Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 136 (2005) 103–116
www.elsevier.com/locate/entcs

Lazy Strong Normalization

Luca Paolini[1,2]

Dipartimento di Informatica
Università di Torino (ITALIA)

Elaine Pimentel[1,2]

Departamento de Matemática
Universidade Federal de Minas Gerais (BRASIL)
Dipartimento di Informatica
Università di Torino (ITALIA)

Simona Ronchi Della Rocca[1,2]

Dipartimento di Informatica
Università di Torino (ITALIA)

Abstract

Among all the reduction strategies for the untyped λ-calculus, the so called *lazy β-evaluation* is of particular interest due to its large applicability to functional programming languages (e.g. Haskell [3]). This strategy reduces only redexes not inside a lambda abstraction.
The lazy strongly β- normalizing terms are the λ-terms that don't have infinite lazy β-reduction sequences.
This paper presents a logical characterization of lazy strongly β-normalizing terms using intersection types. This characterization, besides being interesting by itself, allows an interesting connection between call-by-name and call-by-value λ-calculus.
In fact, it turns out that the class of lazy strongly β-normalizing terms coincides with that of call-by-value potentially valuable terms. This last class is of particular interest since it is a key notion for characterizing solvability in the call-by-value setting.

Keywords: λ-calculus, lazy evaluation, lazy strong normalization

[1] Email: paolini@di.unito.it, pimentel@di.unito.it, ronchi@di.unito.it
[2] Paper partially supported by IST-2001-33477 DART Project, MIUR-Cofin'02 PROTOCOLLO Project. Pimentel is also supported by CNPq.

doi:10.1016/j.entcs.2005.06.013

1 Introduction

An evaluation is called *lazy* if the body of a function is evaluated only when an argument is supplied. In the λ-calculus setting, this kind of evaluation is modelled by a reduction strategy that does not reduce β-redexes occurring under the scope of a λ-abstraction. Lazy evaluation has been introduced by Plotkin [6] in order to capture into λ-calculus the conceptual difference between the notion of evaluation and that one of code optimization. [3]

The notion of strong β-normalization can be extended to the lazy case in a natural way (see [8]). Namely: a lazy β-redex is a β-redex not occurring under the scope of a λ-abstraction, and a term is in lazy β-normal form if and only if it has no occurrences of lazy β-redexes. So a term is lazy strongly β-normalizing if and only if it has lazy β-normal form and there are not infinite lazy β-reduction sequences starting from it.

In this paper we give a complete characterization of the class of lazy strongly β-normalizing terms in a logical way, using a suitable intersection type assignment system.

This characterization, besides being interesting by itself, allows an interesting connection between call-by-name and call-by-value λ-calculus. Let us remember that the classical λ-calculus is a model for the call-by-name evaluation, while the call-by-value evaluation can be modelled by a variant of λ-calculus, the $\lambda\beta_v$-calculus, introduced in [6]. The $\lambda\beta_v$-calculus is obtained from the λ-calculus by restricting the β-rule to the case where the argument is a value, i.e., it is either a variable or a λ-abstraction. The fact that all the λ-abstractions are values, independently from their bodies, implies that the natural evaluation for such a calculus is a lazy one. Some syntactical properties of the $\lambda\beta_v$-calculus have been studied in [5], where the notion of solvability has been adapted to this calculus, and the set of solvable terms has been completely characterized, in a logical way.

In particular, in order to give such a characterization, an intermediate class of terms has been introduced: the potentially valuable terms. A term M is potentially valuable if and only if there is a substitution $\mathbf{s}$, replacing free variables by closed values, such that $\mathbf{s}(M)$ reduces to a value. The importance of such a class becomes clearer when we note that, in the $\lambda\beta_v$-calculus, the restriction to the β-rule imposes that every term (or subterm), in order to be manipulated, must be first transformed into a value. The potentially valuable terms have been completely characterized in a logical way in [5], and it has been proved that the call-by-value solvable terms form a proper subclass of

[3] This must not be confused with the notion of lazy evaluation used in functional programming corresponding to a *call-by-need* evaluation strategy.

the class of the potentially valuable terms.

It turns out that the class of potentially valuable terms coincides with the class of strongly β-normalizing terms. We think that this relationship is an interesting bridge between the call-by-name and the call-by-value evaluation.

Besides, the type assignment used in the present work for the characterization of lazy β-strong normalization, if enriched by a suitable subtyping relation, coincides with the one in [4], which induces a filter model for the call-by-value λ-calculus. This is a further semantic witness of the relationship between call-by-name and call-by-value evaluation.

