Direct proofs of strong normalisation in calculi of explicit substitutions

Daniel Dougherty, Pierre Lescanne

Laboratoire de l'Informatique du Parallélisme

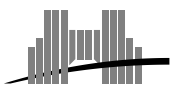
École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON
n° 5668

Direct proofs of strong normalisation in calculi of explicit substitutions

Daniel Dougherty
Pierre Lescanne

7th February 2000

Research Report N° RR2000-05



École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`

Direct proofs of strong normalisation in calculi of explicit substitutions

Daniel Dougherty
Pierre Lescanne

7th February 2000

Abstract

This paper is part of a general programme of treating explicit substitutions as the primary λ-calculi from the point of view of foundations as well as applications. Here we investigate the property of strong normalization.
To date all the proofs of strong normalization of typed calculi of explicit substitutions use a reduction to the strong normalization of classical λ-calculus via the so-called "preservation of strong normalization" property. This paper develops a new approach, namely a direct proof that the strongly normalizing terms are precisely those typable under the intersection-types discipline. We also define an effective perpetual strategy for the general calculus, give an inductive definition of the strongly normalizing terms, and furthermore show that normalization properties are essentially unaffected by the inclusion of a rule for garbage collection. A key role is played by a certain general combinatorial lemma relating the reduction properties of two interacting abstract reductions, which we feel is of interest in its own right.

Keywords: Explicit substitutions, lambda-calculus, strong normalisation, type systems, intersection types termination, perpetual strategy.

Résumé

Cet article fait partie d'un programme plus général visant à traiter les substitutions explicites comme le sont en général les lambda-calculs aussi bien dans la théorie fondamentale que dans les applications. Ici nous analysons, de ce point de vue, la propriété de normalisation forte.
Jusqu'à présent, toutes les preuves de normalisation forte des calculs typés de substitutions explicites s'appuyaient sur une réduction à la normalisation forte du lambda-calcul à travers la propriété dite de "préservation de la normalisation forte". Cet article, quant à lui, développe une approche nouvelle, à savoir une preuve directe que les termes fortement normalisables sont précisément les termes qui sont typables par un système de types avec intersection. Mais dans cet article, nous définissons aussi une stratégie perpétuelle effective pour un calcul général de substitutions explicites, nous donnons une définition inductive des termes fortement normalisables et enfin nous montrons que les propriétés de normalisation ne sont pas essentiellement affectées par l'adjonction d'une règle de glanage de cellules. Dans ces démonstrations, un lemme combinatoire général relie deux réductions abstraites qui interagissent et joue un rôle clé dans la preuve de normalisation forte ; nous pensons que ce lemme a un intérêt par lui-même.

Mots-clés: Substitutions explicites, lambda calcul, normalisation forte, terminaison, systèmes de types, types avec intersection, stratégie perpétuelle.

1 Introduction

The λ-calculus plays a key role in the foundations of logic and of programming language design, and in the implementation of logics and languages as well. The foundation of λ-calculus itself is β-conversion, which relates the primitive notions of abstraction and application in terms of *substitution*. Classical λ-calculus treats substitution as an atomic operation, but in the presence of variable-binding substitution it is a complex operation to define and to implement. So a more careful analysis is required if one is to reason about the correctness of compilers, theorem provers, or proof-checkers. Furthermore the actual cost of performing substitution should be considered when reasoning about complexity of implementations.

Abadi, Cardelli, Curien, and Lévy [1, 2] defined a calculus of *explicit substitutions* to serve as a more faithful model of implementations of the λ-calculus. In this system substitutions are first-class citizens and there is an algebraic/computational structure on the substitutions themselves, derived from the fact that composition is a natural operation on substitutions.

Since then a variety of calculi have been defined. Melliès [16] made the somewhat surprising discovery that the presence of substitution-composition leads to the failure of strong normalization even for simply-typed calculi of explicit substitutions. This suggests that it is useful to analyze the effect of making substitution explicit independently of studying composition of substitutions. Thus composition-free calculi of explicit substitutions have been studied in [15, 7, 5] among others.

It is typically straightforward to prove directly that these calculi inherit the property of confluence from the classical λ-calculus. Normalization properties are more subtle. The key result has been the so-called *preservation of strong normalization* [5, 7]: a pure (substitution-free) term is strongly normalizing under reduction in the explicit substitutions calculus if and only if it is strongly normalizing under β-reduction. This implies, for example, that type systems ensure termination for pure terms.

The original motivation for the Abadi-Cardelli-Curien-Lévy calculus was purely pragmatic, but there is another point of view one may take on such a calculus, namely that making substitution explicit represents a more refined *analysis of substitution* than does classical λ-calculus.

As historical context we note that in their book [12] Curry and Feys insist on the importance of substitution in logic in general and especially in the framework of λ-calculus. They write [page 6] that the synthetic theory of combinators "gives the ultimate analysis of substitutions in terms of a system of extreme simplicity. The theory of lambda-conversion is intermediate in character between synthetic theories and ordinary logic ... and it has the advantage of departing less radically from our intuition."

When one takes this point of view to heart, one can view explicit substitution calculi as a an improvement on both the system of combinators and the classical λ-calculus, a system whose mechanics are first-order and as simple as those of combinatory logic yet which retains the same intensional character as traditional λ-calculus.

In particular we may view explicit substitution calculi as primary and see the classical λ-calculus as a subsystem of these systems, defined by a particular

strategy of "eagerly" evaluating the substitution constructed by contracting a β-redex. In this way the study of explicit substitutions represents a deeper examination of the relationship between abstraction and application.

This setting invites the programme of refining the results of the classical λ-calculus by finding proofs of their explicit-substitutions analogues *in the explicit substitutions system itself.* To the extent that explicit substitution represents a refinement of the basic notions of λ-calculus one can reasonably expect in this way to gain insight into the deeper aspects of λ-calculi in the general sense (even of the original calculus).

As a case study, in this paper we look at strong normalization.

We work in the composition-free calculus $\lambda\mathtt{x}$ of explicit substitutions (which uses names rather than de Bruin indices) and the calculus $\lambda\mathtt{x}_{gc}$ obtained by adding explicit garbage-collection to $\lambda\mathtt{x}$.

Our main results are as follows:

- For the natural generalization of the intersection-types discipline we prove that a term is strongly normalizing in $\lambda\mathtt{x}$ if and only if it is typable. The top-level structure of the proof is a reducibility argument, but the fine structure relies on some new results about reduction, as follows.
- We define an effective perpetual strategy for $\lambda\mathtt{x}$-reduction. It has a somewhat different character from the classical strategy as presented in [3].
- We give a characterization of the strongly normalizing terms by an inductive definition. The proof of perpetuality relyies essentially on the perpetual strategy result.
- As a corollary of the previous, we can see very easily that a term is SN iff it is SN in the calculus extended by garbage-collection. The fact that $\lambda\mathtt{x}_{gc}$ has preservation of strong normalization was first proved in [8]
- We prove a certain general combinatorial lemma relating the reduction properties of two interacting abstract reductions. This plays a central role in our paper, and we feel it is of interest in its own right.

A subtle point of difference between the direct proofs here and preservation of strong normalization results is that the former guarantee termination for all typable terms of $\lambda\mathtt{x}$, not just the pure, substitution-free terms. Our results support the claim that garbage-collection is a very natural addition to the system, even from a purely theoretical point of view: the resulting calculus has more convenient closure-properties than the pure calculus.

Finally, we remark that the characterization of SN terms allows us to adapt Girard's "Candidats de reductibilité" technique [14] to prove strong normalization for the terms typable in a polymorphic-types system as well. This will appear in a future paper.

A full version of the paper with all the proofs can be found on the web at
`http://www.ens-lyon.fr/~plescann/SN_lx.ps`
or at `http://www.wesleyan.edu/~ddougherty/SN_lx.ps`.

Noetherian relations and rank.

A relation R is strongly normalizing, or noetherian, out of x, if there is no infinite sequence $(x_n)_{n\in N}$ with $x_0 = x$ and $x_n R x_{n+1}$.

A relation R is strongly normalizing if it is strongly normalizing out of every x in its domain.

If R is a relation (or a set of rules defining a relation) we write $\mathcal{SN}_R$ for the set of objects that are strongly normalizing for R.

Noetherian relations play a key role in this paper. We recall here the concept of *rank* of an element x, namely an ordinal associated with x that "counts" the length of the longest chain out of x.

The following definition and accompanying lemma are well-known.

