overline{V_n}$. Then $\mathtt{type}(V) = \mathtt{type}(v) < \mathtt{type}(u) = \mathtt{type}(x)$ so that $U\{y/V\} \in \mathcal{SN}_{\lambda\mathtt{ex}}$ by the i.h. Write $U\{y/V\}\overline{V_n} = (z\overline{V_n})\{z/U\{y/V\}\}$. We have $\mathtt{type}(U\{y/V\}) = \mathtt{type}(U) < \mathtt{type}(u)$ so that $U\{y/V\}\overline{V_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$ by the i.h. We conclude $U[y/V]\overline{V_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$ by Prop. 1.
- $t = (\lambda y.s)v\overline{v_n}$. The i.h. gives $S = s\{x/u\}$, $V = v\{x/u\}$ and $V_i = v_i\{x/u\}$ in $\mathcal{SN}_{\lambda\mathtt{ex}}$. To show $t\{x/u\} = (\lambda y.S)V\overline{V_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$ we show that all its reducts are in $\mathcal{SN}_{\lambda\mathtt{ex}}$. We reason by induction on $\eta_{\lambda\mathtt{ex}}(S)+\eta_{\lambda\mathtt{ex}}(V)+\Sigma_{i\in 1\ldots n}\ \eta_{\lambda\mathtt{ex}}(V_i)$.
  If reduction takes place in a subterm of $(\lambda y.S)V\overline{V_n}$, we conclude by the i.h.
  Otherwise suppose $(\lambda y.S)V\overline{V_n} \to S[y/V]\overline{V_n}$. Now, take $T = s\{y/v\}\ \overline{v_n}$. Since $\eta_{\lambda\mathtt{ex}}(T) < \eta_{\lambda\mathtt{ex}}(t)$, then the i.h. gives $T\{x/u\} \in \mathcal{SN}_{\lambda\mathtt{ex}}$. We write $S\{y/V\}\overline{V_n} = T\{x/u\}$ so that Prop. 1 gives $S[y/V]\overline{V_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$. Thus all the reducts of $t\{x/u\}$ are $\mathcal{SN}_{\lambda\mathtt{ex}}$ and we can conclude $t\{x/u\} \in \mathcal{SN}_{\lambda\mathtt{ex}}$.
- $t = s[y/v]\overline{v_n}$. The proof proceeds as in the previous case.

Theorem 3 (SN for $\lambda\mathtt{ex}$). *If* t *is a typed term, then* $t \in \mathcal{SN}_{\lambda\mathtt{ex}}$.

Proof. By induction on t. The cases $t = x$ and $t = \lambda x.u$ are straightforward. If $t = uv$, then u, v are typed and by the i.h. $u, v \in \mathcal{SN}_{\lambda\mathtt{ex}}$. We write $t = (z\ v)\{z/u\}$, where $z\ v$ is $\mathcal{SN}_{\lambda\mathtt{ex}}$ by Definition 2 and appropriately typed. Lemma 6 then gives $t \in \mathcal{SN}_{\lambda\mathtt{ex}}$. If $t = u[x/v]$, then u, v are typed and by the i.h. $u, v \in \mathcal{SN}_{\lambda\mathtt{ex}}$ so that Lemma 6 gives $u\{x/v\} \in \mathcal{SN}_{\lambda\mathtt{ex}}$. Prop. 1 allows us to conclude $u[x/v] \in \mathcal{SN}_{\lambda\mathtt{ex}}$.

5 The (IE) Property

The aim of this section is to show the key argument used to guarantee that our strategy (Definition 1) is perpetual. More precisely, we show that normalisation of **I***mplicit* substitution implies normalisation of **E***xplicit* substitution:

$$(\textbf{IE})\ u \in \mathcal{SN}_{\lambda\mathtt{ex}}\ \&\ t\{x/u\} \in \mathcal{SN}_{\lambda\mathtt{ex}}\ \text{imply}\ t[x/u] \in \mathcal{SN}_{\lambda\mathtt{ex}}$$

To show the **(IE)** property we adapt the labelling technique [4, 2] to the equational case. Given a set of variables $\mathbb{S}$, the $\mathbb{S}$-*labelled terms* (or simply *labelled terms* if $\mathbb{S}$ is clear from the context), are given by:

$$\mathcal{T}_{\mathbb{S}} ::= x \mid \mathcal{T}_{\mathbb{S}}\ \mathcal{T}_{\mathbb{S}} \mid \lambda x.\mathcal{T}_{\mathbb{S}} \mid \mathcal{T}_{\mathbb{S}}[x/\mathcal{T}_{\mathbb{S}}] \mid \mathcal{T}_{\mathbb{S}}[\![x/v]\!]\ (v \in \mathcal{SN}_{\lambda\mathtt{ex}}\ \&\ \mathtt{fv}(v) \subseteq \mathbb{S})$$

Thus, labelled substitutions can only contain $\mathtt{x}$-terms so in particular they cannot contain other labelled substitutions inside them.

