INFORMATIQUE THÉORIQUE ET APPLICATIONS

LUCA PAOLINI
SIMONA RONCHI DELLA ROCCA

Call-by-value solvability

Informatique théorique et applications, tome 33, n° 6 (1999), p. 507-534

<http://www.numdam.org/item?id=ITA_1999__33_6_507_0>

Theoretical Informatics and Applications
Theoret. Informatics Appl. **33** (1999) 507–534

CALL-BY-VALUE SOLVABILITY

LUCA PAOLINI[1] AND SIMONA RONCHI DELLA ROCCA[2]

Abstract. The notion of solvability in the call-by-value λ-calculus is defined and completely characterized, both from an operational and a logical point of view. The operational characterization is given through a reduction machine, performing the classical β-reduction, according to an innermost strategy. In fact, it turns out that the call-by-value reduction rule is too weak for capturing the solvability property of terms. The logical characterization is given through an intersection type assignment system, assigning types of a given shape to all and only the call-by-value solvable terms.

AMS Subject Classification. 68Q05, 03D10.

1. INTRODUCTION

The call-by-value λ-calculus ($\lambda\beta_v$-calculus) is a paradigmatic language which captures two features present in many real functional programming languages: the call-by-value parameter passing and the lazy evaluation. The parameters are passed in a call-by-value way, when they are evaluated before being passed and a function is evaluated in a lazy way when its body is evaluated only when parameters are supplied. The real programming languages are all lazy, and almost all call-by-value (*e.g.* ML [9], Scheme [13], while Haskell [14] is one of the few examples of a language using the call-by-name evaluation). Note that the call-by-value parameter passing cannot be modelled in the classical λ-calculus, since the β-reduction rule is intrinsecally a call-by-name rule. The $\lambda\beta_v$-calculus is a restriction of the classical λ-calculus based on the notion of *value*. Values are either variables or abstractions and they represent the already evaluated terms. Since the evaluation is lazy, an abstraction is always a value, independently from

Keywords and phrases: Models of computations, λ-calculus, call-by-value.

[1] DISI, Università di Genova, Dipartimento di Informatica e Scienze dell'Informazione, Via Dodecaneso 35, 16146 Genova, Italy; e-mail: paolini@disi.unige.it

[2] Università di Torino, Dipartimento di Informatica, C.so Svizzera 185, 10149 Torino, Italy; e-mail: ronchi@di.unito.it

its body. The call-by-value evaluation mechanism in the $\lambda\beta_v$-calculus is realized by defining a suitable reduction rule (the β_v-rule), which is a restriction of the classical β-rule, in the sense that $(\lambda x.M)N$ reduces to $M[N/x]$ if and only if N is a value, *i.e.*, it has been already evaluated.

The $\lambda\beta_v$-calculus and the machine for its evaluation, that we call secd, has been introduced by Plotkin [11] inspired by the seminal work of Landin [8] on the language ISWIM and the SECD machine.

In this paper we are dealing with the pure (*i.e.* without constants) version of the $\lambda\beta_v$-calculus. So a closed term is said *valuable* if its evaluation, through the secd machine, stops.

The notion of terminating programs and so of valuable terms is central for studying the operational equivalence between terms induced by the secd machine. Let a *context* $C[\ ]$ be a term with some occurrences of an hole, and let $C[M]$ be the term obtained from it once the holes have been filled by the term M. The operational equivalence is defined as follows:

$M \approx_s N$ if and only if $\forall C[\ ]$ such that $C[M]$ and $C[N]$ are closed, the secd machine stops on $C[M]$ if and only if it stops on $C[N]$.

This equivalence corresponds to the Liebnitz equality on programs. In fact a context $C[\ ]$ can be viewed as a *partially specified program*, and $C[M]$ as a program using M as subprogram. So two terms are equivalent if and only if they can be replaced each other in the same program without changing its observational behaviour. In a language (like the $\lambda\beta_v$-calculus) without constants, the natural behaviour to be observed is the termination property.

Plotkin proved that the $\lambda\beta_v$-calculus enjoys some of the good properties we expected from a calculus, namely the Church-Rosser and the standardization property. But the notion of solvability, in the call-by-value setting, has never been explored. In this paper we want to study such a notion.