2 Language

Definition 2.1 Let Var be a countable set of variables. The set Λ of λ-terms is defined by the following grammar:

$$M ::= x \mid MM \mid \lambda x.M$$

As usual, terms will be considered modulo α-conversion, i.e., modulo names of bound variables. α-conversion will be denoted by $\equiv$. We will use the syntactic conventions as in [2]. λ-terms will be ranged over by Latin capital letters.

The evaluation of a term is said *lazy* if no reduction is made under the scope of a λ-abstraction. It is possible to define directly the lazy reduction, as shown in the next definition.

Definition 2.2 i) The β-rule is defined as $(\lambda x.M)N \to M[N/x]$.

ii) The *β-reduction* is the contextual closure of the β-rule. We will denote by $\to_\beta$ the β-reduction, by $\to_\beta^*$ its reflexive and transitive closure, and by $=_\beta$ its symmetric, reflexive and transitive closure.

iii) The *lazy β-reduction* is the applicative closure of the β-rule. We will denote by $\to_{\beta\ell}$ the *lazy* β-reduction, by $\to_{\beta\ell}^*$ its reflexive and transitive closure, and by $=_{\beta\ell}$ its symmetric, reflexive and transitive closure.

iv) The *η-reduction* is defined as the contextual closure of the following rule:

$$\lambda x.Mx \to_\eta M$$

and $\to_\eta^*$ is its reflexive and transitive closure.

Notice that the definition of lazy β-reduction, at point iii), is not standard. In fact, the reduction is defined by closing the reduction rule only under application, while in the standard case the closure is under abstraction too.

The notion of normal form can be adapted for the lazy β-reduction in the following way.

Definition 2.3 i) A term M *is in lazy β-normal form* if and only if it has no occurrences of β-redexes, but under the scope of a λ-abstraction.

ii) A term M *has lazy β-normal form* if and only if there is a term N in lazy β-normal form such that $M \to^*_{\beta\ell} N$.

Clearly β-normal forms are lazy β-normal form.

Note that the lazy β-normal form of a term, if there exists, may not be unique. In fact, $(\lambda xy.x)(II) \to^*_{\beta\ell} \lambda y.II$ and $(\lambda xy.x)(II) \to^*_{\beta\ell} \lambda y.I$ where both $\lambda y.II$ and $\lambda y.I$ are lazy β-normal forms.

Now we can define the key notion of $\beta\ell$-strong normalization.

Definition 2.4 A term M is *$\beta\ell$-strongly normalizing* if and only if it has lazy β-normal form, and moreover there is not an infinite sequence of lazy β-reductions starting from it.

3 An intersection type assignment system

Definition 3.1 i) Let C be a countable set of *type-constants* (ranging over $\alpha, \beta, ..$) containing at least the type constant ν.
The set $T(\mathsf{C})$ of *types*, ranging over by $\sigma, \tau, \pi, \rho, ..$ is inductively defined as follows:

$$\begin{array}{lcl} \sigma \in \mathsf{C} & \Rightarrow & \sigma \in T(\mathsf{C}) \\ \sigma, \tau \in T(\mathsf{C}) & \Rightarrow & (\sigma \to \tau) \in T(\mathsf{C}) \\ \sigma, \tau \in T(\mathsf{C}) & \Rightarrow & (\sigma \wedge \tau) \in T(\mathsf{C}). \end{array}$$

Types will be considered modulo associativity, commutativity and idempotency of the constructor $\wedge$ (i.e., modulo an equivalence $\simeq$ which is the contextual, reflexive and transitive closure of the following rules: $\sigma \wedge \tau \simeq \tau \wedge \sigma$, $\sigma \simeq \sigma \wedge \sigma$ and $(\sigma \wedge \tau) \wedge \pi \simeq \sigma \wedge (\tau \wedge \pi)$). We use the convention that the constructor $\wedge$ take precedence over $\to$.

ii) A basis is a partial function from Var to $T(\mathsf{C})$ having a finite domain of definition. If B is a basis then $B[\sigma/x]$ denotes the basis such that

$$B[\sigma/x](y) = \begin{cases} \sigma & \text{if } \ y \equiv x, \\ B(y) & \text{otherwise.} \end{cases}$$

Furthermore, the basis B such that $dom(B) = \{x_1, ..., x_n\}$ and $B(x_i) = \sigma_i$, for $1 \leq i \leq n$ will be denoted by $[\sigma_1/x_1, ..., \sigma_n/x_n]$.

iii) The *type assignment system* $\vdash_\nu$ is a formal system proving typing judgments of the shape:

$$B \vdash_\nu M : \sigma$$

where M is a term, $\sigma \in T(\mathsf{C})$ and B is a basis.