Note that we can always assume that subterms $u[x/v]$ and $u[\![x/v]\!]$ inside $t \in \mathcal{T}_{\mathbb{S}}$ are s.t. $x \notin \mathbb{S}$. Indeed, α-conversion allows to choose names outside $\mathbb{S}$ for the bound variables of $\mathbb{S}$-terms. The idea behind the operational semantics of $\mathbb{S}$-terms, specified by the following set of equations and reduction rules, is that labelled substitutions may commute/traverse ordinary substitutions but these last ones cannot traverse the labelled ones. This behaviour of labelled substitutions is later used to simulate application of implicit substitution.

Equations :			
$t[y/u][\![x/v]\!]$	$=_{\underline{\mathtt{C}}}$	$t[\![x/v]\!][y/u]$	if $x \notin \mathtt{fv}(u)\ \&\ y \notin \mathtt{fv}(v)$
$t[\![y/u]\!][\![x/v]\!]$	$=_{\underline{\mathtt{C}}}$	$t[\![x/v]\!][\![y/u]\!]$	if $x \notin \mathtt{fv}(u)\ \&\ y \notin \mathtt{fv}(v)$
Rules :			
$x[\![x/v]\!]$	$\to_{\underline{\mathtt{Var}}}$	v	
$t[\![x/v]\!]$	$\to_{\underline{\mathtt{Gc}}}$	t	if $x \notin \mathtt{fv}(t)$
$(tu)[\![x/v]\!]$	$\to_{\underline{\mathtt{App}}}$	$t[\![x/v]\!]\ u[\![x/v]\!]$	
$(\lambda y.t)[\![x/v]\!]$	$\to_{\underline{\mathtt{Lamb}}}$	$\lambda y.t[\![x/v]\!]$	
$t[y/u][\![x/v]\!]$	$\to_{\underline{\mathtt{Comp}}}$	$t[\![x/v]\!][y/u[\![x/v]\!]]$	if $x \in \mathtt{fv}(u)$

The $\underline{\mathtt{x}}$ (resp. $\mathbb{EX}$) reduction relation is generated by the previous rules modulo α (resp. $\alpha \cup \underline{\mathtt{C}}$) conversion. In particular, they enjoy termination.

As expected, reduction on labelled terms can be simulated by reduction on their underlying $\mathtt{x}$-terms.

Definition 3. Unlabelled *of* $\mathbb{S}$*-terms are* $\mathtt{x}$*-terms defined by induction.*

$$\begin{array}{lll} \mathtt{U}(x) := x & \mathtt{U}(\lambda x.t) := \lambda x.\mathtt{U}(t) & \mathtt{U}(t[x/u]) := \mathtt{U}(t)[x/\mathtt{U}(u)] \\ & \mathtt{U}(tu) := \mathtt{U}(t)\mathtt{U}(u) & \mathtt{U}(t[\![x/u]\!]) := \mathtt{U}(t)[x/u] \end{array}$$

Consider the relation $\lambda\underline{\mathtt{ex}} = \lambda\mathtt{ex} \cup \mathbb{EX}$ on labelled terms.

Lemma 7. *Let $t \in \mathcal{T}_{\mathbb{S}}$. If $t \in \mathcal{SN}_{\lambda\underline{\mathtt{ex}}}$, then $\mathtt{U}(t) \in \mathcal{SN}_{\lambda\mathtt{ex}}$.*

Proof. We first prove by induction on $\rightarrow_{\lambda\mathtt{ex}}$ the following: for $t \in \mathcal{T}_{\mathbb{S}}$, if $\mathtt{U}(t) \rightarrow_{\lambda\mathtt{ex}} u'$, then $\exists\ u \in \mathcal{T}_{\mathbb{S}}$ s.t. $t \rightarrow_{\lambda\underline{\mathtt{ex}}} u$ and $\mathtt{U}(u) = u'$. To conclude, we prove that every $\lambda\mathtt{ex}$-reduct of $\mathtt{U}(t)$ is in $\mathcal{SN}_{\lambda\mathtt{ex}}$ by induction on $\eta_{\lambda\underline{\mathtt{ex}}}(t)$ using the first property.