The notion of solvability has been introduced in the classical λ-calculus for characterizing terms with *good* operational behaviour. Using a programming paradigm, M is solvable if and only if, for every output value P, there is a program $C_P[M]$, using effectively M as subprogram, such that $C_P[M]$ evaluates to P. The fact that $C_P[M]$ uses effectively M can be formalized as: not for all Q, $C_P[Q]$ evaluates to P.

In the case of classical λ-calculus, it has been proved [15] that, for all term M, if such a context $C[\ ]$ exists, then there is also a *head context*, *i.e.*, a context of the shape:

$$(\lambda x_1 \ldots x_n.[\ ])M_1 \ldots M_m$$

(for some m, n) with the same behaviour, where $\{x_1, ..., x_n\}$ is the set of free variables of M ($FV(M)$). This means that the λ-terms have a functional behaviour, and so the notion of solvability can be defined in the following standard way:

A term M is solvable if and only if there is a finite sequence of closed terms, $N_1,\ldots,N_m$ such that

$$(\lambda x_1\ldots x_n.M)N_1\ldots N_m =_\beta I$$

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

Solvable terms have been completely characterized from a syntactical point of view. A closed term M is solvable if and only if it has a head normal form (*i.e.* there are integers n,m and terms $M_1,\ldots,M_m$ such that $M =_\beta \lambda x_1\ldots x_n.x_iM_1\ldots M_m$ $(1\leq i\leq n)$, for some n).

From an operational point of view, solvable terms are the terminating programs, in the head reduction machine [12]. From a semantic point of view, all unsolvable terms (*i.e.*, the non terminating programs) can be all consistently equated [15]. From a logical point of view, a term M is solvable if and only if it can be typed in the intersection type assignment system defined by Coppo and Dezani [3].

Let recall also the notion of solvability in the lazy λ-calculus, introduced by Abramsky and Ong [1] for modelling the call-by-name lazy evaluation. The lazy λ-calculus is the classical one, equipped with the β-reduction rule, but, in the evaluation of terms, no reduction is made under the scope of an abstraction. Abramsky and Ong in [1] noted that the notion of solvability in this setting is the same as in the call-by-name case (a term is solvable if and only if it has a head normal form). But in this case the set of solvable terms does not coincide anymore with the set of terminating terms, with respect to the lazy evaluation. Indeed the term $\lambda x.\Delta\Delta$, where $\Delta\equiv\lambda x.xx$, is unsolvable, but the lazy evaluation stops on it. In order to clarify the relation between solvable terms and termination in the lazy setting, let us recall the notion of unsolvable of order n $(n\geq 0)$.

Let P be unsolvable. P is of order 0 if and only if there is no Q such that $P =_\beta \lambda x.Q$; P is of order n if and only if n is the maximum integer such that $P =_\beta \lambda x_1\ldots x_n.Q$; P is of infinite order if such a n does not exist. So the terminating terms in the lazy λ-calculus are the solvable terms plus the unsolvable ones of order greater than 0.

Semantically the unsolvable terms of order 0 (*i.e.*, the non terminating programs) can be consistently equated, but a model equating all unsolvable terms is not correct with respect to the lazy operational semantics.

As far as the logical characterization of lazy solvability is concerned, it is easy to show that the logical system defined in [1] can give such a characterization.

Now let us consider the call-by-value λ-calculus.

First of all we must ask ourselves how the general notion of solvability can be specialized in this setting. In [4] (Th. 33) it has been proved that the $\lambda\beta_v$-calculus has a functional behaviour, as the classical λ-calculus. More precisely, the operational behaviour of a term M can be studied by considering just the (call-by-value) head-contexts, *i.e.* contexts of the shape:

$$(\lambda x_1\ldots x_n.[\ ])P_1\ldots P_m$$

(for some m, n) where all P_i $(1 \leq i \leq m)$ are values and $FV(M) = \{x_1, ..., x_n\}$.

So we can define:

> a term M is call-by-value solvable *(v-solvable)* if and only if there is a finite sequence of closed values $N_1, \ldots, N_m$ such that
>
> $$(\lambda x_1 \ldots x_n.M)N_1 \ldots N_m =_v I$$
>
> where $FV(M) = \{x_1, ..., x_n\}$.

