A short proof that adding some permutation rules to β preserves SN

René David
LAMA - Equipe LIMD - Université de Chambéry
e-mail : rene.david@univ-savoie.fr

August 20, 2010

Abstract

I show that, if a term is SN for β, it remains SN when some permutation rules are added.

1 Introduction

Strong normalization (abbreviated as SN) is a property of rewriting systems that is often desired. Since about 10 years many researchers have considered the following question : If a λ-term is SN for the β-reduction, does it remain SN if some other reduction rules are added ? They are mainly interested with permutation rules they introduce to be able to delay some β-reductions in, for example, *let* $x = ...$ *in* ... constructions or in *calculi with explicit substitutions*. Here are some papers considering such permutations rules: L. Regnier [7], F. Kamareddine [3], E. Moggi [5], R. Dyckhoff and S. Lengrand [2], A. J. Kfoury and J. B. Wells [4], Y. Ohta and M. Hasegawa [6], J. Espírito Santo [8], [9], and [10].

Some of these papers show that SN is preserved by the addition of the permutation rules they introduce but, most often, authors do not consider the whole set of rules or add restrictions to some rules. For example the rule $(M\ (\lambda x.N\ P)) \triangleright (\lambda x.(M\ N)\ P)$ is often restricted to the case when M is an abstraction (in this case it is usually called $assoc$).

I give here a simple and short proof that the permutations rules preserve SN when they are added all together and with no restriction. It is done as follows. I show that every term which is typable in the system (often called system $\mathcal{D}$) of types built with $\rightarrow$ and $\wedge$ is strongly normalizing for all the rules (β and the permutation rules). Since it is well known that a term is SN for the β-rule iff it is typable in this system, the result follows. The proof is an extension of my proof of SN for the simply typed λ-calculus where the main result is a substitution theorem (here Theorem 3.3): if t and a are in SN, then so is $t[x := a]$.

To my knowledge, only one other paper ([9] and its recent version [10]) considers all the rules with no restriction. The technic used there is completely different from the one used in this paper.

2 Definitions and notations

Definition 2.1 • *The set of λ-terms is defined by the following grammar*

$$\mathcal{M} := x \mid \lambda x.\mathcal{M} \mid (\mathcal{M}\ \mathcal{M})$$

- *The set $\mathcal{T}$ of types is defined (simultaneously with the set $\mathcal{S}$ of simple types) by the following grammars where $\mathcal{A}$ is a set of atomic constants*

$$\mathcal{S} ::= \mathcal{A} \mid \mathcal{T} \to \mathcal{S}$$
$$\mathcal{T} ::= \mathcal{S} \mid \mathcal{S} \wedge \mathcal{T}$$

- *The typing rules are the following where Γ is a set of declarations as $x : A$ where x is a variable and the mentioned types (A, B) are in $\mathcal{T}$:*

$$\frac{}{\Gamma, x : A \vdash x : A}$$

$$\frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash (M\ N) : B} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B}$$

$$\frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash M : A} \qquad \frac{\Gamma \vdash M : A \wedge B}{\Gamma \vdash M : B}$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash M : B}{\Gamma \vdash M : A \wedge B}$$

Remarks and Notation

1. To avoid too many brackets in the lambda terms I will adopt the following conventions. An application (or a sequence of applications) is always surrounded by brackets (i.e. the application of M to N is written $(M\ N)$ with a blank between M and N) and, as usual, application associates to the left i.e. $(M\ N\ P)$ means $((M\ N)\ P)$. An abstraction is always written as $\lambda x.M$ (i.e. there is a dot after the variable but no blank between the dot and M) where either M is a letter or an application (and thus between brackets) or another abstraction.

   For example $\lambda y.(MN)$ represents an abstraction and $(\lambda y.MN)$ a redex.

2. Note that in the usual definition of the types with intersection $\to$ and $\wedge$ can be used with no restriction. Here we forbid to have an $\wedge$ at the right of an $\to$. For example $A \to (B \wedge C)$ is forbidden and must be replaced by $(A \to B) \wedge (A \to C)$. It is well known that both systems are equivalent since it is easily proved that any type derivation in the unrestricted system can be transformed into a type derivation in the restricted one. Actually note that, in fact, the type derivation given by Theorem 3.2 already satisfies this restriction.