Definition 1 *Let R be a binary relation on a set S. Define a partial function $rank_R$ from S to the ordinals by: $rank_R(x) = \sup\{1 + rank_R(y) \mid xRy\}$.*

Lemma 1 *The function $rank_R$ is defined for element x if and only if R is strongly normalizing out of x.*

> **Proof:** Suppose $\mathrm{rank}_R(x)$ is defined. Let y be any element in S such that xRy, clearly $rank_R(y) < \mathrm{rank}_R(x)$. By ordinal induction y is strongly normalizing out of y, hence R is strongly normalizing out of x.
>
> Suppose R is strongly normalizing out of x. R is strongly normalizing on $X = \{y \in S \mid xR^*y\}$ and one can proceed by noetherian induction on R over X. The function rank_R is defined on X and especially on $\bar{X} = \{y \in S \mid xRy\}$, hence $\mathrm{rank}_R(x)$ is defined. □

2 The calculus of explicit substitutions $\lambda \mathtt{x}_{gc}$

Definition 2 *The set of terms with explicit substitutions $\Lambda\mathtt{x}$ is the set of terms M defined as follows:*

$$M, N \quad ::= \quad x \mid \lambda x \cdot M \mid M\,N \mid M\langle x = N\rangle$$

The set of free variables *of a term is defined just as for classical λ-calculus, with an additional clause ensuring that the free variables of $M\langle x = N\rangle$ are the same as the free variables of $(\lambda x.M)N$. In particular, x is bound in $M\langle x = N\rangle$.*

The superterm order $\sqsupset$ is defined as $M \sqsupseteq N$ and $M \neq N$, where $\sqsupseteq$ is defined as follows:

- *$M \sqsupseteq M$,*
- *If $M \sqsupseteq M'$, then $\lambda x \cdot M \sqsupseteq M'$, $M\,N \sqsupseteq M'$ and, $N\,M \sqsupseteq M'$.*

We assume Barendregt's [3] convention, namely that a variable does not occur free and bound in the same subterm. For instance, we assume that x does not occur free in N in the term $M\langle x = N\rangle$. The rules we define further assume this convention and the reader should keep this fact in mind when reading them.

It will be very convenient to have a notation to describe a term M on which is applied a sequence of closures $\langle z_1 = S_1\rangle$, ..., $\langle z_m = S_m\rangle$ then a sequence of applications of terms T_1,..., T_n. Such a term $M\langle z_1 = S_1\rangle ... \langle z_m = S_m\rangle T_1 ... T_n$ will be abbreviated by $M\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$.

Lemma 2 *Every term is of precisely one of the following forms:*

Lmb	$\lambda x.B$
VarHd	$xT_1 \cdots T_n$ *with* $n \geq 0$
βHd	$(\lambda x.B)AT_1 \cdots T_n$ *with* $n \geq 0$
I Clo	$x\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle \boldsymbol{T}$
K Clo	$y\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle \boldsymbol{T}$ *with* $x \not\equiv y$
AbsClo	$(\lambda y.B)\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle \boldsymbol{T}$
AppClo	$(UV)\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle \boldsymbol{T}$

Proof: Clearly those terms are well formed according to Definition 2. On the other hand, each term M has this form. If M is an abstraction, this is covered by Lmb. If M is a variable, this is covered by VarHd with $n = 0$. If M is a closure, then this is covered by one of the last four cases (I Clo, K Clo, AbsClo, AppClo) as we will see below. If M is an application, M is of the form $N\overrightarrow{T}$ where N is either a variable (VarHd) or an abstraction (βHd) or a closure. If N is a closure, N is of the form $P\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle \boldsymbol{T}$ where P is either a variable (I Clo or K Clo) or an abstraction (AbsClo) or an application (AppClo). □

The following concepts, namely $\lambda\mathtt{x}$ and $\lambda\mathtt{x}_{gc}$ are due to R. Bloo and K. Rose [9, 19, 6]. We renamed the rule VarI and VarK, called respectively xv and xvgc by Rose, to recall the distinction between the λ_I and λ_K calculi.

Definition 3 *Let us consider the following rules*

(B)	$(\lambda x B)\, A$	$\to$	$B\langle x = A\rangle$	
(App)	$(MN)\langle x = A\rangle$	$\to$	$M\langle x = A\rangle\, N\langle x = A\rangle$	
(Abs)	$(\lambda y M)\langle x = N\rangle$	$\to$	$\lambda y M\langle x = N\rangle$	
(VarI)	$x\langle x = N\rangle$	$\to$	N	
(VarK)	$y\langle x = N\rangle$	$\to$	y	
(gc)	$M\langle x = A\rangle$	$\to$	M	*if* $x \notin M$

We define

$$\begin{aligned} \mathtt{x} &= \{\mathit{App}, \mathit{Abs}, \mathit{VarI}, \mathit{VarK}\} \\ \lambda\mathtt{x} &= \mathtt{x} \cup \{B\} \\ \mathtt{x}_{gc} &= \{\mathit{App}, \mathit{Abs}, \mathit{VarI}, \mathit{gc}\} \\ \lambda\mathtt{x}_{gc} &= \mathtt{x}_{gc} \cup \{B\} \end{aligned}$$

The rule gc is called *garbage-collection*, as it removes "useless" substitutions.

For simplicity we write $\mathcal{SN}$ for $\mathcal{SN}_{\lambda\mathtt{x}}$, the set of terms strongly normalizing under $\lambda\mathtt{x}$. In fact, we will see below that $\mathcal{SN}_{\lambda\mathtt{x}} = \mathcal{SN}_{\lambda\mathtt{x}_{gc}}$.

3 A commutation lemma

In this section we prove a general commutation lemma which is key to our main result. We present it in the framework of abstract reduction systems, as we feel that this result may have applications in other termination problems.

Definition 4 ((Gentle Commutation)) *Two relations* $\rightarrow$ *and* $\Rightarrow$ gently commute *if for every* M, N *and* P *such that* $M \Rightarrow N$ *and* $M \rightarrow P$, *there exists a* Q *such that* $P \Rightarrow^* Q$ *and* $N \rightarrow^+ Q$.

The importance of gentle commutation is that when the target of the $\Rightarrow$-reduction is $\mathcal{SN}$ for $\rightarrow$-reduction, then the two relations commute.

Lemma 3 ((Gentle Commutation Lemma)) *Suppose* $\rightarrow$ *and* $\Rightarrow$ *gently commute.*

- *For every* M, N *and* P *such that* $M \Rightarrow^* N$, $M \rightarrow^* P$, *and* $N \in \mathcal{SN}_{\rightarrow}$ *there exists a* Q *such that* $P \Rightarrow^* Q$ *and* $N \rightarrow^* Q$. *Furthermore the number of steps in* $N \rightarrow^* Q$ *is no less than the number of steps in* $M \rightarrow^* P$.
- *If* $M \Rightarrow^* N$ *and* $N \in \mathcal{SN}_{\rightarrow}$ *then* $M \in \mathcal{SN}_{\rightarrow}$; *in fact* $rank_{\rightarrow} M \leq rank_{\rightarrow} N$.

  Proof: To prove the first claim: we proceed by lexicographic induction over $\langle \text{rank}_{\rightarrow} N, n \rangle$ where n is the number of steps in the reduction $M \Rightarrow^* N$. If $n = 0$ then we may take Q to be P; if $M \equiv P$ we may take Q to be N.

  Otherwise suppose $M \Rightarrow N_1 \Rightarrow^* N$ and $M \rightarrow P_1 \rightarrow^* P$. By gentle commutation applied to M, N_1, and P_1 there is an R with $P_1 \Rightarrow^* R$ and $N_1 \rightarrow^+ R$. By induction hypothesis applied to N_1, N, and R there is Q_1 such that $R \Rightarrow^* Q_1$ and $N \rightarrow^+ Q_1$. Now note that $\text{rank}_{\rightarrow} Q_1 < \text{rank}_{\rightarrow} N$ so that the induction hypothesis applies to P_1, P, and Q_1, and we obtain Q with $P \Rightarrow^* Q$ and $Q_1 \rightarrow^+ Q$. This Q witnesses the first assertion.

  The second assertion follows from the first by an easy diagram chase.

  □

Here is another argument, due to Frédéric Lang, (personal communication) which yields the second assertion of the previous lemma. It takes the form of a diamond lemma involving the following relation:

$$\mapsto \; = \; (\Rightarrow \cup \leftarrow)^*$$

Lemma 4 ((Gentle Commutation Diamond Lemma)) *Assume* $\rightarrow$ *and* $\Rightarrow$ *gently commute. If* $M \mapsto N$ *and* $M \rightarrow P$, *there exists* Q *such that* $N \rightarrow Q$ *and* $P \mapsto Q$.