The type assignment system $\vdash_\nu$ consists of the following rules:

$$\frac{}{B[\sigma/x] \vdash_\nu x : \sigma}\ (var) \qquad \frac{}{B \vdash_\nu \lambda x.M : \nu}\ (\nu)$$

$$\frac{B[\sigma/x] \vdash_\nu M : \tau}{B \vdash_\nu \lambda x.M : \sigma \to \tau}\ (\to I) \qquad \frac{B \vdash_\nu M : \sigma \to \tau \quad B \vdash_\nu N : \sigma}{B \vdash_\nu MN : \tau}\ (\to E)$$

$$\frac{B \vdash_\nu M : \sigma \quad B \vdash_\nu M : \tau}{B \vdash_\nu M : \sigma \wedge \tau}\ (\wedge I)$$

$$\frac{B \vdash_\nu M : \sigma \wedge \tau}{B \vdash_\nu M : \sigma}\ (\wedge E_l) \qquad \frac{B \vdash_\nu M : \sigma \wedge \tau}{B \vdash_\nu M : \tau}\ (\wedge E_r)$$

If B, B' are bases then $B \cap B'$ is the basis defined as follows:

$$(B \cap B')(y) = \begin{cases} B(y) \wedge B'(y) & \text{if both } B(y) \text{ and } B'(y) \text{ are defined,} \\ B(y) & \text{if } B(y) \text{ is defined and } B'(y) \text{ is undefined,} \\ B'(y) & \text{if } B'(y) \text{ is defined and } B(y) \text{ is undefined,} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The type assignment system $\vdash_\nu$ enjoys some interesting properties.

Lemma 3.2 (Generation)

i) *If $B \vdash_\nu M : \sigma$ then $B \cap B' \vdash_\nu M : \sigma$, for any basis B'.*

ii) *If $B \vdash_\nu MN : \sigma$ then there are types ρ_i and τ_i with $1 \leq i \leq n$, such that $\sigma \simeq \rho_1 \wedge \ldots \wedge \rho_n$, $B \vdash_\nu M : \tau_i \to \rho_i$ and $B \vdash_\nu N : \tau_i$.*

iii) *$B \vdash_\nu \lambda x.M : \sigma \to \tau$ if and only if $B[\sigma/x] \vdash_\nu M : \tau$.*

Proof

i) Easy, by induction on the derivation d proving $B \vdash_\nu M : \sigma$.

ii) Easy, b induction on the derivation d proving $B \vdash_\nu MN : \sigma$.

iii) ($\Leftarrow$) By rule $(\to I)$.
($\Rightarrow$) It is easy to prove that $B \vdash_\nu \lambda x.M : \sigma \to \tau \wedge \pi_1 \wedge ... \wedge \pi_n$ $(n \in \mathbb{N})$ implies $B[\sigma/x] \vdash_\nu M : \tau$, by induction on derivations. □

The type system $\vdash_\nu$ enjoys the subject-reduction property and a restricted form of subject-expansion.

Property 3.3 (Subject-reduction)
If $B \vdash_\nu M : \sigma$ and $M \to_\beta N$ then $B \vdash_\nu N : \sigma$.

Proof Standard. □

Property 3.4 (Typed subject-expansion)
Let $C[.]$ be a context. Then $B \vdash_\nu C[P[Q/x]] : \sigma$ and $B' \vdash_\nu Q : \tau$ imply $B \cap B' \vdash_\nu C[(\lambda x.P)Q] : \sigma$.

Proof The proof is by induction on $C[.]$. Let d be a derivation proving $B \vdash_\nu C[P[Q/x]] : \sigma$. We may assume, without loss of generality, that B is undefined on x and that all typings in d have the same basis B. Indeed, $(\to I)$ is the only rule having a basis, in the premises, different from the basis in the conclusion; but we can assume that free and bound variables have different names in M.
In case $C[.] = [.]$, there are two cases to analyze.