   We have used this restricted version to make simpler the analysis of type derivations in the proof of Theorem 3.3

3. Also note (this is well known and easy to prove) that any type derivation can be transformed into a normal derivation i.e. a derivation in which the introduction of an $\wedge$ is never immediately followed by its elimination.

4. The lemmas and theorems using types will be indicated by the mention "typed". If a type derivation is given to M, $type(M)$ will denote the size (i.e the number of symbols) of the type of M.

Definition 2.2 *The reduction rules are the following.*

- β : $(\lambda x.M\ N) \triangleright M[x := N]$
- δ : $(\lambda y.\lambda x.M\ N) \triangleright \lambda x.(\lambda y.M\ N)$
- γ : $(\lambda x.M\ N\ P) \triangleright (\lambda x.(M\ P)\ N)$
- $assoc$: $(M\ (\lambda x.N\ P)) \triangleright (\lambda x.(M\ N)\ P)$

Using Barendregt's convention for the names of variables, we assume that, in γ (resp. δ, $assoc$), x is not free in P (resp. in N, in M).

The rules δ and γ have been introduced by Regnier in [7] and are called there the σ-reduction. It seems that the first formulation of *assoc* appears in Moggi [5] in the restricted case where M is an abstraction and in a "*let ... in ...*" formulation.

Note that γ (resp. δ, $assoc$) are called θ_1 (resp. γ, θ_3) in [4] and π_1 or σ_1 (resp. σ_2, π_2) in [10].

Notation 2.1

- *If t is a term, $size(t)$ denotes its size.*
- *If $t \in SN$ (i.e. every sequence of reductions starting from t is finite), $\eta(t)$ denotes the length of the longest reduction of t. Since various notions of reductions are considered in this paper, by default these concepts are relative to the union of all four reduction rules. When this is not the case (e.g. SN wrt to β), then the reduction rule intended is indicated explicitly.*
- *Let σ be a substitution. We say that σ is fair if the $\sigma(x)$ for $x \in dom(\sigma)$ all have the same type (that will be denoted as $type(\sigma)$). We say that $\sigma \in SN$ if, for each $x \in dom(\sigma)$, $\sigma(x) \in SN$.*
- *Let $\sigma \in SN$ be a substitution and t be a term. We denote by $size(\sigma, t)$ (resp. $\eta(\sigma, t)$) the sum, over $x \in dom(\sigma)$, of $nb(t, x).size(\sigma(x))$ (resp. $nb(t, x).\eta(\sigma(x))$) where $nb(t, x)$ is the number of free occurrences of x in t.*
- *If $\overrightarrow{M}$ is a sequence of terms, $lg(\overrightarrow{M})$ denotes its length, $M(i)$ denotes the i-th element of the sequence and $tail(\overrightarrow{M})$ denotes $\overrightarrow{M}$ from which the first element has been deleted.*
- *Assume $t = (H\ \overrightarrow{M})$ where H is an abstraction or a variable and $lg(\overrightarrow{M}) \geq 1$.*
  - *If H is an abstraction (in this case we say that t is β-head reducible), then $M(1)$ will be denoted as $Arg[t]$ and $(R'\ tail(\overrightarrow{M}))$ will be denoted by $B[t]$ where R' is the reduct of the β-redex $(H\ Arg[t])$.*
  - *If $H = \lambda x.N$ and $lg(\overrightarrow{M}) \geq 2$ (in this case we say that t is γ-head reducible), then $(\lambda x.(N\ M(2))\ M(1)\ M(3)\ ...\ M(lg(\overrightarrow{M})))$ will be denoted by $C[t]$.*
  - *If $H = \lambda x.\lambda y.N$ (in this case we say that t is δ-head reducible), then $(\lambda y.(\lambda x.N\ M(1))\ M(2)\ ...\ M(lg(\overrightarrow{M})))$ will be denoted by $D[t]$.*
  - *If $M(i) = (\lambda x.N\ P)$, then the term $(\lambda x.(H\ M(1)\ ...\ M(i-1)\ N)\ P\ M(i+1)\ ...\ M(lg(\overrightarrow{M})))$ will be denoted by $A[t, i]$ and we say that $M(i)$ is the β-redex put in head position.*
- *Finally, in a proof by induction, IH will denote the induction hypothesis.*