Proof: Assume $M \mapsto N$ and $M \rightarrow P$. If $M = N$, we may take Q to be P. Otherwise there are two cases depending on the nature of $M \mapsto N$.

Suppose $M \Rightarrow M' \mapsto N$. By gentle commutation, there exists R such that $P \Rightarrow^* R$ and $M' \rightarrow^+ R$. Let us write $M' \rightarrow^+ R$, as $M' \rightarrow M'' \rightarrow^* R$. Applying the induction hypothesis to the triple (M', M'', N), we obtain a Q such that $M'' \mapsto Q$ and $N \rightarrow Q$. Q is the answer to the $M \mapsto N$ and $M \rightarrow P$ diagram, since $P \Rightarrow^* R \leftarrow^* M'' \mapsto Q$ means $P \mapsto Q$.

Suppose $M \leftarrow M' \mapsto N$. Applying the induction hypothesis to the triple (M', M, N), we obtain a Q such that $M \mapsto Q$ and $N \to Q$. Q is the answer to the $M \mapsto N$ and $M \to P$ diagram, since $P \leftarrow M \mapsto Q$ means $P \mapsto Q$.

□

Corollary 1 *If $\to$ and $\Rightarrow$ gently commute, if $M \mapsto N$ and if $N \in \mathcal{SN}_{\to}$ then $M \in \mathcal{SN}_{\to}$.*

Proof: By Lemma 4, $\to$ and $\mapsto$ strongly commute and clearly if $N \in \mathcal{SN}_{\to}$ then $M \in \mathcal{SN}_{\to}$. □

Since $\Rightarrow^* \subseteq \mapsto$ the second assertion in Lemma 3 is a consequence of Corollary 1.

4 An effective perpetual strategy for λx-reduction

Following Barendregt [3], we say that a term M is *infinite* with respect to a reduction-relation R if there is an infinite R-reduction out of M. A strategy for R-reduction is *perpetual* if whenever M is R-infinite and M reduces to M' via the strategy, then M' is R-infinite.

In this section we construct an effective perpetual strategy for $\lambda\mathbf{x}_{gc}$-reduction. We first define a relation $\rightsquigarrow$ below which is a perpetual non-deterministic strategy for $\lambda\mathbf{x}_{gc}$-reduction and then show that a certain effective restriction of it is perpetual for $\lambda\mathbf{x}$-reduction. We will be also able to conclude that if a term M is $\lambda\mathbf{x}_{gc}$-infinite then it is also $\lambda\mathbf{x}$-infinite.

Definition 5 *The relation $\rightsquigarrow$ is defined inductively as follows.*

$\mathcal{P}$ *Lmb*	$\lambda x.B$	$\rightsquigarrow$	$\lambda x.B'$	*if* $B \rightsquigarrow B'$
$\mathcal{P}$ *VarHd*	$xT_1 \cdots T_n$	$\rightsquigarrow$	$xT_1 \cdots T_i' \cdots T_n$	*if* $T_i \rightsquigarrow T_i'$.
$\mathcal{P}\beta$*Hd*	$(\lambda x.B)AT_1 \cdots T_n$	$\rightsquigarrow$	$B\langle x = A\rangle T_1 \cdots T_n$.	
$\mathcal{P}$ *I Clo*	$x\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$	$\rightsquigarrow$	$A\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$	
$\mathcal{P}$ *K Clo*	$y\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$	$\rightsquigarrow$	$y\langle x = A'\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$	*if* $A \rightsquigarrow A'$.
$\mathcal{P}$ *K Clo*′	$y\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$	$\rightsquigarrow$	$A\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$	*if A is a normal form*
$\mathcal{P}$ *AbsClo*	$(\lambda y.B)\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$	$\rightsquigarrow$	$\lambda y.B\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$	
$\mathcal{P}$ *AppClo*	$(UV)\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$	$\rightsquigarrow$	$(U\langle x = A\rangle V\langle x = A\rangle)\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$	
$\mathcal{P}$ *gc*	$M\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$	$\rightsquigarrow$	$M\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$ *if* $x \notin M$ *and* $A \in \mathcal{SN}$	

It is easy to see that if $M \rightsquigarrow M'$ then $M \xrightarrow[\lambda\mathbf{x}_{gc}]{} M'$ and that if M is $\rightsquigarrow$-irreducible then M is $\lambda\mathbf{x}$-irreducible. The relation $\rightsquigarrow$ fails to be a deterministic strategy due to the $\mathcal{P}$gc rule and the fact that in the rule for terms $xT_1 \cdots T_n$ there is potentially a choice as to which T_i to reduce. The relation $\rightsquigarrow$ is not effective, since application of the $\mathcal{P}$gc rule requires testing whether a certain sub-term is strongly normalizing.

But we will see below (Corollary 4) that there is a natural subrelation of $\rightsquigarrow$ which is an effective deterministic perpetual strategy.

Now, with the exception of rule $\mathcal{P}$gc and $\mathcal{P}$AppClo it is easy to see that each of the reductions comprising $\rightsquigarrow$ preserves non-$\mathcal{SN}$. We address the two latter rules in turn.

4.1 The rule $\mathcal{P}$gc

We first show that the rule $\mathcal{P}$gc preserves the existence of infinite reductions.

Definition 6 *A n-multi-context is a term with n holes in which we can insert n terms. If n is understood, we say a multi-context.*

If $C[\![..., ..., ...]\!]$ is a multi-context and $M_1 ... M_n$ are terms, then the insertions of those terms in $C[\![..., ..., ...]\!]$ is $C[\![M_1, ..., M_n]\!]$.

Lemma 5 *For all multi-context $C[\![...]\!]$, for terms $A_1 ... A_n$ and terms $M_1 ... M_n$, if $x \notin C[\![M_1, ..., M_n]\!]$ and if for $1 \leq i \leq n$, $A_i \in \mathcal{SN}_{gc}$ and $C[\![M_1, ..., M_n]\!] \in \mathcal{SN}_{gc}$ then $C[\![M_1\langle x = A_1\rangle, ..., M_n\langle x = A_n\rangle]\!] \in \mathcal{SN}_{gc}$.*

Proof: The proof is by induction on triples $(D, \mathcal{M}, \mathcal{N})$ where D is a term, $\mathcal{M}$ and $\mathcal{N}$ are multisets of terms. $(D, \mathcal{M}, \mathcal{N}) \gg (D', \mathcal{M}', \mathcal{N}')$ if and only if $D \xrightarrow[\lambda \mathtt{x}_{gc}]{} D'$ or $D = D'$ and $\mathcal{M} \sqsupset\sqsupset \mathcal{M}'$ or $D = D'$, $\mathcal{M} = \mathcal{M}'$ and, $\mathcal{N} \xrightarrow[\lambda \mathtt{x}_{gc}]{} \xrightarrow[\lambda \mathtt{x}_{gc}]{} \mathcal{N}'$. $\sqsupset\sqsupset$ is the multiset extension [13]. of the superterm order $\sqsupset$ and $\xrightarrow[\lambda \mathtt{x}_{gc}]{} \xrightarrow[\lambda \mathtt{x}_{gc}]{}$ is the multiset extension of the reduction relation. In what follows, D will be $C[\![M_1, ..., M_n]\!]$ and $\xrightarrow[\lambda \mathtt{x}_{gc}]{}$ will be well-founded; $\mathcal{M}$ will be $\{M_1, ..., M_n\}$; $\mathcal{N}$ will be $\{A_1, ..., A_n\}$ and $\xrightarrow[\lambda \mathtt{x}_{gc}]{} \xrightarrow[\lambda \mathtt{x}_{gc}]{}$ will be well-founded.

Assume that for $1 \leq i \leq n$, $A_i \in \mathcal{SN}_{\mathtt{gc}}$, $C[\![M_1, ..., M_n]\!] \in \mathcal{SN}_{\mathtt{gc}}$ and, by induction over $\gg$ that the statement of the lemma is true. Let us prove that $C[\![M_1\langle x = A_1\rangle, ..., M_n\langle x = A_n\rangle]\!]$ reduces only to terms that are in $\mathcal{SN}_{\mathtt{gc}}$.