In this paper we will give a complete characterization, from both an operational and a logical point of view, of v-solvable terms.

A key observation is that, in order to characterize the class of v-solvable terms from an operational point of view, the β_v-reduction is too weak. In fact there are β_v-normal forms which are v-unsolvable, as for example the term:

$$\lambda x.(\lambda y.\Delta)(xI)\Delta$$

which is operationally equivalent to $\lambda x.\Delta\Delta$. So, in order to characterize operationally the v-solvability, a more refined tool must be designed. To do so, we extend the notion of valuability (*i.e.*, termination) to open terms, by defining a term M being *potentially valuable* if and only if there is a substitution s, replacing variables by closed values, such that $s(M)$ is valuable. It turns out that the class of the v-solvable terms is properly contained in that one of the potentially valuable terms. We will show that the potentially valuable terms are completely characterized through an evaluation machine, that we call *inner machine*, performing the classical β-reduction according to the innermost-lazy strategy. It is important to notice that the operational equivalence induced by the inner machine coincides with $\approx_s$. Another evaluation machine, the *ahead machine*, which is based on the previous one, gives the desired characterization of v-solvable terms. It turns out that a term M is v-solvable if and only if it reduces, using the classical β-reduction with the leftmost-innermost strategy, to a term of the shape:

$$\lambda x_1 \ldots x_n.x_i P_1 \ldots P_m$$

where P_i is potentially valuables $(1 \leq i \leq m)$. Note that this definition cannot be expressed through the β_v-reduction. A preliminary version of these machines has been presented in [10].

Moreover we characterize both the potential valuability and the v-solvability from a logical point of view, defining an intersection type assignment system, which gives type exactly to the potentially valuable terms, and gives a type of a particular shape exactly to the v-solvable terms. Such a type assignment system is inspired to that one defined by [4] for reasoning about canonical denotational semantics of $\lambda\beta_v$-calculus.

Let recall that a $\lambda\beta$-theory is called *sensible* if it equates all unsolvable terms, and *semi-sensible* if it never equates a solvable term to an unsolvable one. We can extend in an obvious way this definition to a $\lambda\beta_v$-theory, calling it *v-sensible*

if it equates all the v-unsolvable terms, and *v-semi-sensible* if it never equates a v-solvable term to a v-unsolvable one. According to the previous definition of v-solvability, the secd-operational theory, *i.e.*, the theory $T_{secd} = \{(M, N) | M \approx_s N\}$ is not v-sensible, as expected. Indeed $\Delta\Delta$ and $\lambda x.\Delta\Delta$ are two different unsolvable terms which are not equated in T_{secd}. This depends on the fact that the secd machine evaluates in a lazy way: indeed also the operational semantics of the lazy λ-calculus is not v-sensible. Moreover, T_{secd} is not v-semi-sensible. In fact it turns out that it equates the identity combinator I to a v-unsolvable term. This equivalence is not surprising, since it is a consequence of the fact that, in the minimal canonical model of $\lambda\beta_v$-calculus, showed in [4], which is built by an inverse limit construction, all projections are λ-representable. We will give here a purely syntactic proof of it.

The paper is organized as follows. In Section 2 the $\lambda\beta_v$-calculus and its operational semantics are recalled. In Section 3 the notions of potentially valuable and v-solvable term are introduced. The operational characterizations of potentially valuable and v-solvable terms are given in Sections 4 and 5 respectively. Section 6 contains the logical characterization. The two appendices contain the more technical proofs.

2. The call-by-value λ-calculus

In this section we briefly recall the syntax and the operational semantics of the $\lambda\beta_v$-calculus, as stated by Plotkin [11]. The $\lambda\beta_v$-calculus is a restriction of the classical λ-calculus, based on the notion of value. In particular, the restriction concerns the evaluation rule, the β-rule, which is replaced by the β_v-rule.

Definition 2.1. Let *Var* be a denumerable set of variables, ranged over by $x, y, z, \cdots$

Let Λ be the set of λ-terms, built out by the following grammar:

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

Terms will be ranged over by $M, N, P, Q, \ldots$ A term of the form MN is called *application* while a term of the form $\lambda x.M$ is called *abstraction.*

The set of *values* is the set $Val \subset \Lambda$ defined as follows:

$$Val = Var \cup \{\lambda x.M | \quad x \in Var \quad and \quad M \in \Lambda\}.$$

It is straightforward to check that every term is of the shape:

$$\lambda x_1 \ldots x_n.\zeta M_1 \ldots M_m$$

for some $n, m \geq 0$, where $M_i \in \Lambda$, and ζ is either a variable or an abstraction.