3 The theorem

Theorem 3.1 *Let t be a term. Assume t is strongly normalizing for β. Then t is strongly normalizing for β, δ, γ and assoc.*

Proof This follows immediately from Theorem 3.2 and corollary 3.1 below. □

Theorem 3.2 *A term is SN for the β-rule iff it is typable in system $\mathcal{D}$.*

Proof This is a classical result. For the sake of completeness I recall here the proof of the only if direction given in [1]. Note that it is the only direction that is used in this paper and that corollary 3.1 below actually gives the other direction. The proof is by induction on $\langle \eta(t), size(t)\rangle$.

- If $t = \lambda x\ u$. This follows immediately from the IH.

- If $t = (x\ v_1\ ...\ v_n)$. By the IH, for every j, let $x : A_j, \Gamma_j \vdash v_j : B_j$. Then $x : \bigwedge A_j \wedge (B_1, ..., B_n \to C)$, $\bigwedge \Gamma_j \vdash t : C$ where C is any type, for example any atomic type.

- If $t = (\lambda x.a\ b\ \overrightarrow{c})$. By the IH, $(a[x := b]\ \overrightarrow{c})$ is typable. If x occurs in a, let $A_1\ ...\ A_n$ be the types of the occurrences of b in the typing of $(a[x := b]\ \overrightarrow{c})$. Then t is typable by giving to x and b the type $A_1 \wedge ... \wedge A_n$. Otherwise, by the induction hypothesis b is typable of type B and then t is typable by giving to x the type B. □

From now on, $\triangleright$ denotes the reduction by one of the rules β, δ, γ and assoc.

Lemma 3.1
1. *The system satisfies subject reduction i.e. if $\Gamma \vdash t : A$ and $t \triangleright t'$ then $\Gamma \vdash t' : A$.*
2. *If $t \triangleright t'$ then $t[x := u] \triangleright t'[x := u]$.*
3. *If $t' = t[x := u] \in SN$ then $t \in SN$ and $\eta(t) \leq \eta(t')$.*

Proof Immediate. □

Lemma 3.2 *Let $t = (H\ \overrightarrow{M})$ be such that H is an abstraction or a variable and $lg(\overrightarrow{M}) \geq 1$. Assume $H, \overrightarrow{M} \in SN$ and that*

1. *If t is δ-head reducible (resp. γ-head reducible, β-head reducible), then $D[t] \in SN$ (resp. $C[t] \in SN$, $Arg[t], B[t] \in SN$).*
2. *For each i such that $M(i)$ is a β-redex, $A[t, i] \in SN$,*

Then $t \in SN$.

Proof By induction on $\eta(H) + \sum \eta(M(i))$. Show that each reduct of t is in SN. Note that the assumption $H, \overrightarrow{M} \in SN$ is implied by the others if at least one of them is not "empty" i.e. if t is head reducible for at least one rule. □

Lemma 3.3 (typed) *If $(t\ \overrightarrow{u}) \in SN$ then $(\lambda x.t\ x\ \overrightarrow{u}) \in SN$.*

Proof Note that, if $(\lambda x.t\ x\ \overrightarrow{u})$ has a head redex for the δ-rule, its reduct has not the desirable shape and an induction hypothesis will not be applicable. We thus generalize a bit the statement with the notion of left context, i.e. a context with exactly one hole on the left branch. More precisely the set $\mathcal{L}$ of left contexts is defined by the following grammar: $\mathcal{L} := [] \mid \lambda x.\mathcal{L} \mid (\mathcal{L}\ \mathcal{M})$. The result is thus a special case of the following claim.