1. $C[\![M_1\langle x = A_1\rangle, ..., M_n\langle x = A_n\rangle]\!] \xrightarrow[\lambda \mathtt{x}_{gc}]{} C'[\![M_{i_1}\langle x = A_{i_1}\rangle, ..., M_{i_p}\langle x = A_{i_p}\rangle]\!]$ (where the $i_j \in [1..n]$), then
$$C[\![M_1, ..., M_n]\!] \xrightarrow[\lambda \mathtt{x}_{gc}]{} C'[\![M_{i_1}, ..., M_{i_p}]\!],$$
and by induction $C'[\![M_{i_1}\langle x = A_{i_1}\rangle, ..., M_{i_p}\langle x = A_{i_p}\rangle]\!] \in \mathcal{SN}_{\mathtt{gc}}$.
2. $M_i \xrightarrow[\lambda \mathtt{x}_{gc}]{} M'_i$, works also by induction.
3. $A_j \xrightarrow[\lambda \mathtt{x}_{gc}]{} A'_j$, works also by induction.
4. $M_i = M_i^1 M_i^2$ and $M_i\langle x = A_i\rangle \xrightarrow[\lambda \mathtt{x}_{gc}]{} M_i^1\langle x = A_i\rangle M_i^2\langle x = A_i\rangle$.
$$\{M_1, ..., M_i, ..., M_n\} \sqsupset\sqsupset \{M_1, ..., M_i^1, M_i^2, ..., M_n\},$$
hence $C[\![M_1\langle x = A_1\rangle, ..., M_i^1\langle x = A_i\rangle M_i^2\langle x = A_i\rangle, ..., M_n\langle x = A_n\rangle]\!] \in \mathcal{SN}_{\mathtt{gc}}$ by induction.
5. $M_i = \lambda y M'_i$ and $M_i\langle x = A_i\rangle \xrightarrow[\lambda \mathtt{x}_{gc}]{} \lambda y(M'_i\langle x = A_i\rangle)$.
$$\{M_1, ..., M_i, ..., M_n\} \sqsupset\sqsupset \{M_1, ..., M'_i, ..., M_n\},$$
hence $C[\![M_1\langle x = A_1\rangle, ..., \lambda y(M'_i\langle x = A_i\rangle), ..., M_n\langle x = A_n\rangle]\!] \in \mathcal{SN}_{\mathtt{gc}}$ by induction.

6. $M_i\langle x = A_i\rangle \xrightarrow[\lambda \mathtt{x}_{gc}]{} M_i$, which is always applicable since $x \notin M_i$.

$$\{M_1, ..., M_i, ..., M_n\} \sqsupset\sqsupset \{M_1, ..., \ , ..., M_n\},$$

hence $C[\![M_1\langle x = A_1\rangle, ..., y, ..., M_n\langle x = A_n\rangle]\!] \in \mathcal{SN}_{\mathtt{gc}}$ by induction.

□

Corollary 2 *If P is $\lambda \mathtt{x}_{gc}$-infinite and $P \rightsquigarrow P'$ via rule $\mathcal{P}gc$ then P' is $\lambda \mathtt{x}_{gc}$-infinite.*

Proof: $P \rightsquigarrow P'$ via rule $\mathcal{P}\mathsf{gc}$ means $P \equiv M\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle \boldsymbol{T}$ and $P' \equiv M\langle \boldsymbol{z} = \boldsymbol{S}\rangle \boldsymbol{T}$ with $x \notin M$ and $A \in \mathcal{SN}_{\mathtt{gc}}$. If we call $C[\![\ ]\!]$ the 1-multicontext $[\![\ ]\!]\langle \boldsymbol{z} = \boldsymbol{S}\rangle \boldsymbol{T}$, then $P' \equiv C[\![M]\!]$ and $P' \equiv C[\![M\langle x = A\rangle]\!]$. By Lemma 5, if $P' \in \mathcal{SN}_{\mathtt{gc}}$ then $P \in \mathcal{SN}_{\mathtt{gc}}$. □

4.2 The rule $\mathcal{P}$AppClo

Next we argue that application of $\mathcal{P}\mathsf{AppClo}$ preserves the property of being non-$\mathcal{SN}_{\mathtt{gc}}$, that is, if a term $(UV)\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle \boldsymbol{T}$ is $\lambda \mathtt{x}_{gc}$-infinite, then $(U\langle x = A\rangle V\langle x = A\rangle)\langle \boldsymbol{z} = \boldsymbol{S}\rangle \boldsymbol{T}$ is $\lambda \mathtt{x}_{gc}$-infinite as well. It is just here that we make essential use of the Gentle Commutation lemma. It would suffice to show that ordinary $\lambda \mathtt{x}_{gc}$-reduction and the App rule gently commute, but this is false: we must pass to slightly richer sets of rules with better closure properties.

The reader may wonder why ω contains those three rules and why gentle commutation fits well with it. Actually this was not a straightforward result and we are rather proud that this can be explained with so few words and concepts. For us, this is really the heart of the paper and its main contribution.

Definition 7 *The Composition rule is:*

$$M\langle x = P\rangle\langle y = Q\rangle \ \rightarrow \ M\langle y = Q\rangle\langle x = P\langle y = Q\rangle\rangle$$

Let us use ω to refer to the set of rules $\{\mathsf{App}, \mathsf{Abs}, \mathsf{Comp}\}$.

Lemma 6 $\xrightarrow[\lambda \mathtt{x}_{gc}]{}$ *and* $\xrightarrow[\omega]{}$ *gently commute.*

Proof: Given a term M such that $M \xrightarrow[\lambda \mathtt{x}_{gc}]{} P$ and $M \xrightarrow[\omega]{} N$. If the rules used to reduce M do not overlap, there is no problem. Actually one can only consider the case where M is an instance of a lefthand side of a rule in $\lambda \mathtt{x}_{gc}$ or in ω and the other rule overlaps with this lefthand side.

B in $\lambda \mathtt{x}_{gc}$ overlaps with App in ω.

$$((\lambda x B)A)\langle y = C\rangle \xrightarrow[\lambda \mathtt{x}_{gc}]{} B\langle x = A\rangle\langle y = C\rangle$$

and

$$(\lambda x B)A\langle y = C\rangle \xrightarrow[\omega]{} (\lambda x B)\langle y = C\rangle \ A\langle y = C\rangle.$$

Since

$$B\langle x = A\rangle\langle y = C\rangle \xrightarrow[\mathsf{Comp}]{} B\langle y = C\rangle\langle x = A\langle y = C\rangle\rangle$$

and

$$(\lambda x B)\langle y = C\rangle\ A\langle y = C\rangle \xrightarrow[\mathsf{Abs}]{} (\lambda x B\langle y = C\rangle)A\langle y = C\rangle \xrightarrow[B]{} B\langle y = C\rangle\langle x = A\langle y = C\rangle\rangle,$$

$\xrightarrow[\lambda \mathbf{x}_{gc}]{}$ and $\xrightarrow[\omega]{}$ gently commute over $(\lambda x B)A\langle y = C\rangle$.

Now we see that App, Abs, Varl and, gc overlap with Comp. Let us check the overlap $(MN)\langle x = A\rangle\langle y = B\rangle$ of App over Comp. The other rules can be treated routinely according to the same scheme and are left to the reader.

$$(MN)\langle x = A\rangle\langle y = B\rangle \xrightarrow[\lambda \mathbf{x}_{gc}]{} (M\langle x = A\rangle N\langle x = A\rangle)\langle y = B\rangle$$

and

$$(MN)\langle x = A\rangle\langle y = B\rangle \xrightarrow[\omega]{} (MN)\langle y = B\rangle\langle x = A\langle y = B\rangle\rangle.$$

We close the diagram of gentle commutation, by

$(M\langle x = A\rangle N\langle x = A\rangle)\langle y = B\rangle$

$\xrightarrow[\mathsf{App}]{} (M\langle x = A\rangle\langle y = B\rangle)\ N\langle x = A\rangle\langle y = B\rangle$

$\xrightarrow[\mathsf{Comp}]{} \xrightarrow[\mathsf{Comp}]{} (M\langle y = B\rangle\langle x = A\langle y = B\rangle\rangle)\ N\langle y = B\rangle\langle x = A\langle y = B\rangle\rangle$

and by

$(MN)\langle y = B\rangle\langle x = A\langle y = B\rangle\rangle$

$\xrightarrow[\mathsf{App}]{} \xrightarrow[\mathsf{App}]{} (M\langle y = B\rangle\langle x = A\langle y = B\rangle\rangle)\ N\langle y = B\rangle\langle x = A\langle y = B\rangle\rangle$. □

Corollary 3 *If M is $\lambda \mathbf{x}_{gc}$-infinite and $M \rightsquigarrow M'$ via rule $\mathcal{P}$AppClo then M' is $\lambda \mathbf{x}_{gc}$-infinite.*

Proof:

Let $M \equiv (UV)\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$ be $\lambda \mathbf{x}_{gc}$-infinite, then M' is $(U\langle x = A\rangle V\langle x = A\rangle)\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$.