Notation 2.2. *Free and bound variables are defined as usual, $FV(M)$ denotes the set of free variables of the term M and $\Lambda^0 \subset \Lambda$ denotes the set of closed terms,*

i.e., terms whose set of free variables is empty. Moreover $Val^0 \subset Val$ *denotes the subset* $Val \cap \Lambda^0$. *A context (denoted by* $C[\ ]$*) is a term with some occurrences of a hole; it can be built by a grammar obtained from that one for* λ*-terms by adjoining the hole to the set of variables.* $C[M]$ *denotes the context* $C[\ ]$ *once every occurrence of the hole has been replaced by the term* M. *Note that in the replacement free variables can be captured, and so they can become bound. As usual, terms are considered modulo* α*-conversion, i.e., modulo renaming of bound variables. Moreover* $\lambda x_1 \ldots x_n.M$ *is an abbreviation for* $\lambda x_1.(\lambda x_2.(\ldots\ (\lambda x_n.M)))$ *and* $M_1 \ldots M_m$ *is an abbreviation for* $((\ldots((M_1 M_2)M_3)\ldots)M_m)$. $\equiv$ *denotes the syntactical identity on terms.*

Definition 2.3. The *call-by-value evaluation rule* is defined as follows:

$$(\beta_v) \qquad (\lambda x.M)N \to M[N/x] \qquad if \quad N \in Val$$

where $M[N/x]$ denotes the simultaneous replacement of every free occurrences of x in M by N, renaming bound variables in M to avoid variable clash.

Let $\to_v$, $\to_v^*$ and $=_v$ denote respectively the contextual closure of the β_v-rule, the reflexive and transitive closure of $\to_v$ and the reflexive, symmetric and transitive closure of $\to_v$.

The β_v reduction satisfies both the Church-Rosser property and the Standardization property (see [11]).

The evaluation of a program (closed term) is formalized through a reduction machine, which we call secd machine for pointing out that it is equivalent (w.r.t. the termination property) to the S.E.C.D. machine defined by Landin for evaluating expressions [8], once its input is restricted to pure λ-calculus terms. We give here a logical presentation of this machine, *i.e.*, the machine is defined as a set of logical rules, and the evaluation process is mimicked by a logical derivation.

The operational equivalence between terms is determined by observing the termination of computations carried out by the secd machine.

Definition 2.4. i) The *secd-machine* is a set of rules proving statement of the shape:

$$M \Downarrow_s N$$

where $M \in \Lambda^0$ and $N \in Val$. The rules are:

$$\frac{}{\lambda x.M \Downarrow_s \lambda x.M}(\text{abs})$$

$$\frac{M \Downarrow_s \lambda x.M' \quad N \Downarrow_s N' \quad M'[N'/x] \Downarrow_s P}{MN \Downarrow_s P}(\text{app}).$$

If $M \Downarrow_s N$, we will say that M is the *input* of the secd machine and N is the corresponding *output.*

Let $M \Downarrow_s$ be an abbreviation for: $\exists N$ such that $M \Downarrow_s N$. If $M \Downarrow_s$ we will say that M is *valuable.*

ii) Two terms M and N are *secd-operationally equivalent* ($M \approx_s N$) if and only if for all context $C[\ ]$ such that $C[M], C[N] \in \Lambda^0$,

$$C[M] \Downarrow_s \quad \Leftrightarrow \quad C[N] \Downarrow_s .$$

It is immediate to verify that the secd-machine is deterministic, *i.e.*, if $M \Downarrow_s$ then there is exactly one N such that $M \Downarrow_s N$ and moreover there is exactly one derivation proving $M \Downarrow_s N$. So, if $M \Downarrow_s$, then we can define the *number of steps* of the secd-machine when filled with input M (notation: $step_s(M)$) as the number of applications of rules in the derivation proving $M \Downarrow_s N$.