Claim : Let L be a left context and t be a term. If $L[t]$ is in SN then so is $w = L[(\lambda x.t\ x)]$.

Proof : By induction on $\langle type(t), \eta(L[t])\rangle$. We show that every reduct of w is in SN. There are 4 possibilities for the reduced redex. If it is in L or in t, the result follows immediately from the IH. If it is the $(\lambda x.t\ x)$ substituted in the hole of L the result is clear. The last situation is when the redex is created by the substitution in the hole of L. These cases are given below. Note that the *assoc* and β rules can only be used either in t or in L.

- $t = \lambda y.t_1$ and $w \triangleright_\delta L[\lambda y.(\lambda x.t_1\ x)] = L'[(\lambda x.t_1\ x)]$ where $L' = L[\lambda y.[]]$. The result follows from the IH applied to L' and t_1 (since t_1 can be given a type less than the one of t).

- $L = L'[([]\ v)]$ and $w \triangleright_\gamma L'[(\lambda x.(t\ v)\ x)]$. The result follows from the IH applied to L' and $t_1 = (t\ v)$ (since t_1 can be given a type less than the one of t). □

Theorem 3.3 (typed) *Let $t \in SN$ and $\sigma \in SN$ be a fair substitution. Then $\sigma(t) \in SN$.*

Proof Formally, what we prove is the following. Let $U = \{(t, \sigma, A) \mid t \in SN$, $\sigma \in SN$ and A is assignable to each $\sigma(x)\}$. Then, for all $(t, \sigma, A) \in U$, $\sigma(t) \in SN$. Theorem follows since, if σ is fair, $(t, \sigma, A) \in U$ for some A .

We assume all the derivations are normal (see the remark after definition 2.1). The proof is by induction on $\langle size(A), \eta(t), size(t), \eta(\sigma, t), size(\sigma, t)\rangle$. We will have to use the induction hypothesis to some (t', σ', A') for which we have to give type derivations and to show that the 5-uplet has decreased. For the types (since the verification is fastidious but easy) we give some details only for one example (the first time in case 1.c below) and, for the others, we simply say "$type(t_1) < type(t_2)$" (resp. "$type(t_1) = type(t_2)$") instead of saying something as "t_1 can be given a type less than (resp. equal to) $type(t_2)$".

Note that this theorem will be only used with unary substitutions but its proof needs the general case because, starting with a unary substitution, it may happen that we have to use the induction hypothesis with a non unary substitution. It will be the case, for example, in 1.c below.

Let $(t, \sigma, A) \in U$. If t is an abstraction or a variable the result is trivial. Thus assume $t = (H\ \overrightarrow{M})$ where H is an abstraction or a variable and $n = lg(\overrightarrow{M}) \geq 1$. Let $\overrightarrow{N} = \sigma(\overrightarrow{M})$.
Claim : Let $\overrightarrow{P}$ be a (strict) initial or a final sub-sequence of $\overrightarrow{N}$. Then $(z\ \overrightarrow{P}) \in SN$.
Proof : Let $\overrightarrow{Q}$ be the sub-sequence of $\overrightarrow{M}$ corresponding to $\overrightarrow{P}$. Then $(z\ \overrightarrow{P}) = \tau(t')$ where $t' = (z\ \overrightarrow{Q})$ and τ is the same as σ for the variables in $\overrightarrow{Q}$ and $z \notin dom(\tau)$. The result follows from the IH since $size(t') < size(t)$. □

We use Lemma 3.2 to show that $\sigma(t) \in SN$.

1. Assume $\sigma(t)$ is δ-head reducible. We have to show that $D[\sigma(t)] \in SN$. There are 3 cases to consider.

   (a) If t was already δ-head reducible, then $D[\sigma(t)] = \sigma(D[t])$ and the result follows from the IH.