a) Suppose that either $x \notin \mathrm{FV}(P)$ (hence $P[Q/x] \equiv P$) or Q occurs in subterms of P which are subjects of an application of the rule (ν).
In both cases, $B \vdash_\nu P : \sigma$; therefore $B[\tau/x] \vdash_\nu P : \sigma$, by Lemma 3.2.i). Then $B \vdash_\nu \lambda x.P : \tau \to \sigma$, by rule $(\to I)$ and, by Lemma 3.2.i), both $B \cap B' \vdash_\nu \lambda x.P : \tau \to \sigma$ and $B \cap B' \vdash_\nu Q : \tau$. Hence, by rule $(\to E)$,

$$B \cap B' \vdash_\nu (\lambda x.P)Q : \sigma.$$

b) Suppose that Q occurs in $P[Q/x]$ and there is a subderivation of d having Q as subject of the typing of its conclusion. The derivation d can be transformed into a derivation d' proving $B[\tau/x] \vdash_\nu P : \sigma$ by performing the following operations.
- Replace each typing $B \vdash_\nu Q : \tau$ occurring in the derivation d by:

$$\frac{}{B[\tau/x] \vdash_\nu x : \tau}\ (var).$$

- Replace each typing $B \vdash_\nu P'[Q/x] : \mu$ occurring in the derivation d by the typing $B[\tau/x] \vdash_\nu P' : \mu$.

It is easy to see by induction on d that d' is well defined. Thus the proof proceeds as in case (a).

For the general case, where $C[.] = \lambda x.C'[.]$ or $C[.] = C_1[.]C_2[.]$, the result follows easily by induction. □

Note that *typing* in the type assignment system $\vdash_\nu$ is not preserved by η-expansion if the set C has any type constants other than ν. Besides, $\vdash_\nu$ is not preserved by η-reduction. In fact,

$$\emptyset \vdash_\nu \lambda y.xy : \nu$$

while $x : \nu$ is not provable from the empty context.

Moreover, the η-reduction is not valid even in the case that we consider only terms having a functional type, as shown in the next example.

Example 3.5 Let $\pi = (\sigma \to \tau_0) \wedge (\sigma \to \tau_1)$ and $B = [\pi/x, \sigma/y]$. Then $[\pi/x] \vdash_\nu \lambda y.xy : \sigma \to \tau_0 \wedge \tau_1$ since:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{B \vdash_\nu x : \pi}\,(var)}{B \vdash_\nu x : \sigma \to \tau_0}\,(\wedge E_l) \quad \dfrac{}{B \vdash_\nu y : \sigma}\,(var)}{B \vdash_\nu xy : \tau_0}\,(\to E) \quad \dfrac{\dfrac{\dfrac{}{B \vdash_\nu x : \pi}\,(var)}{B \vdash_\nu x : \sigma \to \tau_1}\,(\wedge E_r) \quad \dfrac{}{B \vdash_\nu y : \sigma}\,(var)}{B \vdash_\nu xy : \tau_1}\,(\to E)}{B \vdash_\nu xy : \tau_0 \wedge \tau_1}\,(\wedge I)}{[\pi/x] \vdash_\nu \lambda y.xy : \sigma \to \tau_0 \wedge \tau_1}\,(\to I)$$

But it is easy to check that there isn't a derivation proving $[\pi/x] \vdash_\nu x : \sigma \to \tau_0 \wedge \tau_1$.

It occurs that the standard proofs of the strong normalization property usually depend on the fact that the considered system enjoys a restricted form of η-reduction, namely that the η-reduction holds in the case of arrow types.

A similar situation can be found in, for example, Pottinger [7], that solved the problem by adding to the type system an explicit η-rule. We use a different technical approach to this problem noting that, although *typings* are not preserved by η-reduction, *typability* is preserved.

Lemma 3.6 *Let $B \vdash_\nu M : \sigma$ and $x \notin FV(M)$.*
If $M \to_\eta^* P[Q/x]\overrightarrow{Q}$ and $B' \vdash_\nu Q : \rho$ then there is a term M' such that $M' \to_\eta^* (\lambda x.P)Q\overrightarrow{Q}$ and $B \cap B' \vdash_\nu M' : \sigma$.

Proof The proof is by induction on the number m of η-reductions with subordinate induction on the derivation d proving $B \vdash_\nu M : \sigma$. If $m = 0$ then the result follows from Property 3.4.
Let $m \geq 1$. There are three cases to analyze.

a) $M \equiv P'[Q'/x]\overrightarrow{Q'}$ with $P' \to_\eta^* P$, $Q' \to_\eta^* Q$ and $\overrightarrow{Q'} \to_\eta^* \overrightarrow{Q}$. Then the result follows from Property 3.4.