If U or V or A or one of the S_i's or one of the T_j's is $\lambda \mathbf{x}_{gc}$-infinite, so is M'.

Otherwise there are U', V', A', $\boldsymbol{S}'$ and $\boldsymbol{T}'$ such that $U \xrightarrow[\lambda \mathbf{x}_{gc}]{}\!\!\!\!\!\rightarrow\ U'$, $V \xrightarrow[\lambda \mathbf{x}_{gc}]{}\!\!\!\!\!\rightarrow\ V'$, $A \xrightarrow[\lambda \mathbf{x}_{gc}]{}\!\!\!\!\!\rightarrow\ A'$, $\boldsymbol{S} \xrightarrow[\lambda \mathbf{x}_{gc}]{}\!\!\!\!\!\rightarrow\ \boldsymbol{S}'$ and $\boldsymbol{T} \xrightarrow[\lambda \mathbf{x}_{gc}]{}\!\!\!\!\!\rightarrow\ \boldsymbol{T}'$ with

- either $(U'\langle x = A'\rangle V'\langle x = A'\rangle)\langle \boldsymbol{z} = \boldsymbol{S}'\rangle\, \boldsymbol{T}'$ $\lambda \mathbf{x}_{gc}$-infinite,
- or $U' \equiv \lambda y \cdot U''$ and $P \equiv U''\langle y = V'\rangle\langle x = A'\rangle\langle \boldsymbol{z} = \boldsymbol{S}'\rangle\, \boldsymbol{T}'$ $\lambda \mathbf{x}_{gc}$-infinite. But $P \xrightarrow[\mathsf{Comp}]{} P' \equiv U''\langle x = A'\rangle\langle y = V'\langle x = A'\rangle\rangle\langle \boldsymbol{z} = \boldsymbol{S}'\rangle\, \boldsymbol{T}'$. Lemma 6 and the gentle commutation lemma tell us that P' is also $\lambda \mathbf{x}_{gc}$-infinite. Then $((\lambda y \cdot U'')\langle x = A'\rangle V'\langle x = A'\rangle)\langle \boldsymbol{z} = \boldsymbol{S}'\rangle\, \boldsymbol{T}'$ is $\lambda \mathbf{x}_{gc}$-infinite.

In each case M' is $\lambda \mathbf{x}_{gc}$-infinite. □

A remark on the Substitution Lemma

The Substitution Lemma of the classical λ-calculus [3] states a fundamental property of (implicit) substitutions, namely that, when x is not free in L:

$$M[x := N][y := L] \;\equiv\; M[y := L][x := N[y := L]]$$

Observe that the two terms are syntactically identical above. When generalized to an explicit substitutions calculus the analogous statement is weakened:

$$M\langle x = N\rangle\langle y = L\rangle \;=_{\mathtt{x}}\; M\langle y = L\rangle\langle x = N\langle y = L\rangle\rangle$$

It is not hard to see that the two terms above can have quite different reduction-behavior; in particular one may readily construct an example (for instance $M \equiv z$, $N \equiv yy$ and $L \equiv \lambda u \cdot uu$) in which the left-hand side is $\mathcal{SN}$ under $\lambda\mathtt{x}$, while the right-hand side is infinite. But as a consequence of Lemma 6 one can see that if the right-hand side is $\mathcal{SN}$ then so is the left-hand side: just observe that the left-hand side reduces to the right-hand side by the Comp rule, which is an instance of $\xrightarrow[\omega]{}$.

4.3 Perpetual strategies

Theorem 1 *The relation $\rightsquigarrow$ is $\xrightarrow[\lambda\mathtt{x}_{gc}]{}$ -perpetual. That is, if M is $\lambda\mathtt{x}_{gc}$-infinite and $M \rightsquigarrow M'$ then M' is $\lambda\mathtt{x}_{gc}$-infinite.*

> **Proof:** An examination of cases on the definition of $\rightsquigarrow$. The hard cases, namely $\mathcal{P}$gc and $\mathcal{P}$AppClo, were treated in Corollaries 2 and 3. □

Recall that $\rightsquigarrow$ is non-deterministic. We stress that Theorem 1 says that *any* $\rightsquigarrow$-reduction out of a non-$\mathcal{SN}_{\mathtt{gc}}$ term will yield a non-$\mathcal{SN}_{\mathtt{gc}}$ term. So we are saying more than, "if M is $\xrightarrow[\lambda\mathtt{x}_{gc}]{}$ -infinite then M is $\rightsquigarrow$-infinite."

Definition 8 *The relation $\rightsquigarrow_e$ is the restriction of $\rightsquigarrow$ obtained by*

- *omitting rule $\mathcal{P}$gc, and*
- *in rule $\mathcal{P}$VarHd, reducing the T_i with smallest index among those permitting a $\rightsquigarrow_e$-reduction*

Clearly $\rightsquigarrow_e$ is an *effective* strategy for $\lambda\mathtt{x}$-reduction (hence the subscript e). We will see that it is perpetual.

Lemma 7 *If T admits a $\rightsquigarrow$-reduction, then T admits an $\rightsquigarrow_e$-reduction.*

> **Proof:** By induction on terms. If T admits a $\mathcal{P}$VarHd reduction, clearly it admits such a reduction on the leftmost of the T_i. It remains to prove that a term $T \equiv M\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$ with M not a variable admits a $\rightsquigarrow_e$-reduction. But inspection of the possible forms of M (see Lemma 2) shows that M itself will admit a $\rightsquigarrow$-reduction; by induction we may ensure that this is in fact a $\xrightarrow[\lambda\mathtt{x}]{}$ -reduction, and so this represents a $\rightsquigarrow_e$-reduction out of T itself. □

Corollary 4

1. *The relation $\rightsquigarrow_e$ is a perpetual strategy for $\lambda\mathbf{x}_{gc}$.*
2. *The relation $\rightsquigarrow_e$ is a perpetual strategy for $\lambda\mathbf{x}$.*
3. *A term is $\lambda\mathbf{x}$-infinite iff it is $\lambda\mathbf{x}_{gc}$-infinite. Equivalently, $\mathcal{SN}_{\lambda\mathbf{x}} = \mathcal{SN}_{\lambda\mathbf{x}_{gc}}$.*

Proof:

1. Follows from the theorem and Lemma 7.
2. Follows from the fact that $\lambda\mathbf{x}$ is a subsystem of $\lambda\mathbf{x}_{gc}$.
3. The non-trivial direction follows from the first part and the fact that $\rightsquigarrow_e$ is a sub-relation of $\lambda\mathbf{x}$.

□

5 An inductive characterization of the strongly normalizing terms.

In this section we give an inductive characterization of the class $\mathcal{SN}$ of terms which are strongly normalizing with respect to $\lambda\mathbf{x}$.

Definition 9 *The class of terms $\mathcal{S}$ is inductively defined by the following rules.*

$$\mathcal{S}Lmb \quad \frac{B}{\lambda x.B}$$

$$\mathcal{S}VarHd \quad \frac{A_1 \dots A_n}{v A_1 \cdots A_n}$$

$$\mathcal{S}\beta Hd \quad \frac{B\langle x = A\rangle\, \overrightarrow{T}}{(\lambda x.B) A \overrightarrow{T}}$$

$$\mathcal{S}I\ Clo \quad \frac{A\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}}{x\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}}$$

$$\mathcal{S}K\ Clo \quad \frac{x\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T} \qquad A}{x\langle y = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}} \quad x \not\equiv y$$

$$\mathcal{S}AbsClo \quad \frac{(\lambda y.B\langle x = A\rangle)\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}}{(\lambda y.B)\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}}$$

$$\mathcal{S}AppClo \quad \frac{(U\langle x = A\rangle V\langle x = A\rangle)\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}}{(UV)\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}}$$

Lemma 8 $\mathcal{SN} \subseteq \mathcal{S}$.

Proof: When $M \in \mathcal{SN}$ the relation $(\longrightarrow \cup \sqsupset)$ is well-founded out of M. Since $\mathcal{S}$ is syntax-directed, there is a unique rule of inference from the definition of $\mathcal{S}$ which could show $M \in \mathcal{S}$. So it suffices to observe that the term(s) M' comprising the hypotheses of the rule appropriate to M satisfy $M(\longrightarrow \cup \sqsupset)M'$. $\Box$

Lemma 9 $\mathcal{S} \subseteq \mathcal{SN}$.