   (b) If H is a variable and $\sigma(H) = \lambda x.\lambda y.a$, then $D[\sigma(t)] = t'[z := \lambda y.(\lambda x.a\ N(1))]$ where $t' = (z\ tail(\overrightarrow{N}))$. By the claim, $t' \in SN$ and since $type(z) < size(A)$ it is enough, by the IH, to check that $\lambda y.(\lambda x.a\ N(1)) \in SN$. But this is $\lambda y.(z'\ N(1))[z' := \lambda x.a]$. But, by the claim, $(z'\ N(1)) \in SN$ and we conclude by the IH since $type(z') < size(A)$.

   (c) If $H = \lambda x.z$ and $\sigma(z) = \lambda y.a$, then $D[\sigma(t)] = (\lambda y.(\lambda x.a\ N(1))\ tail(\overrightarrow{N})) = \tau(t')$ where $t' = (z'\ tail(\overrightarrow{M}))$ and τ is the same as σ on the variables of $tail(\overrightarrow{M})$ and $\tau(z') = \lambda y.(\lambda x.a\ N(1))$. Note that, by Lemma 3.1, t' is in SN and $\eta(t') \leq \eta(t)$. Since $size(t') < size(t)$ to get the result by the IH we have to show that (1) $(t', \tau, A) \in U$ and (2) that $(\lambda x.a\ N(1)) \in SN$.
   To prove (1) it is enough to show that we can give to $Q = \lambda y.(\lambda x.a\ M(1))$ the same type as $P = (\lambda x.\lambda y.a\ M(1))$. In the typing of P, $\lambda x.\lambda y.a$ has type $(A_1 \to B_1 \to C_1) \wedge ... \wedge (A_k \to B_k \to C_k)$ and $M(1)$ has type $A_1 \wedge ... \wedge A_k$ and thus P has type $(B_1 \to C_1) \wedge ... \wedge (B_k \to C_k)$. It follows that we can type Q by typing $(\lambda x.a\ M(1))$ with type $C_1 \wedge ... \wedge C_k$ and thus Q with type $(B_1 \to C_1) \wedge ... \wedge (B_k \to C_k)$.
   To prove (2) we remark that $(\lambda x.a\ N(1)) = (\lambda x.z''\ N(1))[z'' := a]$ and, since $type(a) < size(A)$ it is enough, by the IH, to show that $u = (\lambda x.z''\ N(1)) \in SN$. This is done as follows: $u = \sigma'(t'')$ where $t'' = (\lambda x.z''\ M(1))$ (which is, up to the renaming of z into z'' a sub-term

of t) and σ' is as σ but where z'' is not in the domain of σ' whereas the occurrence of z in H was in the domain of σ. Thus, $size(\sigma', t'') < size(\sigma, t)$ and the result follows from the IH.

2. Assume $\sigma(t)$ is γ-head reducible. We have to show that $L[\sigma(t)] \in SN$. There are 4 cases to consider.

   (a) If H is an abstraction, then $C[\sigma(t)] = \sigma(C[t])$ and the result follows immediately from the IH.

   (b) H is a variable and $\sigma(H) = \lambda y.a$, then $C[\sigma(t)] = (\lambda y.(a\ N(2))\ N(1)\ N(3)\ ...\ N(n)) = (\lambda y.(a\ N(2))\ y\ N(3)\ ...\ N(n))[y := N(1)]$. Since $type(N(1)) < size(A)$, it is enough, by the IH, to show $(\lambda y.(a\ N(2))\ y\ N(3)\ ...\ N(n)) \in SN$ and so, by Lemma 3.3, that $u = (a\ N(2)\ N(3)\ ...\ N(n)) \in SN$. By the claim, $(z\ tail(\overrightarrow{N})) \in SN$ and the result follows from the IH since $u = (z\ tail(\overrightarrow{N}))[z := a]$ and $type(a) < size(A)$.

   (c) H is a variable and $\sigma(H) = (\lambda y.a\ b)$, then $C[\sigma(t)] = (\lambda y.(a\ N(1))\ b\ N(2)\ ...\ N(n)) = (z\ tail(\overrightarrow{N}))[z := (\lambda y.(a\ N(1))\ b)]$. Since $type(z) < size(A)$, by the IH it is enough to show that $u = (\lambda y.(a\ N(1))\ b) \in SN$. We use Lemma 3.2.