b) $M \equiv (\lambda u.\bar{M}u)\overrightarrow{Q'}$ where $\overrightarrow{Q'}$ is a sequence of $n > 0$ terms and $\bar{M}\overrightarrow{Q'} \to_\eta^* P[Q/x]\overrightarrow{Q}$. Then,

$$M \to_{\beta\ell} \bar{M}\overrightarrow{Q'} \to_\eta^* P[Q/x]\overrightarrow{Q}$$

hence $B \vdash_\nu \bar{M}\overrightarrow{Q'} : \sigma$ by Property 3.3. The proof follows by induction.

c) $M \equiv \lambda x_1.\bar{M}x_1$ with $\lambda x_1.\bar{M}x_1 \to_\eta \bar{M} \to_\eta^* P[Q/x]\overrightarrow{Q}$.
Then the last rule applied in d can only be: $(\to I)$, $(\wedge I)$, $(\wedge E_l)$, $(\wedge E_r)$ or (ν). The only not trivial case is the first one. So suppose that $\sigma \simeq \pi \to \mu$ and d ends with

$$\frac{B[\pi/x_1] \vdash_\nu \bar{M}x_1 : \mu}{B \vdash_\nu \lambda x_1.\bar{M}x_1 : \pi \to \mu}\ (\to I)$$

Clearly $Mx_1 \to_\eta^* P[Q/x]\overrightarrow{Q}x_1$, thus by inductive hypothesis on d there exists a term M'' such that $M'' \to_\eta^* (\lambda x.P)Q\overrightarrow{Q}x_1$ and $B[\pi/x_1] \cap B' \vdash_\nu M'' : \mu$.
Without loss of generality, we may assume that $B'(x_1)$ is undefined; so $B[\pi/x_1] \cap B' = (B \cap B')[\pi/x_1]$ and $(B \cap B')[\pi/x_1] \vdash_\nu M'' : \mu$.
Therefore $B \cap B' \vdash_\nu \lambda x_1.M'' : \pi \to \mu$, by rule $(\to I)$. The proof is done, since

$$\lambda x_1.M'' \to_\eta^* (\lambda x.P)Q\overrightarrow{Q}.$$

□

Consider the type assignment system obtained from $\vdash_\nu$ by erasing the rule (ν): it is well known that it characterizes the β-strongly normalizing terms (see [7]). We will prove that the whole system $\vdash_\nu$ characterizes the $\beta\ell$-strong normalizing terms.

Theorem 3.7 *There are B, σ such that $B \vdash_\nu M : \sigma$ if and only if M is $\beta\ell$-strongly normalizing.*

Proof The proof is given in Subsections 3.1 and 3.2. □

3.1 Typability in $\vdash_\nu$ implies $\beta\ell$-strong normalization

Let $\mathcal{S}(B, \sigma, M)$ be the following predicate: "M is $\beta\ell$-strongly normalizing and there exists a λ-term M' such that $M' \to_\eta^* M$ and $B \vdash_\nu M' : \sigma$".

The following property holds.

Property 3.8 *$\mathcal{S}(B, \sigma \to \tau, x\overrightarrow{M})$ and $S(B', \sigma, N)$ imply $\mathcal{S}(B \cap B', \tau, x\overrightarrow{M}N)$.*

Proof Trivial, since $\overrightarrow{M}$ and N are independent. □

The predicate $\mathcal{S}$ is used to define a computability predicate.

Definition 3.9
The predicate $Comp$ is defined by induction on types as follows:

- $Comp(B, \alpha, M)$ if and only if $\mathcal{S}(B, \alpha, M)$, for all $\alpha \in \mathsf{C}$;
- $Comp(B, \sigma \to \tau, M)$ if and only if, for all $N \in \Lambda$, $B' \vdash_\nu N : \sigma$ and $Comp(B', \sigma, N)$ imply $Comp(B \cap B', \tau, MN)$;
- $Comp(B, \sigma \wedge \tau, M)$ if and only if $Comp(B, \sigma, M)$ and $Comp(B, \tau, M)$.

$Comp$ is closed under $\beta\ell$-reduction and under a restricted form of $\beta\ell$-expansion.

Property 3.10 *Let Q be $\beta\ell$-strongly normalizing.*
If $Comp(B, \sigma, P[Q/x]\overrightarrow{Q})$ and $B' \vdash_\nu Q : \mu$ then $Comp(B' \cap B, \sigma, (\lambda x.P)Q\overrightarrow{Q})$.