Taking $\mathbb{S} = \mathtt{fv}(u)$ and transforming the $\mathtt{x}$-term $s[x/u]\overline{u_n}$ into the $\lambda\underline{\mathtt{ex}}$-term $s[\![x/u]\!]\overline{u_n}$ we have the following special case.

Corollary 1. *If $s[\![x/u]\!]\overline{u_n} \in \mathcal{SN}_{\lambda\underline{\mathtt{ex}}}$, then $s[x/u]\overline{u_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$.*

We now split $\lambda\underline{\mathtt{ex}}$ in two disjoint relations $\lambda\underline{\mathtt{ex}}^i$ and $\lambda\underline{\mathtt{ex}}^e$ which will be projected into $\lambda\mathtt{ex}$-reduction sequences differently.

Definition 4. *The* internal *reduction relation $\lambda\underline{\mathtt{ex}}^i$ is given by $\mathbb{EX}$-reduction together with $\lambda\mathtt{ex}$-reduction in the bodies of labelled substitutions. The* external *reduction relation $\lambda\underline{\mathtt{ex}}^e$ is given by $\lambda\mathtt{ex}$-reduction everywhere except inside bodies of labelled substitutions.*

We will also use the following function $\mathtt{xc}$ from labelled terms to $\mathtt{x}$-terms.

$\mathtt{xc}(x)$	$:= x$	$\mathtt{xc}(tu)$	$:= \mathtt{xc}(t)\mathtt{xc}(u)$
$\mathtt{xc}(\lambda y.t)$	$:= \lambda y.\mathtt{xc}(t)$	$\mathtt{xc}(t[x/u])$	$:= \mathtt{xc}(t)[x/\mathtt{xc}(u)]$
		$\mathtt{xc}(t[\![x/v]\!])$	$:= \mathtt{xc}(t)\{x/v\}$

Corollary 2. *Let t be a labelled term. If $\mathtt{xc}(t) \in \mathcal{SN}_{\lambda\mathtt{ex}}$, then $t \in \mathcal{SN}_{\lambda\underline{\mathtt{ex}}}$.*

Proof. Prove (using Lemma 1) that $\lambda\underline{\mathtt{ex}}$ can be projected into $\lambda\mathtt{ex}$ as follows:

1. $t \rightarrow_{\lambda\underline{\mathtt{ex}}^i} t'$ implies $\mathtt{xc}(t) \rightarrow^*_{\lambda\mathtt{ex}} \mathtt{xc}(t')$.
2. $t \rightarrow_{\lambda\underline{\mathtt{ex}}^e} t'$ implies $\mathtt{xc}(t) \rightarrow^+_{\lambda\mathtt{ex}} \mathtt{xc}(t')$.

Then show that $\lambda\underline{\mathtt{ex}}^i$ is terminating (see [12] for details). Last, apply the abstract Theorem 4 given at the end of this section by taking $\mathtt{a}_1 = \lambda\underline{\mathtt{ex}}^i$, $\mathtt{a}_2 = \lambda\underline{\mathtt{ex}}^e$, $\mathtt{A} = \lambda\mathtt{ex}$ and $u\ \mathcal{R}\ U$ iff $\mathtt{xc}(u) = U$. We get that $\mathtt{xc}(t) \in \mathcal{SN}_{\lambda\mathtt{ex}}$ implies (constructively) $t \in \mathcal{SN}_{\lambda\underline{\mathtt{ex}}^i \cup \lambda\underline{\mathtt{ex}}^e} = \mathcal{SN}_{\lambda\underline{\mathtt{ex}}}$ so we thus conclude.

The previous corollary allows us to conclude with the main property required in the proof of the Perpetuality Theorem:

Lemma 8 (IE Property). *If $u, s\{x/u\}\overline{u_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$, then $s[x/u]\overline{u_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$.*

Proof. Let $s\{x/u\}\overline{u_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$, define $\mathbb{S} = \mathtt{fv}(u)$ and consider the $\mathbb{S}$-labelled term $s[\![x/u]\!]\overline{u_n}$. Then $\mathtt{xc}(s[\![x/u]\!]\overline{u_n}) = \mathtt{xc}(s)\{x/u\}\overline{\mathtt{xc}(u_n)} = s\{x/u\}\overline{u_n}$ so that $\mathtt{xc}(s[\![x/u]\!]\overline{u_n}) \in \mathcal{SN}_{\lambda\mathtt{ex}}$. We get $s[\![x/u]\!]\overline{u_n} \in \mathcal{SN}_{\lambda\underline{\mathtt{ex}}}$ by Corollary 2 and $s[x/u]\overline{u_n} \in \mathcal{SN}_{\lambda\mathtt{ex}}$ by Corollary 1.