Proof: It suffices to see that $\mathcal{SN}$ is closed under the rules of inference defining $\mathcal{S}$. This is elementary in every case except $\mathcal{S}\mathsf{K}\ \mathsf{Clo}$ and $\mathcal{S}\mathsf{AppClo}$: for these we use perpertuality of $\rightsquigarrow$. $\Box$

6 A type system for the strongly normalizing terms

For the classical λ-calculus the set of strongly normalizing terms can be characterized as precisely those terms assigned a type under a certain *intersection types* discipline. In this section we define the natural generalization of this system to the calculus $\lambda\mathbf{x}$ and prove that a term is $\lambda\mathbf{x}$-strongly normalizing if and only if it is typable.

The outline of the proof is the same as the classical one in the sense that it is based on a notion of *reducibility* (Definition 12). But the inductive definition of the set of strongly normalizing terms plays a key role (as it does, often implicitly, in proofs for the classical calculus) so it is here that we reap the rewards of the fine-grained analysis of the previous sections. The proof of the converse result that strongly normalizing terms are typable is also conveniently structured around the definition of $\mathcal{S}$. In a sense the proof is easier than the classical case, essentially because we do not have to analyze β-reduction.

The system of intersection types is due to Coppo and Dezani [10] and Sallé [20]. The fact that the strongly normalizing terms are precisely the typable terms seems to have been first proved in [18]. Our notation is consistent with that of [4], to which we refer the reader for background.

Definition 10 (The system of type assignment $\lambda^\cap$) *Given an infinite set of* type-variables*, the set of* types *is formed by closing the type-variables under the operations $\sigma \to \tau$ and $\sigma \cap \tau$.*

A statement *is an expression of the form $M : \tau$; where M, the* subject *of the statement, is a term and τ is a type. A* basis *is a set of declarations with distinct variables as subjects. A* judgement *is a triple Γ, M, τ where Γ is a basis, M is a term, and τ is a type; the notion of a judgement's being* derivable*, denoted $\Gamma \vdash M : \tau$ is given by the rules of inference in Table 1*

We say that a term M is typable *if there exists a Γ and a τ such that $\Gamma \vdash M : \tau$.*

Definition 11 *Let Γ_1 and Γ_2 be type-environments. The type-environment $\Gamma_1 \sqcap \Gamma_2$ contains $x : \sigma$ if either:*

- *$(x : \sigma)$ is in Γ_1 and $x \notin \mathcal{D}om(\Gamma_2)$, or*
- *$(x : \sigma)$ is in Γ_2 and $x \notin \mathcal{D}om(\Gamma_1)$, or*

$$\text{start}\ \frac{(x\colon\sigma)\in\Gamma}{\Gamma\vdash x\colon\sigma} \qquad \text{cut}\ \frac{x\colon\sigma,\Gamma\vdash M\colon\tau \qquad \Gamma\vdash N\colon\sigma}{\Gamma\vdash M\langle x=N\rangle\colon\tau}$$

$$\rightarrow\text{I}\ \frac{\Gamma,x\colon\sigma\vdash M\colon\tau}{\Gamma\vdash\lambda x.M\colon\sigma\rightarrow\tau} \qquad \rightarrow\text{E}\ \frac{\Gamma\vdash M\colon\sigma\rightarrow\tau \qquad \Gamma\vdash N\colon\sigma}{\Gamma\vdash(MN)\colon\tau}$$

$$\cap\text{-I}\ \frac{\Gamma\vdash M\colon\sigma_1 \qquad \Gamma\vdash M\colon\sigma_2}{\Gamma\vdash M\colon\sigma_1\cap\sigma_2} \qquad \cap\text{-E}\ \frac{\Gamma\vdash M\colon\sigma_1\cap\sigma_2}{\Gamma\vdash M\colon\sigma_i}\ \ i\in\{1,2\}$$

Table 1: Typing rules for $\lambda^{\cap}$

- $\sigma=\sigma_1\cap\sigma_2$ *with* $(x\colon\sigma_1)\in\Gamma_1$ *and* $(x\colon\sigma_2)\in\Gamma_2$.

Lemma 10 *If* $\Gamma\vdash M\colon\tau$ *then for all* Γ', $\Gamma\sqcap\Gamma'\vdash M\colon\tau$.

Proof: An easy induction over typing derivations. □

The next proposition collects various standard properties of type derivations and typable terms.

Lemma 11

1. *If* $\Gamma\vdash M\colon\tau$ *and* $\Gamma\subset\Gamma'$ *then* $\Gamma'\vdash M\colon\tau$.
2. *If* $\Gamma\vdash M\colon\tau$ *and* Γ' *agrees with* Γ *on the free variables of* M *then* $\Gamma'\vdash M\colon\tau$.
3. $\Gamma\vdash M\langle x=S\rangle\colon\tau$ *if and only if there is a* σ *such that* $x\colon\sigma,\Gamma\vdash M\colon\tau$ *and* $\Gamma\vdash S\colon\sigma$

Proof: Parts 1 and 2 are routine inductions.

For part 3, If the last inference is an instance of the cut rule we are done. Suppose the last inference is an instance of $\cap$-I. Then τ is $\tau_1\cap\tau_2$ and for $1\leq i\leq 2$ we have $\Gamma\vdash M\langle x=S\rangle\colon\tau_i$. By induction there are σ_i such that $x\colon\sigma_i,\Gamma\vdash M\colon\tau$ and $\Gamma\vdash S\colon\sigma_i$. But then $x\colon(\sigma_1\cap\sigma_2),\Gamma\vdash M\colon\tau$ by Lemma 10, and $\Gamma\vdash S\colon(\sigma_1\cap\sigma_2)$ by $\rightarrow$ I, completing the proof in this case. When the last inference is an instance of $\cap$-E the argument is similar. □

6.1 Typable terms are strongly normalizing

Definition 12 *For each type* τ *we define a set of terms* $\mathcal{R}_\tau$ *as follows:*

- $\mathcal{R}_t$ *is* $\mathcal{SN}$ *when* t *is a type variable*
- $\mathcal{R}_{\tau_1\cap\tau_2}$ *is* $\mathcal{R}_{\tau_1}\cap\mathcal{R}_{\tau_2}$
- $\mathcal{R}_{\alpha\rightarrow\beta}$ *is* $\{M\mid$ *if* A *is in* $\mathcal{R}_\alpha$, *then* MA *is in* $\mathcal{R}_\beta\ \}$

Lemma 12

1. $\mathcal{R}_\tau\subseteq SN$

2. *If M is not an abstraction, then each of the following is a sufficient condition for $M \in \mathcal{R}_\tau$:*

   - *M is a normal form*
   - *there exists M' with $M \rightsquigarrow M'$, $M' \in \mathcal{R}_\tau$.*

   Proof: By induction on types. At type-variables t the first claim holds by definition, and the second is just the statement that $\rightsquigarrow$ is perpetual. At intersection-types each claim is an immediate consequence of the induction hypothesis.

   When τ is $\alpha \to \beta$: To establish the first claim, suppose $M \in \mathcal{R}_{\alpha\to\beta}$, and note that as a consequence of the second claim at type α, each variable is in $\mathcal{R}_\alpha$. So $Mx \in \mathcal{R}_\beta$. Since this is SN by induction at type β, M is clearly SN.

   For the second claim let M be given, not an abstraction, and let A be in $\mathcal{R}_\alpha$: we seek $MA \in \mathcal{R}_\beta$, and we show this by induction over $\rightsquigarrow$-reduction out of A; this is well-founded since $\mathcal{R}_\alpha \subseteq \mathcal{SN}$.

   Since MA is not an abstraction it suffices to check the $\rightsquigarrow$-reducts of MA. But is easy to see that any such term is either of the form $M'A$ with $M \rightsquigarrow M'$ or MA' with $A \rightsquigarrow A'$. The former terms are in $\mathcal{R}_\beta$ since we assumed $M' \in \mathcal{R}_{\alpha\to\beta}$ and the latter are in $\mathcal{R}_\beta$ by the sub-induction hypothesis. □

In light of (the first part of) the previous lemma the fact that typable terms are strongly normalizing will follow if we can show that if M is typable with type τ then $M \in \mathcal{R}_\tau$. We note the folllowing lemma, whose use below motivate our work on gently-commuting reductions in Section 3.

Lemma 13 *If $M \longrightarrow M'$ via Comp and $M' \in \mathcal{R}_\tau$ then $M \in \mathcal{R}_\tau$.*

Proof: By induction on types. At type-variables t we must show that if M is infinite and M' is obtained from M by Comp-reduction then M' is infinite. This follows from the Gentle Commutation Lemma 3 since $\lambda\mathbf{x}$-reduction gently commutes with the reduction $\xrightarrow[\omega]{}$ (Lemma 6).