Proof The proof is given by induction on the structure of types.
Assume $\sigma \in \mathsf{C}$. Then, by definition, $Comp(B, \sigma, P[Q/x]\overrightarrow{Q})$ implies that there exists a term $M \to_\eta^* P[Q/x]\overrightarrow{Q}$ such that $B \vdash_\nu M : \sigma$ and $P[Q/x]\overrightarrow{Q}$ is $\beta\ell$-strongly normalizing. As $(\lambda x.P)Q\overrightarrow{Q} =_{\beta\ell} P[Q/x]\overrightarrow{Q}$ and Q is $\beta\ell$-strongly normalizing, we have that $(\lambda x.P)Q\overrightarrow{Q}$ is also $\beta\ell$-strongly normalizing and by Lemma 3.6 there exists $M' \to_\eta^* (\lambda x.P)Q\overrightarrow{Q}$ such that $B \vdash_\nu M' : \sigma$. Hence, $Comp(B, \sigma, (\lambda x.P)Q\overrightarrow{Q})$ by definition.
Let $\sigma \simeq \tau \to \rho$. Then $Comp(B, \tau \to \rho, P[Q/x]\overrightarrow{Q})$ implies that $\forall N$ such that $B' \vdash_\nu N : \tau$ and $Comp(B', \tau, N)$ we have $Comp(B \cap B', \rho, P[Q/x]\overrightarrow{Q}N)$.
Hence $Comp(B \cap B', \rho, (\lambda x.P)Q\overrightarrow{Q}N)$ by induction, and therefore by definition $Comp(B \cap B', \tau \to \rho, (\lambda x.P)Q\overrightarrow{Q})$.
The case $\sigma \simeq \tau \wedge \rho$ is trivial, by induction. □

We prove that $B \vdash_\nu M : \sigma$ implies $Comp(B, \sigma, M)$, which in turn implies $\mathcal{S}(B, \sigma, M)$.

Lemma 3.11 i) *$\mathcal{S}(B, \sigma, x\overrightarrow{M})$ implies $Comp(B, \sigma, x\overrightarrow{M})$.*
ii) *$Comp(B, \sigma, M)$ implies $\mathcal{S}(B, \sigma, M)$.*

Proof The proof is done by mutual induction on σ.
The only not obvious case is when $\sigma \simeq \tau \to \rho$

i) We will prove that $Comp(B', \tau, N)$ and $B' \vdash_\nu N : \tau$ imply $Comp(B \cap B', \rho, x\overrightarrow{M}N)$, thus $Comp(B, \tau \to \rho, x\overrightarrow{M})$ follows by definition.
$Comp(B', \tau, N)$ implies $\mathcal{S}(B', \tau, N)$, by induction on ii).

By hypothesis $\mathcal{S}(B, \tau \to \rho, x\overrightarrow{M})$; thus $\mathcal{S}(B \cap B', \rho, x\overrightarrow{M}N)$ by Property 3.8. The proof follows, since by induction $Comp(B \cap B', \rho, x\overrightarrow{M}N)$.

ii) Let $z \notin \mathrm{FV}(M)$, and let B be such that $B(z)$ is undefined. Note that, for any x, $[\tau/x] \vdash_\nu x : \tau$; so in particular, $\mathcal{S}([\tau/z], \tau, z)$. Hence $Comp([\tau/z], \tau, z)$ by induction on i). Thus $Comp(B[\tau/z], \rho, Mz)$ by definition of $Comp$ and this implies $\mathcal{S}(B[\tau/z], \rho, Mz)$, by induction. That is, Mz is $\beta\ell$-strongly normalizing and clearly also M is $\beta\ell$-strongly normalizing. Moreover, there exists a term M' such that both $M' \to_\eta^* Mz$ and $B[\tau/z] \vdash_\nu M' : \rho$. Hence $B \vdash_\nu \lambda z.M' : \tau \to \rho$ and since

$$\lambda z.M' \to_\eta^* \lambda z.Mz \to_\eta M$$

$\mathcal{S}(B, \tau \to \rho, M)$ follows by definition.

□

Lemma 3.12 *Let $FV(M) \subseteq \{x_1, ..., x_n\}$ and $B = B^*[\sigma_1/x_1, ..., \sigma_n/x_n]$. If $Comp(B_i, \sigma_i, N_i)$, $B_i \vdash_\nu N_i : \sigma_i$ $(1 \le i \le n)$ and $B \vdash M : \tau$, then*

$$Comp(B^* \cap B_1 \cap ... \cap B_n, \tau, M[N_1/x_1, ..., N_n/x_n]).$$

Proof By induction on the derivation d of $B \vdash M : \tau$. The most interesting case is when the last rule applied of d is $(\to I)$. Let $M \equiv \lambda x.M'$, $\tau \simeq \mu \to \rho$ and

$$\frac{B[\mu/x] \vdash M' : \rho}{B \vdash \lambda x.M' : \mu \to \rho}\ (\to I)$$