It can be checked that the secd-machine reduces at the every step the leftmost outermost β_v-redex occurring in the input term not inside the scope of an abstraction, until a value is reached. The following proposition holds:

Proposition 2.5. *$M \in \Lambda^0$, $M \to_v^* N$ and $N \in Val$ if and only if M is valuable.*

Proof. By the standardization property of β_v-reduction, see [11]. □

3. Potentially valuable and v-solvable terms

In this section, both the notions of potentially valuable and v-solvable term are introduced, and their relation is discussed. The notion of potentially valuable term is the extension to open terms of the notion of termination in the secd machine. Note that this extension cannot be defined in the standard way, by defining an open term being potentially valuable if its closure is valuable, since the secd machine evaluates terms in a lazy way, so all abstractions are terminating.

Definition 3.1. A term M is *potentially valuable* if and only if there is a substitution s, replacing variables by closed values, such that $s(M)$ is valuable.

It is immediate to verify that a closed term is potentially valuable if and only if it is valuable.

Now, let us define the notion of v-solvability, for grasping the functional behaviour of terms.

Definition 3.2. A term M is *v-solvable* if and only if there are values $N_1, \ldots, N_n \in Val^0$ such that:

$$(\lambda x_1 \ldots x_m.M)N_1 \ldots N_n =_v I$$

where $I \equiv \lambda x.x$ and $FV(M) = \{x_1, \ldots, x_m\}$.

A term is *v-unsolvable* if and only if it is not *v-solvable.*

Lemma 3.3. *The class of v-solvable terms is properly included in the class of potentially valuable terms.*

Proof. Let first prove the inclusion. Let M be v-solvable, so for some closed values $N_1, \ldots, N_n$, $(\lambda x_1 \ldots x_m.M)N_1 \ldots N_n =_v I$. Without loss of generality, we can assume $m \leq n$. In fact, if $m > n$, then there are P_j $(n+1 \leq j \leq m)$ such that

$$(\lambda x_1 \ldots x_m.M)N_1 \ldots N_n P_{n+1} \ldots P_m =_v I;$$

just take $P_j = I$, for all j. Since I is a normal form, $(\lambda x_1 \ldots x_m.M)N_1 \ldots N_n \to_v^* I$ and so $M[N_1/x_1, \ldots, N_m/x_m]N_{m+1} \ldots N_n \to_v^* I$. By Proposition 2.5,

$$M[N_1/x_1, \ldots, N_m/x_m]N_{m+1} \ldots N_n \Downarrow_s$$

and this implies $M[N_1/x_1, \ldots, N_m/x_m] \Downarrow_s$. The inclusion is proper, since $\lambda x.\Delta\Delta$ is valuable, and so potentially valuable, but clearly v-unsolvable. □

4. Operational characterization of potentially valuable terms

In this section a new reduction machine, the *inner-machine*, is introduced, which operationally characterizes the potentially valuable terms, in the sense that it stops if and only if the input term is potentially valuable. The shape of the output results of such a machine, which we call canonical terms, is particularly interesting.

Definition 4.1. i) A term M is *canonical* if and only if it is either a value or of the shape:

$$xM_1 \ldots M_m \qquad (m \geq 0)$$

where M_i $(1 \leq i \leq m)$ is canonical. Let $\mathcal{C}$ be the set of canonical terms.

ii) The *inner-machine* is a set of rules proving statements of the shape:

$$M \Downarrow_i N$$

where $M \in \Lambda$ and $N \in \mathcal{C}$. The rules are:

$$\frac{m \geq 0 \quad M_j \Downarrow_i N_{j(1 \leq j \leq m)}}{xM_1 \ldots M_m \Downarrow_i xN_1 \ldots N_m}(\text{var})$$

$$\frac{Q \Downarrow_i R \quad P[R/z]M_1 \ldots M_m \Downarrow_i N}{(\lambda z.P)QM_1 \ldots M_m \Downarrow_i N}(\text{head})$$

$$\frac{}{\lambda x.Q \Downarrow_i \lambda x.Q}(\text{lazy}).$$

Let $M \Downarrow_i$ be an abbreviation for: $\exists N$ such that $M \Downarrow_i N$.

If $M \Downarrow_i N$, then M is the *input* of the i-machine and N is the corresponding *output*.