   - We first have to show that $B[u] \in SN$. But this is $(a[y := b]\ N(1))$ which is in SN since $u_1 = (a[y := b]\ \overrightarrow{N}) \in SN$ since $u_1 = \tau(t_1)$ where t_1 is the same as t but where we have given to the variable H the fresh name z, τ is the same as σ for the variables in $dom(\sigma)$ and $\tau(z) = a[y := b]$ and thus we may conclude by the IH since $\eta(\tau, t) < \eta(\sigma, t)$.

   - We then have to show that, if b is a β-redex say $(\lambda z.b_1\ b_2)$, then $A[u, 1] = (\lambda z.(\lambda y.a\ N(1)\ b_1)\ b_2) \in SN$. Let $u_2 = \tau(t_2)$ where t_2 is the same as t but where we have given to the variable H the fresh name z, τ is the same as σ for the variables in $dom(\sigma)$ and $\tau(z) = A[\sigma(H), 1]$. By the IH, $u_2 \in SN$. Note that that $t_2 \in SN$, $\eta(t_2) \leq \eta(t)$ by Lemma 3.1 and that $\eta(\tau, t2) < \eta(\sigma, t)$. But $u_2 = (\lambda z.(\lambda y.a\ b_1)\ b_2\ \overrightarrow{N})$ and thus $u_3 = (\lambda z.(\lambda y.a\ b_1)\ b_2\ \ N(1)) \in SN$. Since u_3 reduces to $A[u, 1]$ by using twice by the γ rule, it follows that $A[u, 1] \in SN$.

   (d) If H is a variable and $\sigma(H)$ is γ-head reducible, then $C[\sigma(t)] = \tau(t')$ where t' is the same as t but where we have given to the variable H the fresh name z and τ is the same as σ for the variables in $dom(\sigma)$ and $\tau(z) = C[\sigma(H)]$. The result follows then from the IH since $\eta(\tau, t') < \eta(\sigma, t)$.

3. Assume that $\sigma(t)$ is β-head reducible. We have to show that $Arg[\sigma(t)] \in SN$ and that $B[\sigma(t)] \in SN$. There are 3 cases to consider.

   (a) If H is an abstraction, the result follows immediately from the IH since then $Arg[\sigma(t)] = \sigma(Arg[t])$ and $B[\sigma(t)] = \sigma(B[t])$.

   (b) If H is a variable and $\sigma(H) = \lambda y.v$ for some v. Then $Arg[\sigma(t)] = N(1) \in SN$ by the IH and $B[\sigma(t)] = (v[y := N(1)]\ tail(\overrightarrow{N})) = (z\ tail(\overrightarrow{N}))[z := v[y := N(1)]]$. By the claim, $(z\ tail(\overrightarrow{N})) \in SN$. By the IH, $v[y := N(1)] \in SN$ since $type(N(1)) < size(A)$. Finally the IH implies that $B[\sigma(t)] \in SN$ since $type(v) < size(A)$.

   (c) H is a variable and $\sigma(H) = (R\ \overrightarrow{M'})$ where R is a β-redex. Then $Arg[\sigma(t)] = Arg[\sigma(H)] \in SN$ and $B[\sigma(t)] = (R'\ \overrightarrow{M'}\ \overrightarrow{N})$ where R' is the reduct of R. But then $B[\sigma(t)] = \tau(t')$ and t' is the same as t but

where we have given to the variable H the fresh name z and τ is the same as σ for the variables in $dom(\sigma)$ and $\tau(z) = (R' \ \overrightarrow{M'})$. Note that that $t' \in SN$ and $\eta(t') \leq \eta(t)$, by Lemma 3.1. We conclude by the IH since $\eta(\tau, t') < \eta(\sigma, t)$.

4. We, finally, have to show that, for each i, $A[\sigma(t), i] \in SN$. There are again 3 cases to consider.

   (a) If the β-redex put in head position is some $N(j)$ and $M(j)$ was already a redex. Then $A[\sigma(t), j] = \sigma(A[t, j])$ and the result follows from the IH.