Let $Comp(B', \mu, N)$ and $B' \vdash_\nu N : \mu$. So $\mathcal{S}(B', \mu, N)$ by Lemma 3.11.ii); hence N is $\beta\ell$-strong normalizing. By induction

$$Comp(B^* \cap B' \cap B_1 \cap ... \cap B_n, \rho, M'[N_1/x_1, ..., N_n/x_n, N/x])$$

which implies $Comp(B^* \cap B' \cap B_1 \cap ... \cap B_n, \rho, (\lambda x.M'[N_1/x_1, ..., N_n/x_n])N)$ by Property 3.10. Hence, $Comp(B^* \cap B_1 \cap ... \cap B_n, \mu \to \rho, M[N_1/x_1, ..., N_n/x_n])$ by definition of $Comp$. All other cases follow directly from the inductive hypothesis. □

Proof of Theorem 3.7 ($\Rightarrow$ part).
Suppose that $B \vdash_\nu M : \sigma$ and let $\mathrm{FV}(M) \subseteq \{x_1, ..., x_n\}$, and $B(x_i) = \sigma_i$. Since $Comp(B, \sigma_i, x_i)$ $(1 \le i \le n)$ by Lemma 3.11.i), then $Comp(B, \sigma, M)$ by Lemma 3.12. Hence the proof is done due to Lemma 3.11.ii). □

3.2 $\beta\ell$-strong normalization implies typability in $\vdash_\nu$

Let M be in lazy β-normal form. It is easy to see that either $M \equiv \lambda x.M'$ or $M \equiv xM_1 \dots M_n$ with $n \geq 0$ where M_i are in lazy β-normal form for all $1 \leq i \leq n$.

Lemma 3.13 *If M is in lazy β-normal form, then there are a basis B and a type $\sigma \in T(\mathsf{C})$ such that $B \vdash_\nu M : \sigma$.*

Proof Let M be in lazy β-normal form. The proof is done by induction on the shape of M. If $M \equiv \lambda x.M'$ then $B \vdash_\nu M : \nu$ for any basis B.
Let $M \equiv xM_1 \dots M_n$. If $n = 0$, then M is a variable and

$$\frac{}{x : \sigma \vdash_\nu x : \sigma}\ (var)$$

for any $\sigma \in T(\mathsf{C})$. Suppose $n > 0$. By inductive hypothesis there are $B_1, \dots, B_n$ and $\sigma_1, \dots, \sigma_n$ such that:

$$B_i \vdash_\nu M_i : \sigma_i$$

Then M has type σ in the basis $B' = B_1 \cap \dots \cap B_n \cap [\sigma_1 \to \dots \to \sigma_n \to \sigma/x]$ since:

$$\frac{\dfrac{}{B' \vdash_\nu x : \sigma_1 \to \dots \to \sigma_n \to \sigma}\ (\wedge E_r)/(var) \qquad B' \vdash_\nu M_i : \sigma_i}{B' \vdash_\nu xM_1 \dots M_n : \sigma}\ (\to E)$$

□

Remember that a lazy β-redex is a β-redex that does not occur under the scope of a λ-abstraction.

Property 3.14 *Let M be not in lazy β-normal form.*
Then there are subterms P, Q of M such that Q in lazy β-normal form and $(\lambda x.P)Q$ is a lazy β-redex of M.

Proof The proof is by induction on M. □

Proof of Theorem 3.7 ($\Leftarrow$ part).
Suppose that M is $\beta\ell$-strong normalizing, that is, there is not an infinite sequence of $\beta\ell$-reductions starting from M.
Without loss of generality, by Property 3.14, we can assume that there is a lazy β-reduction sequence

$$M \equiv M_0 \to_{\beta\ell} \dots \to_{\beta\ell} M_n \equiv N$$

reducing only lazy β-redexes of the shape $(\lambda x.P)Q$ such that Q in lazy β-normal form.

The proof is given by induction on n.
If $n = 0$, the result follows from Lemma 3.13. Suppose $n \geq 1$. By induction hypothesis, there are a base B_1 and a type σ such that $B_1 \vdash_\nu M_1 : \sigma$. Moreover, there is a basis B_2 and a type τ such $B_2 \vdash_\nu Q : \tau$ by Lemma 3.13. Then the result follows trivially from Property 3.4. □

4 $\beta\ell$-strong normalization and call-by-value solvability

The notion of $\beta\ell$-strong normalization allows for stating an interesting relationship between call-by-name and call-by-value evaluation of λ-calculus.