It is easy to prove that the *inner*-machine is well-defined, *i.e.*, if $M \Downarrow_i N$ then $N \in \mathcal{C}$, and moreover the machine is deterministic. So the notion of the number of steps (defined for the secd-machine in the previous section) can be extended to the i-machine in a straightforward way: if $M \Downarrow_i$, $step_i(M)$ denotes the number of steps performed by the i-machine on input M.

Note that the *inner*-machine executes the classical β-reduction (*call by name*) with an innermost-lazy strategy. In fact it performs at every step the lefmost innermost β-redex not inside the scope of a λ-abstraction, until either an abstraction or a head variable is reached, and, in the last case, it performs the same reduction strategy in parallel inside all the arguments.

Let us introduce a new reduction rule:

$$(\lambda z.M)N \rightarrow_{inner} M[N/z] \quad if \quad N \in \mathcal{C}.$$

Let $\rightarrow_i$, $\rightarrow_i^*$ and $=_i$ denote respectively the closure under application of the $\rightarrow_{inner}$, the reflexive and transitive closure of $\rightarrow_i$ and the reflexive, symmetric and transitive closure of $\rightarrow_i$.

Note that $\rightarrow_i$, $\rightarrow_i^*$ and $=_i$ are not contextual closed, but they are just closed under application. Indeed, the reduction relation obtained by the contextual closure of $\rightarrow_{inner}$ is not Church-Rosser (*e.g.*, the term $(\lambda x.(\lambda z.I)(x\Delta))\Delta$ would reduce both to I and $(\lambda z.I)(\Delta\Delta)$, which do not have a common reduct).

$\rightarrow_i^*$, as have been defined, is Church-Rosser (it can be easily proved), and moreover, being not closed under abstraction, it is intrinsecally lazy. As far as the example before is concerned, note that the term $(\lambda x.(\lambda z.I)(x\Delta))\Delta$ has just one i-redex, and it i-reduces only to $(\lambda z.I)(\Delta\Delta)$.

The *inner*-machine can be alternatively described as performing the $\rightarrow_i$ reduction. More precisely, it performs the lefmost outermost i-redex not inside the scope of a λ-abstraction, until either an abstraction or a head variable is reached, and in this last case it performs the same reduction strategy inside all the arguments. Moreover canonical terms are lazy normal forms with respect to the i-reduction rule, *i.e.*, a canonical term does not contain i-redexes, but inside the scope of a λ-abstraction. The following proposition clarifies the relation between the i-machine and the i-reduction.

Proposition 4.2. *$M \Downarrow_i N$ if and only if $M \rightarrow_i^* N$ and N is canonical.*

Proof. ($\Rightarrow$) By induction on the definition of the i-machine. ($\Leftarrow$) By induction on the lenght of the reduction $M \rightarrow_i^* N$. □

The behaviour of the *inner*-machine and of the secd-machine coincide on closed terms, as proved by the following proposition.

Proposition 4.3. *Let $M \in \Lambda^0$. Then $M \Downarrow_i N$ if and only if $M \Downarrow_s N$.*

Proof. ($\Rightarrow$) By induction on $step_i(M)$. In case the last used rule is (*lazy*) then the proof is obvious. The last used rule cannot be (*var*), since M is closed. Let the last used rule be (*head*), and let the derivation be:

$$\frac{Q \Downarrow_i R \quad P[R/x]M_1 \ldots M_m \Downarrow_i N}{(\lambda x.P)QM_1 \ldots M_m \Downarrow_i N}(\text{head}).$$

By induction $Q \Downarrow_s R$ and $P[R/x]M_1 \ldots M_m \Downarrow_s N$. This implies there is a derivation:

$$\frac{P[R/x]M_1 \ldots M_{m-1} \Downarrow_s \lambda z.S \quad M_m \Downarrow_s T \quad S[T/z] \Downarrow_s N}{P[R/x]M_1 \ldots M_m \Downarrow_s N}(\text{app}).$$

We will prove that $P[R/x]M_1 \ldots M_m \Downarrow_s N$ implies $(\lambda x.P)QM_1 \ldots M_m \Downarrow_s N$, by induction on m. The first step is obvious, by the rule (*app*). For the induction step, looking at the derivation showed before, we can assume $(\lambda x.P)QM_1 \ldots M_{m-1} \Downarrow_s \lambda z.S$. So we can build the derivation:

$$\frac{(\lambda x.P)QM_1 \ldots M_{m-1} \Downarrow_s \lambda z.S \quad M_m \Downarrow_s T \quad S[T/z] \Downarrow_s N}{(\lambda x.P)QM_1 \ldots M_m \Downarrow_s N}(\text{app}).$$