Theorem 4. *Let $\mathtt{a}_1$ and $\mathtt{a}_2$ be two reduction relations on $\mathtt{s}$ and let $\mathtt{A}$ be a reduction relation on $\mathtt{S}$. Let $\mathcal{R} \subseteq \mathtt{s} \times \mathtt{S}$. Suppose $\mathtt{a}_1$ is well-founded and also*

- *For every u, v, U ($u\ \mathcal{R}\ U$ & $u\ \mathtt{a}_1\ v$ imply $\exists V$ s.t. $v\mathcal{R}V$ and $U\ \mathtt{A}^*\ V$).*
- *For every u, v, U ($u\ \mathcal{R}\ U$ & $u\ \mathtt{a}_2\ v$ imply $\exists V$ s.t. $v\ \mathcal{R}\ V$ and $U\ \mathtt{A}^+\ V$).*

Then, $t\ \mathcal{R}\ T$ & $T \in \mathcal{SN}_{\mathtt{A}}$ imply $t \in \mathcal{SN}_{\mathtt{a}_1 \cup \mathtt{a}_2}$.

Proof. A constructive proof of this theorem can be found in Corollary 26 of [17].

6 Deriving SN for other calculi

We now informally derive SN for other calculi with ES (having or not safe composition) from SN of $\lambda\mathtt{ex}$, thus suggesting the existence of self-contained SN proofs also for them. However, while the correspondence/translation between yet another calculi without composition and $\lambda\mathtt{ex}$ seems to be unproblematic, the relation with *all* possible forms of safe composition is not claimed.

• The $\lambda\mathtt{x}$-calculus [20, 23] is a sub-calculus of $\lambda\mathtt{ex}$. The fact that $t \to_{\lambda\mathtt{x}} t'$ implies $t \to^{+}_{\lambda\mathtt{ex}} t'$ is then straightforward. Since typed terms in both calculi are the same, we thus deduce that typed $\mathtt{x}$-terms are $\lambda\mathtt{x}$-strongly normalising.

• The $\lambda\mathtt{es}$-calculus [11] can be seen as a refinement of $\lambda\mathtt{ex}$, where propagation of substitution with respect to application and substitution is done in a controlled way. We refer the reader to [11] for details on the rules. The fact that $t \to_{\lambda\mathtt{es}} t'$ implies $t \to^{+}_{\lambda\mathtt{ex}} t'$ is straightforward. Typed terms in both calculi are the same, we thus deduce that typed $\mathtt{x}$-terms are $\lambda\mathtt{es}$-strongly normalising.

• Milner's calculus with partial substitution [22], called λ_{sub}, is able to encode λ-calculus in terms of a bigraphical reactive system. Syntax of λ_{sub} is given by $\mathtt{x}$-terms and reduction rules *completely* propagate a substitution $[x/u]$ only on one occurrence of x at a time (see for example [22] for details). In [13] it is shown that there exist a translation $\mathtt{T}$ from $\mathtt{x}$-terms to $\mathtt{x}$-terms such that $t \to_{\lambda_{sub}} t'$ implies $\mathtt{T}(t) \to^{+}_{\lambda\mathtt{es}} \mathtt{T}(t')$. Since translation $\mathtt{T}$ preserves typability, we conclude that typed $\mathtt{x}$-terms are λ_{sub}-strongly normalising from the previous point.

• A λ-calculus with *partial* β-steps appears in [5]. Syntax is given by pure λ-terms and semantics is very similar to that of λ_{sub}. Similarly to [13], a translation $\mathtt{T}$ from λ-terms to $\mathtt{x}$-terms can be defined to project one-step reduction in λ_{β_p} into at least one-step reduction in λ_{sub}. Since typed λ-terms translate to typed $\mathtt{x}$-terms, then typed λ-terms are λ_{β_p}-strongly normalising from the previous point.