At intersection-types each claim is established as an immediate consequence of the induction hypothesis. When τ is $\alpha \to \beta$: let $M' \in \mathcal{R}_\tau$ be such that $M \longrightarrow M'$ via rule Comp. Let A be in $\mathcal{R}_\alpha$; we seek $MA \in \mathcal{R}_\beta$. But $M'A \in \mathcal{R}_\beta$ and $MA \longrightarrow M'A$ via rule Comp, so by induction $MA \in \mathcal{R}_\beta$ as desired. □

Theorem 2 *Let Γ be the basis $(x_1\colon \alpha_1), (x_2\colon \alpha_2), \ldots, (x_n\colon \alpha_n)$. Suppose*

- *$\Gamma \vdash M\colon \tau$, and*
- *$A_i \in \mathcal{R}_{\alpha_i}$ for $1 \le i \le n$, with $x_{i+j} \notin FV(A_i)$ for $1 \le i \le n$ and $j \ge 0$.*

Then $M\langle x_1 = A_1\rangle \ldots \langle x_n = A_n\rangle \in \mathcal{R}_\tau$.

Proof:

We induct on the derivation of $\Gamma \vdash M\colon \tau$. We consider the possible cases as to the last rule of inference in the derivation.

The start rule. Here, for some i, $M \equiv x_i$ and $\tau = \alpha_i$. To show that $x_i\langle x_1 = A_1\rangle ... \langle x_n = A_n\rangle \in \mathcal{R}_\tau$ we essentially iterate Lemma 12. Formally, we induct over the pair (k, n) where k is the sum of the lengths of the longest reductions out of the A_i. There are two cases. If $i = 1$ then by Lemma 12 it suffices to check that $A_1\langle x_2 = A_2\rangle \cdots \langle x_n = A_n\rangle$ is in $\mathcal{R}_\tau$. But since none of the indicated x_i is free in A_1 and each A_j is $\mathcal{SN}$, this term $\rightsquigarrow$-reduces (in $n-1$ steps) to A_1. Since this is in $\mathcal{R}_{\alpha_1}$ by hypothesis, we are done. If $i \neq 1$ then a $\rightsquigarrow$-reduction yields the term $x_i\langle x_2 = A_2\rangle \cdots \langle x_n = A_n\rangle$, which submits to the sub-induction hypothesis.

The rules $\cap$-I and $\cap$-E. These are each very easy applications of the induction hypothesis.

The $\rightarrow$-E rule. We have

$$\frac{\Gamma \vdash U : \alpha \rightarrow \beta \qquad \Gamma \vdash V : \alpha}{\Gamma \vdash UV : \beta}$$

To show that $(UV)\langle x_1 = A_1\rangle ... \langle x_n = A_n\rangle \in \mathcal{R}_\beta$, it suffices (by iterating Lemma 12) to argue that $(U\langle x_1 = A_1\rangle ... \langle x_n = A_n\rangle)(V\langle x_1 = A_1\rangle ... \langle x_n = A_n\rangle) \in \mathcal{R}_\beta$.

But by induction $(U\langle x_1 = A_1\rangle ... \langle x_n = A_n\rangle) \in \mathcal{R}_{\alpha\rightarrow\beta}$ and $(V\langle x_1 = A_1\rangle ... \langle x_n = A_n\rangle) \in \mathcal{R}_\alpha$ so the result follows by definition of $\mathcal{R}_{\alpha\rightarrow\beta}$.

The $\rightarrow$-I rule. We have

$$\frac{x : \alpha, \Gamma \vdash B : \beta}{\Gamma \vdash \lambda x.B : \alpha \rightarrow \beta}$$

We may assume that the variable x is not free in any of the A_i. To show that $\lambda x.B\langle x_1 = A_1\rangle ... \langle x_n = A_n\rangle \in \mathcal{R}_{\alpha\rightarrow\beta}$ we may iterate Lemma 12 and argue that $\lambda x.B\langle x_1 = A_1\rangle ... \langle x_n = A_n\rangle \in \mathcal{R}_{\alpha\rightarrow\beta}$. So choose $A \in \mathcal{R}_\alpha$, we seek $(\lambda x.B\langle x_1 = A_1\rangle ... \langle x_n = A_n\rangle)A \in \mathcal{R}_\beta$. Again by Lemma 12, it suffices to see that $B\langle x_1 = A_1\rangle ... \langle x_n = A_n\rangle\langle x = A\rangle \in \mathcal{R}_\beta$. But (since x is not free in any A_i) this is an application of the induction hypothesis applied to the derivation of $x : \alpha, \Gamma \vdash B : \beta$.

The cut rule. Here we have

$$\frac{x : \sigma, \Gamma \vdash M : \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\langle x = N\rangle : \tau}$$

We wish to show that $M\langle x = N\rangle\langle x_1 = A_1\rangle ... \langle x_n = A_n\rangle \in \mathcal{R}_\tau$; we may assume without loss of generality that x is not free in any of the A_i.

By iterating Lemma 13 we see that it suffices to show the following term to be in $\mathcal{R}_\tau$:

$$M\langle x_1 = A_1\rangle ... \langle x_n = A_n\rangle \langle x = N\langle x_1 = A_1\rangle ... \langle x_n = A_n\rangle\rangle$$

By the induction hypothesis applied to the derivation $\Gamma \vdash N : \sigma$, the term $N\langle x_1 = A_1\rangle ... \langle x_n = A_n\rangle$ is in $\mathcal{R}_\sigma$. Then since x is not free in any of the A_i, we may use the induction hypothesis applied to the derivation $x : \sigma, \Gamma \vdash M : \tau$ to finish the argument. □

Corollary 5 *If there exists* Γ *such that* $\Gamma \vdash M : \tau$ *then* M *is strongly normalizing.*

Proof: An easy consequence of the previous theorem and the fact that $\mathcal{R}_\tau \subseteq \mathcal{SN}$. □

6.2 Strongly normalizing terms are typable

Theorem 3 *Suppose that* M *is* $\lambda\mathbf{x}$*-strongly normalizing. Then there exists* Γ *and* σ *such that* $\Gamma \vdash M : \sigma$

Proof: We prove that if M is in $\mathcal{S}$ then there exist Γ and τ such that $\Gamma \vdash M : \tau$. This amounts to showing that the typable terms are closed under the rules defining $\mathcal{S}$. We consider each rule from Definition 9 in turn: in each case we assume that the hypotheses of the rule are typable and show that the conclusion is typable.

$\mathcal{S}$Lmb. Here M is $\lambda x.B$. We are given Γ and τ with $\Gamma \vdash B : \tau$. There are two sub-cases. If there is a σ such that $\Gamma \equiv x : \sigma, \Gamma'$ then $\Gamma' \vdash \lambda x.B : (\sigma \to \tau)$. Otherwise, let s be a type-variable not occurring in Γ. Then $x : s, \Gamma \vdash B : \tau$ by Lemma 11. 1, and so $\Gamma \vdash \lambda x.B : (s \to \tau)$.

$\mathcal{S}$VarHd. Here M is $vA_1 \cdots A_n$. For each i, $1 \leq i \leq n$ we are given Γ_i and τ_i with $\Gamma_i \vdash A_i : \tau_i$. Let t be a fresh type-variable and let φ be the type $(\tau_1 \to \cdots \to \tau_n \to t)$. Now set Γ to be $\Gamma_1 \sqcap \cdots \sqcap \Gamma_n \sqcap \{(v : \varphi)\}$. Then by Lemma 10 $\Gamma \vdash A_i : \tau_i$ for each i, and $\Gamma \vdash v : \varphi$, so in fact $\Gamma \vdash vA_1 \cdots A_n : t$.

$\mathcal{S}\beta$Hd. Here M is $(\lambda x.B)A\overrightarrow{T}$. It clearly suffices to show that for any Γ,

$$\Gamma \vdash B\langle x = A\rangle : \tau \qquad \text{implies} \qquad \Gamma \vdash (\lambda x.B)A : \tau.$$

By Lemma 11.3 we have α such that $x : \alpha, \Gamma \vdash B : \tau$ and $\Gamma \vdash A : \alpha$ But then $\to$I yields $\Gamma \vdash \lambda x.B : \alpha \to \tau$ and the result follows by an application of $\to$E.