   (b) If the β-redex put in head position is some $N(j)$ and $M(j) = (x \ a)$ and $\sigma(x) = \lambda y.b$ then $A[\sigma(t), i] = \lambda y.(\sigma(H) \ N(1) \ ... \ N(j-1) \ b) \ \sigma(a) \ N(j+1) \ ... \ N(n))$. Since $type(\sigma(a)) < size(A)$ it is enough, by the IH, to show that $\lambda y.(\sigma(H) \ N(1) \ ... \ N(j-1) \ b) \ y \ N(j+1) \ ... \ N(n))$ and so, by Lemma 3.3, that $(\sigma(H) \ N(1) \ ... \ N(j-1) \ b \ N(j+1) \ ... \ N(n)) \in SN$. Since $type(b) < size(A)$ it is enough, by the IH, to show $u = (\sigma(H) \ N(1) \ ... \ N(j-1) \ z \ N(j+1) \ ... \ N(n)) \in SN$. Let $t' = (H \ \overrightarrow{M'})$ where $\overrightarrow{M'}$ is defined by $M'(k) = M(k)$, for $k \neq j$, $M'(j) = z$. Since $t = t'[z := (x \ a)]$ and $u = \sigma(t')$ the result follows from Lemma 3.1 and the IH.

   (c) If, finally, H is a variable, $\sigma(H) = (H' \ \overrightarrow{M'})$ and the β-redex put in head position is some $M'(j)$. Then, $A[\sigma(t), j] = \tau(A[t', j])$ where t' is the same as t but where we have given to the variable H the fresh variable z and τ is the same as σ for the variables in $dom(\sigma)$ and $\tau(z) = A[\sigma(H), j]$. Note that that $t' \in SN$ and $\eta(t') \leq \eta(t)$, by Lemma 3.1. We conclude by the IH since $\eta(\tau, t') < \eta(\sigma, t)$.

□

Corollary 3.1 *Let t be a typable term. Then t is strongly normalizing.*

Proof By induction on $size(t)$. If t is an abstraction or a variable the result is trivial. Otherwise $t = (u \ v)$ and, by the IH, $u, v \in SN$. Thus, by Theorem 3.3, $(u \ y) = (x \ y)[x := u] \in SN$ and, by applying again Theorem 3.3, $(u \ v) = (u \ y)[y := v] \in SN$. □

References

[1] R. David. *Normalization without reducibility.* APAL 107 (2001) p 121-130.

[2] R. Dyckhoff and S. Lengrand. *Call-by-value λ-calculus and LJQ.* Journal of Logic and Computation, 17:1109-1134, 2007.

[3] F. Kamareddine. *Postponement, Conservation and Preservation of Strong Normalisation for Generalised Reduction* . Journal of Logic and Computation, volume 10 (5), pages 721-738, 2000

[4] A. J. Kfoury and J. B. Wells. *New notions of reduction and non-semantic proofs of beta -strong normalization in typed lambda -calculi.* In Proc. 10th Ann. IEEE Symp. Logic in Comput. Sci., pages 311-321, 1995.

[5] E. Moggi. *Computational lambda-calculus and monads.* LICS 1989.

[6] Y. Ohta and M. Hasegawa. *A terminating and confluent linear lambda calculus.* In Proc. 17th International Conference on Rewriting Techniques and Applications (RTA'06). Springer LNCS 4098, pages 166-180, 2006.

[7] L Regnier. *Une équivalence sur les lambda-termes*, in TCS 126(2) pp 281-292, (1994).

[8] J. Espírito Santo. *Delayed substitutions*, in Proceedings of RTA 2007, Lecture Notes in Computer Science, volume 4533, pp. 169-183, Springer, 2007,

[9] J. Espírito Santo. *Addenda to Delayed Substitutions*, Manuscript (available in his web page), July 2008.

[10] J. Espírito Santo. *A note on the preservation of strong normalisation in the λ-calculus*, Manuscript, September 2009.