($\Leftarrow$) By induction on $step_s(M)$. If the last used rule is (*abs*) then the proof is obvious. Otherwise, let the derivation be:

$$\frac{P \Downarrow_s \lambda x.R \quad Q \Downarrow_s S \quad R[S/x] \Downarrow_s N}{PQ \Downarrow_s N}(\text{app}).$$

By induction $P \Downarrow_i \lambda x.R$, $Q \Downarrow_i S$ and $R[S/x] \Downarrow_i N$. So $(\lambda x.R)Q \Downarrow_i N$ (by the rule (*head*)). We will prove, by induction on $step_i(P)$, that $(\lambda x.R)Q \Downarrow_i N$ and $P \Downarrow_i \lambda x.R$ imply $PQ \Downarrow_i N$. The case $step_i(P) = 1$ is obvious. Otherwise, let $P \equiv (\lambda z.S)TP_1 \ldots P_r$ ($r \geq 0$). Then:

$$\frac{T \Downarrow_i T' \quad S[T'/z]P_1 \ldots P_r \Downarrow_i \lambda x.R}{(\lambda x.S)TP_1 \ldots P_r \Downarrow_i \lambda x.R}(\text{head}).$$

By induction, $S[T'/z]P_1 \ldots P_r \Downarrow_i \lambda x.R$ and $(\lambda x.R)Q \Downarrow_i N$ imply $S[T'/z]P_1 \ldots P_rQ \Downarrow_i \lambda x.R$. So we can build the following derivation:

$$\frac{T \Downarrow_i T' \quad S[T'/z]P_1 \ldots P_rQ \Downarrow_i \lambda x.R}{(\lambda x.S)TP_1 \ldots P_rQ \Downarrow_i \lambda x.R}(\text{head}).$$

□

In order to prove that the *inner*-machine completely characterizes the potentially valuable terms, we need some lemmas. Moreover, for proving them, we need to introduce a measure to be used for currying out the induction. Informally such a measure, that we call *weight*, is an upper bound to both the number of lazy β_v-reductions and of i-reductions needed for reducing a term to a value, if it is possible.

Definition 4.4. The *weight* of M (denoted by $\langle M \rangle$), is the partial function defined as follows:

- $\langle M \rangle = 0$ if $M \in Val$
- $\langle (\lambda x.M_0)M_1 \ldots M_m \rangle = 1 + \langle M_1 \rangle + \langle M_0[M_1/x]M_2 \ldots M_m \rangle$.

Proposition 4.5. i) $M \to_v^* N \in Val$ *implies* $\langle M \rangle$ *is defined.*
ii) $M \to_\beta^* N \in Val$ *implies* $\langle M \rangle \geq \langle N \rangle$.
iii) $M \to_v^* N \in Val$ *implies* $\langle M \rangle \geq \langle N \rangle$.

Proof. See Appendix A. □

Lemma 4.6. *Let $M \equiv (\lambda x.M_0)M_1 \ldots M_m \in \Lambda^0$. If there is $N \in Val$ such that $M \to_v^* N$ then, for all i $(1 \leq i \leq m)$, $M_i \to_v^* N_i$, for some value N_i, and $\langle M_i \rangle < \langle M \rangle$.*

Proof. By Proposition 4.3, $M \to_v^* N \in Val$ implies $M \Downarrow_i N$. So we will give the proof by induction on $step_i(M)$. If $step_i(M) = 1$, then $m = 0$, and the proposition is vacuosly true. Otherwise, the derivation of $M \Downarrow_i N$ is of the shape:

$$\frac{M_1 \Downarrow_i R_1 \quad M_0[R_1/x]M_2 \ldots M_m \Downarrow_i N}{(\lambda x.M_0)M_1 \ldots M_m \Downarrow_i N}(\text{head}).$$