Let us recall the definition of call-by-value λ-calculus [6].

Definition 4.1 Let the set Val of *values* be $\mathrm{Var} \cup \{\lambda x.M \mid M \in \Lambda\}$.

i) The β_v*-reduction* ($\to_{\beta v}$) is the contextual closure of the following rule:

$$(\lambda x.M)N \to M[N/x] \quad \text{if and only if} \quad N \in \mathsf{Val}.$$

ii) $\to^*_{\beta_v}$ and $=_{\beta_v}$ are respectively the reflexive and transitive closure of $\to_{\beta_v}$ and the symmetric, reflexive and transitive closure of $\to_{\beta_v}$.

iii) The $\lambda\beta_v$-calculus is the language Λ equipped with the β_v-reduction

Plotkin proved that the $\lambda\beta_v$-calculus is confluent. The notion of solvability can be extended to the $\lambda\beta_v$-calculus in the following way.

Definition 4.2 A term M is β_v-solvable if and only if there is a sequence $\overrightarrow{P}$ of values such that:

$$(\lambda x_1 ... x_n.M)\overrightarrow{P} =_{\beta_v} I$$

where $\mathrm{FV}(M) = \{x_1, ...x_n\}$ and $I = \lambda x.x$ is the identity term.

The main problem on reasoning in an operational way in the $\lambda\beta_v$-calculus has to do with the fact that every term (or subterm) must be transformed into a value in order to be manipulated. In fact, in [5], in order to prove syntactical properties of the $\lambda\beta_v$-calculus, it was introduced the key notion of potential valuability.

Definition 4.3 i) A term M is *valuable* if and only if it β_v-reduces to a term belonging to Val.

ii) A term M is *potentially valuable* if and only if there is a substitution $\mathbf{s}$, replacing variables by closed terms belonging to Val, such that $\mathbf{s}(M)$ is valuable.

In [5] it was proved that the set of β_v-solvable terms is a proper subset of the set of potentially valuable terms. Moreover, a logical characterization of both the potentially valuable and the β_v-solvable terms is given, through an intersection type assignment system which is equivalent (with respect to typability power) to the system $\vdash_\nu$. More precisely, the system in [5] is obtained from $\vdash_\nu$ by restricting the set of types, allowing the use of the intersection only in the left side of an arrow. It is well known that two intersection type assignment systems related to each other by this relation have the same typability power (see for example [1]).

In order to show this characterization, we need to introduce a definition.

Definition 4.4 A type σ is *proper* if it is of the following shape:

$$\sigma \simeq \tau_1 \to \ldots \to \tau_n \to \alpha$$

where $n \geq 0$ and α is a type constant different from ν.

The following theorem holds.

Theorem 4.5 *[5]*

i) *M is potentially valuable if and only if there are B, σ such that $B \vdash_\nu M : \sigma$.*

ii) *M is β_v-solvable if and only if there B, σ such that σ is proper and $B \vdash_\nu M : \sigma$.*

On the basis of this result, and of the Theorem 3.7, we can state the following relation between call-by-name and call-by-value λ-calculi.

Corollary 4.6 *M is $\beta\ell$-strongly normalizing if and only if M is potentially valuable.*

References

[1] van Bakel, S., "Intersection type assignment systems," Theoretical Computer Science, 38(2):246-269, Elsevier, (1997).

[2] Barendregt, H.P., "The Lambda Calculus: its syntax and semantics," N.103 in Studies in Logic and the Foundations of Mathematics (revised edition), North-Holland, Amsterdam (1994).

[3] Bird, R., "Introduction to Functional Programming using Haskell," Series in Computer Science (2nd edition), Prentice Hall, (1998)

[4] Egidi, L., Honsell, F., and Ronchi della Rocca, S., "Operational, denotational and logical descriptions: a case study," Fundamenta Informaticae, 16(2) (1992) 149–170.

[5] Paolini, L., Ronchi Della Rocca, S., "Call-by-value Solvability," Theoretical Informatics and Applications, 33(6) (1999) 507-534.

[6] Plotkin G.D., "Call-by-name, call-by-value and the λ-calculus," Theoretical Computer Science, 1 (1975) 125-159.

[7] G. Pottinger, "A type assignment for the strongly normalizable λ-terms", in To H.B. Curry: essays on combinatory logic, lambda calculus and formalism, pp.561-577, Academic Press, London, 1980.

[8] Ronchi Della Rocca S., Paolini L., "The Parametric λ-calculus: a meta-model for computation", Computer Science-Monograph, Springer Verlag, to appear.