$\mathcal{S}$I Clo. Here M is $x\langle x = A\rangle\langle z = \boldsymbol{S}\rangle \boldsymbol{T}$. Again it suffices to show that for any Γ,

$$\Gamma \vdash A : \tau \qquad \text{implies} \qquad \Gamma \vdash x\langle x = A\rangle : \tau$$

This is easy to see by inspecting the form of the cut typing rule.

$\mathcal{S}$K Clo. Here M is $y\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}, x \not\equiv y$. Because the inference rule witnessing $M \in \mathcal{S}$ has two hypotheses, this case is a little more subtle than previous cases. By the induction hypothesis we have Γ_1, Γ_2, τ and α such that $\Gamma_1 \vdash x\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T} : \tau$ and $\Gamma_2 \vdash A : \alpha$. Now y is not free in M and so we may assume without loss of generality that y is not a subject of Γ_1 or of Γ_2. Let Γ be $\Gamma_1 \sqcap \Gamma_2$. Then by Lemma 10 $(y : \alpha), \Gamma : x\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T} \vdash \tau$ and $\Gamma \vdash A : \alpha$. So by the cut rule $\Gamma \vdash M : \tau$.

$\mathcal{S}$AbsClo. Here M is $(\lambda y.B)\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$. It suffices to show that for any Γ,

$$\Gamma \vdash \lambda y.B\langle x = A\rangle : \tau \qquad \text{implies} \qquad \Gamma \vdash (\lambda y.B)\langle x = A\rangle : \tau$$

We may assume that y is not free in A. Arguing now by induction over the given typing-derivation, if the last rule was an instance of $\cap$-I or $\cap$-E the argument is an immediate application of the induction hypothesis. Otherwise τ is of the form $\tau_1 \to \tau_2$ and $(y : \tau_1), \Gamma \vdash B\langle x = A\rangle : \tau_2$. By Lemma 11. 3. there is a γ such that $(x : \gamma), (y : \tau_1), \Gamma \vdash B : \tau_2$ and $(y : \tau_1), \Gamma \vdash A : \gamma$. So $(x : \gamma), \Gamma \vdash B\langle y = A\rangle : \tau_2$, and since y is not free in A, $\Gamma \vdash A : \gamma$. Then the cut rule yields $\Gamma \vdash (\lambda y.B)\langle x = A\rangle : \tau$.

$\mathcal{S}$AppClo. Here M is $(UV)\langle x = A\rangle\langle \boldsymbol{z} = \boldsymbol{S}\rangle\, \boldsymbol{T}$. It suffices to show that for any Γ,

$$\Gamma \vdash (U\langle x = A\rangle V\langle x = A\rangle) : \tau \qquad \text{implies} \qquad \Gamma \vdash (UV)\langle x = A\rangle : \tau$$

Again arguing by induction over the given typing-derivation, the non-trivial case is when the typing rule applied was $\rightarrow$-eliminiation. We have

$$\Gamma \vdash U\langle x = A\rangle : \sigma \to \tau \qquad \text{and } \Gamma \vdash V\langle x = A\rangle : \sigma.$$

By Lemma 11. 3. there are α_1 and α_2 with

$$(x : \alpha_1), \Gamma \vdash U : \sigma \to \tau, \qquad \Gamma \vdash A : \alpha_1,$$

and

$$(x : \alpha_2), \Gamma \vdash V : \sigma, \qquad \Gamma \vdash A : \alpha_2.$$

Let Γ' be $\Gamma, (x : \alpha_1 \sqcap \alpha_2)$. Then

$$\Gamma' \vdash U : \sigma \to \tau, \Gamma' \vdash V : \sigma, and \Gamma' \vdash A : \alpha_1 \sqcap \alpha_2,$$

so

$$\Gamma' \vdash (UV) : \tau \text{ and } \Gamma \vdash (UV)\langle x = A\rangle : \tau.$$

□

7 Conclusion

In this paper, we have provided tools for a direct proof of strong normalization of typed terms in a calculus of explicit substitutions. These tools include an effective perpetual strategy and an inductive characterization of strongly normalizing terms. The kernel of the paper is a lemma (Lemma 6) which leads to a refinement of the classical Substitution Lemma.

Directions for further research. We plan to extend these results to characterize, as for the classical calculus, the (weakly) normalizing terms and the solvable terms respectively as the terms typable in suitable refinements of the intersection-types discipline.

A line of inquiry that we view as being particularly important is the definition of an appropriate notion of *standard reduction.* The classical theorem that if M reduces to N then M reduces to N via a standard reduction is key to the proof of correctness of "weak reduction" as an implementation of functional programming languages [17], see also [11]. It seems rather difficult to find a successful definition of standard reduction in explcit substitution calculi.

Finally, it is reasonable to hope that the combinatorial techniques and results derived here can lead to a better understanding of normalization properties in the presence of substitution-composition.

Acknowledgements

The authors are grateful to Nachum Dershowitz, Frédéric Lang, and Kristoffer Rose for many helpful discussions.

References

[1] Martìn Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. In Paul Hudak, editor, *POPL '90—Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pages 31–46, San Francisco, California, January 1990. ACM.

[2] Martìn Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.

[3] H. P. Barendregt. *The Lambda-Calculus, its syntax and semantics.* Studies in Logic and the Foundation of Mathematics. Elsevier Science Publishers B. V. (North-Holland), Amsterdam, 1984. Second edition.

[4] H. P. Barendregt. Lambda calculi with types. In Samson Abramsky, Dov M. Gabby, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, chapter 2, pages 117–309. Oxford University Press, 1992.

[5] Z. Benaissa, D. Briaud, P. Lescanne, and J. Rouyer-Degli. λv, a calculus of explicit substitutions which preserves strong normalisation. *Journal of Functional Programming*, 6(5):699–722, September 1996.

[6] R. Bloo. *Preservation of Termination for Explicit Substitution.* PhD thesis, Eindhoven University of Technology, Netherlands, 1997.

[7] Roel Bloo and J. Herman Geuvers. Explicit substitution: on the edge of strong normalisation. Computing Science Reports 96–10, Eindhoven University of Technology, P.O.box 513, 5600 MB Eindhoven, The Netherlands, April 1996. To appear in Theoretical Computer Science.

[8] Roel Bloo and Kristoffer Høgsbro Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *CSN '95—Computing Science in the Netherlands*, pages 62–72, Koninklijke Jaarbeurs, Utrecht, November 1995.

[9] Roel Bloo and Kristoffer Høgsbro Rose. Preservation of strong normalisation in named lambda calculi with explicit substitution and garbage collection. In *CSN '95 – Computer Science in the Netherlands*, pages 62–72, November 1995.

[10] M. Coppo and M. Dezani-Ciancaglini. A new type assignment for lambda-terms. *Archive f. math. Logic u. Grundlagenforschung*, 19:139–156, 1978.

[11] Pierre-Louis Curien, Thérèse Hardin, and Jean-Jacques Lévy. Confluence properties of weak and strong calculi of explicit substitutions. *Journal of the ACM*, 43(2):362–397, March 1996.

[12] H. B Curry and R. Feys. *Combinatory Logic I.* North-Holland, Amsterdam, 1958.

[13] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476, 1979.

[14] J.-Y. Girard. Interpretation fonctionelle et elimination des coupures de l'arithmetique d'ordre superieur. These D'Etat, Universite Paris VII, 1972.

[15] Pierre Lescanne. From $\lambda\sigma$ to $\lambda\upsilon$: a journey through calculi of explicit substitutions. In Hans-J. Boehm, editor, *POPL '94—21st Annual ACM Symposium on Principles of Programming Languages*, pages 60–69, Portland, Oregon, January 1994. ACM.

[16] Paul-André Melliès. Typed λ-calculi with explicit substitution may not terminate. In M. Dezani, editor, *TLCA '95—Int. Conf. on Typed Lambda Calculus and Applications*, volume 902 of *Lecture Notes in Computer Science*, pages 328–334, Edinburgh, Scotland, April 1995. Springer-Verlag.

[17] G.D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[18] G Pottinger. A type assignment for the strongly normalizable λ-terms. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 561–578. Academic Press, 1980.

[19] Kristoffer Høgsbro Rose. *Operational Reduction Models for Functional Programming Languages.* PhD thesis, DIKU, Universitetsparken 1, DK-2100 København Ø, February 1996. DIKU report 96/1.

[20] P. Sallé. Une extension de la théorie des types en λ-calcul. In G. Ausiello and C. Bohm, editors, *Fifth Colloquium on Automata, Languages and Programming*, volume 62 of *Lecture Notes in Computer Science*, pages 398–410. Springer-Verlag, July 1978.