• David and Guillaume [4] defined a calculus with *labels*, called λ_{ws}, which allows *controlled* composition of ES without losing PSN. The calculus λ_{ws} has a strong form of composition which is safe but not full. Its (typed) named notation can be translated into (typed) $\mathtt{x}$-terms in such a way that SN for typed terms in λ_{ws} is a consequence of SN for typed $\lambda\mathtt{ex}$.

• A calculus with a safe notion of composition in director string notation is defined in [25]. Its named version can be understood as $\lambda\mathtt{x}$ together with a composition rule $t[x/u][y/v] \to t[x/u[y/v]]$ where $y \in \mathtt{fv}(u)$ & $y \notin \mathtt{fv}(t)$. The calculus can be easily simulated in $\lambda\mathtt{ex}$ by rules $\mathtt{Comp}$ and $\mathtt{Gc}$. Thus, again, typed $\mathtt{x}$-terms are strongly normalising.

• The $\lambda\mathtt{esw}$-calculus [11] was used as a technical tool to show PSN for $\lambda\mathtt{es}$. The syntax extends $\mathtt{x}$-terms with weakening constructors. It is then straightforward to define a translation $\mathtt{T}$ from $\lambda\mathtt{esw}$-terms to $\mathtt{x}$-terms which forgets these weakening operators. The reduction relation $\lambda\mathtt{esw}$ can be split into an equational system $\mathcal{E}$ and two rewriting relations $\mathcal{L}_1$ and $\mathcal{L}_2$ s.t. $t =_{\mathcal{E}} t'$ or $t \to_{\mathcal{L}_1} t'$ implies $\mathtt{T}(t) =_{\mathtt{C}} \mathtt{T}(t')$ and $t \to_{\mathcal{L}_2} t'$ implies $\mathtt{T}(t) \to^{+}_{\lambda\mathtt{ex}} \mathtt{T}(t')$.

The reduction relation generated by the rules $\mathcal{L}_1$ modulo the equations $\mathcal{E}$ can be easily shown to be terminating. Therefore, every infinite $\lambda\mathtt{esw}$-reduction sequence must contain infinitely many reduction steps generated by the rules

$\mathcal{L}_2$ modulo the equations $\mathcal{E}$, so that we obtain, via the translation $\mathtt{T}$, an infinite $\lambda\mathtt{ex}$-reduction sequence. Also, typed $\lambda\mathtt{esw}$-terms trivially translate via $\mathtt{T}$ to typed $\mathtt{x}$-terms. A consequence is that typed $\mathtt{xw}$-terms are $\lambda\mathtt{esw}$-strongly normalising.

7 Conclusion

We define a simple perpetual strategy for a calculus with ES enjoying full composition. We use this strategy to provide an inductive definition of SN terms which is then used to prove that untyped terms enjoy PSN and typed terms are SN. The proofs are simple, but especially self-contained, no simulation of the source calculus into another SN calculus is used. The inductive characterisation of SN terms and the SN theorem are extremely simple w.r.t. other proofs in the literature [3, 18] for ES. Last but not least, our development is constructive as we make no use of classical logic reasoning.

Some remarks about the application of this method to other calculi might be interesting. First of all, it is worth noticing that full composition alone is not sufficient to achieve the SN proof, otherwise the $\lambda\sigma$-calculus [1], which is known to *not* being strongly normalising [21], could be treated. Indeed, our strategy $\rightsquigarrow$ for $\lambda\mathtt{ex}$ is not perpetual for $\lambda\sigma$: Melliès' counter-example is based on an infinite $\lambda\sigma$-reduction sequence starting from a typed term which is not reached by our perpetual strategy. In other words, $\rightsquigarrow$ is incomplete for $\lambda\sigma$. The definition of a perpetual strategy for $\lambda\sigma$ remains open. The definition of a one-step perpetual strategy (eventually effective) for $\lambda\mathtt{ex}$ also deserves future attention.

We believe that a de Bruijn version of $\lambda\mathtt{ex}$ could be useful in real implementations. This could be achieved by using for example $\lambda\sigma$ technology (so that equation $\mathtt{C}$ can be eliminated) together with the control of composition needed to guarantee strong normalisation.

Acknowledgements*:* I am grateful to M. Fernández, S. Lengrand, F. Renaud, F. R. Sinot, V. van Oostrom and the anonymous referees for useful comments.

References

1. M. Abadi, L. Cardelli, P. L. Curien, and J.-J. Lévy. Explicit substitutions. *Journal of Functional Programming*, 4(1):375–416, 1991.
2. A. Arbiser, E. Bonelli, and A. Ríos. Perpetuality in a lambda calculus with explicit substitutions and composition. WAIT, JAIIO, 2000.
3. E. Bonelli. Perpetuality in a named lambda calculus with explicit substitutions. *Mathematical Structures in Computer Science*, 11(1):47–90, 2001.
4. R. David and B. Guillaume. A λ-calculus with explicit weakening and explicit substitution. *Mathematical Structures in Computer Science*, 11:169–206, 2001.
5. N. G. de Bruijn. Generalizing Automath by Means of a Lambda-Typed Lambda Calculus. In *Mathematical Logic and Theoretical Computer Science*, Lecture Notes in Pure and Applied Mathematics 106, 1987.
6. R. Di Cosmo, D. Kesner, and E. Polonovski. Proof nets and explicit substitutions. *Mathematical Structures in Computer Science*, 13(3):409–450, 2003.

7. G. Dowek, T. Hardin, and C. Kirchner. Higher-order unification via explicit substitutions. *Information and Computation*, 157:183–235, 2000.
8. J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures dans l'arithmétique d'ordre supérieure*. Thèse de doctorat d'état, Univ. Paris VII, 1972.
9. J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50, 1987.
10. T. Hardin and J.-J. Lévy. A confluent calculus of substitutions. In *France-Japan Artificial Intelligence and Computer Science Symposium*, Izu (Japan), 1989.
11. D. Kesner. The theory of calculi with explicit substitutions revisited. In CSL, LNCS 4646, pages 238–252, 2007.
12. D. Kesner. Perpetuality for full and safe composition (in a constructive setting), 2008. `http://www.pps.jussieu.fr/~kesner/papers/`.
13. D. Kesner and S. Ó Conchúir. Fundamental properties of Milner's non-local explicit substitution calculus. `http://www.pps.jussieu.fr/~kesner/papers/`.
14. D. Kesner and S. Lengrand. Resource operators for lambda-calculus. *Information and Computation*, 205(4):419–473, 2007.
15. Z. Khasidashvili, M. Ogawa, and V. van Oostrom Uniform Normalisation beyond Orthogonality. In RTA, LNCS 2051, pages 122–136, 2001.
16. J.-W. Klop. *Combinatory Reduction Systems*. PhD thesis, Mathematical Centre Tracts 127, CWI, Amsterdam, 1980.
17. S. Lengrand. *Normalisation and Equivalence in Proof Theory and Type Theory*. PhD thesis, University Paris 7 and University of St Andrews, Nov. 2006.
18. S. Lengrand, P. Lescanne, D. Dougherty, M. Dezani-Ciancaglini, and S. van Bakel. Intersection types for explicit substitutions. *Information and Computation*, 189(1):17–42, 2004.
19. J.-J. Lévy and L. Maranget. Explicit substitutions and programming languages. In *FSTTCS*, LNCS 1738, pages 181–200, 1999.
20. R. Lins. A new formula for the execution of categorical combinators. In *CADE*, LNCS 230, pages 89–98, 1986.
21. P.-A. Melliès. Typed λ-calculi with explicit substitutions may not terminate. In *TLCA*, LNCS 902, pages 328–334, 1995.
22. R. Milner. Local bigraphs and confluence: two conjectures. In *EXPRESS*, ENTCS 175, 2006.
23. K. Rose. Explicit cyclic substitutions. In *CTRS*, LNCS 656, 1992.
24. T. Sakurai. Strong normalizability of calculus of explicit substitutions with composition. `http://www.math.s.chiba-u.ac.jp/~sakurai/papers.html`.
25. F.-R. Sinot, M. Fernández, and I. Mackie. Efficient reductions with director strings. In *RTA*, LNCS 2706, pages 46–60, 2003.
26. W. Tait. Intensional interpretation of functionals of finite type I. *Journal of Symbolic Logic*, 32, 1967.
27. D. T. van Daalen. *The language theory of automath*. PhD thesis, Technische Hogeschool Eindhoven, 1977.
28. F. van Raamsdonk. *Confluence and Normalization for Higher-Order Rewriting*. PhD thesis, Amsterdam University, Netherlands, 1996.
29. F.-R. Sinot, and V. van Oostrom. Preserving termination of the λ-calculus or not, 2007. Unpublished note.
30. F. van Raamsdonk, P. Severi, M. H. Sorensen, and H. Xi. Perpetual reductions in λ-calculus. *Information and Computation*, 149(2), 1999.