By Proposition 4.3, both M_1 and $M_0[R_1/x]M_2 \ldots M_m$ are reducible to a value, and $\langle M_1 \rangle < \langle M \rangle$ follows by definition of $\langle M \rangle$, while $\langle M_i \rangle < \langle M \rangle$ $(2 \leq i \leq m)$ follows by induction. □

Lemma 4.7. *Let $M \in \Lambda$ and let $FV(M) \subseteq \{x_1, \ldots, x_n\}$. If there are $P_1, \ldots, P_n \in Val^0$ and $\bar{M} \in Val^0$ such that $M[P_1/x_1, \ldots, P_n/x_n] \to_v^* \bar{M}$, then there is $N \in \Lambda$ such that both $M \Downarrow_i N$ and $N[P_1/x_1, \ldots, P_n/x_n] \to_v^* \bar{M}$.*

Proof. In this proof we will denote $R[P_1/x_1, \ldots, P_n/x_n]$ by R', for every $R \in \Lambda$.

The proof is carried out by induction on the weight $\langle M' \rangle = k$. Note that $M' \in \Lambda^0$.

$k = 0$: Then M' is already a value and, since it is closed, it must be $M' \equiv \lambda z.P'$. There are two cases:

1. $M \equiv x_j$ and $P_j \equiv \lambda z.P'$. So $x_j \Downarrow_i x_j$, by the (var) rule, with $m = 0$, and $x_j[y/x_j] \to_v^* y$.
2. $M \equiv \lambda z.P$. This case is obvious since the inner-machine stops by the rule $(lazy)$.

$k > 0$: Then $M' \equiv (\lambda z.P')M_1' \ldots M_m'$ $(m > 0)$. Then two cases are possible (with respect to the shape of M):

1. $M \equiv x_j M_1 \ldots M_m$ and $P_j \equiv \lambda z.P'$. So

$$(x_j M_1 \ldots M_m)[P_1/x_1, \ldots, P_n/x_n] \equiv P_j M_1' \ldots M_m' \to_v^* \bar{M}.$$

By Lemma 4.6 and Proposition 2.5, $M_i' \to_v^* \bar{M}_i \in Val$ and $\langle M_i' \rangle < k$, so by induction there is N_i such that $M_i \Downarrow_i N_i$ and $N_i[P_1/x_1, \ldots, P_n/x_n] \to_v^* \bar{M}_i$ $(1 \leq i \leq n)$.
So $x_j M_1 \ldots M_m \Downarrow_i x_j N_1 \ldots N_m$ by rule (var) and

$$P_j N_1' \ldots N_m' \to_v^* P_j \bar{M}_1 \ldots \bar{M}_m \to_v^* \bar{M}.$$

2. $M \equiv (\lambda z.P)M_1 \ldots M_m$.
By Lemma 4.6 and Proposition 2.5, $M_1' \to_v^* \bar{M}_1 \in Val$, and $\langle M_1' \rangle < k$. So by induction there is R such that $M_1 \Downarrow_i R$ and $R' \to_v^* \bar{M}_1$.

Clearly $P'[\bar{M}_1/z]M'_2\ldots M'_m \to_v^* \bar{M}$, thus $P'[M'_1/z]M'_2\ldots M'_m \to_v^* \bar{M}$. Moreover $\langle P'[M'_1/z]M'_2\ldots M'_m\rangle < k$ by definition of weight, so by Proposition 4.5.ii) $\langle P'[R'/z]M'_2\ldots M'_m\rangle < k$.
Furthermore $P'[R'/z]M'_2\ldots M'_m \to_v^* \bar{M}$, because $P'[\bar{M}_1/z]M'_2\ldots M'_m \to_v^* \bar{M}$ and $R' \to_v^* \bar{M}_1$. So by induction $P[R/z]M_2\ldots M_m \Downarrow_i N$, for some N, and

$$N[P_1/x_1, ..., P_n/x_n] \to_v^* \bar{M}.$$

Then $(\lambda z.P)M_1\ldots M_m \Downarrow_i N$, by the rule *head*.

□

Lemma 4.8. *Let $M \in \Lambda$, $FV(M) \subseteq \{x_1,\ldots,x_n\}$ and $O^r \equiv \lambda x_1\ldots x_{r+1}.x_{r+1}$.*
$M \Downarrow_i N$ implies $\forall r \geq step_i(M)$, there is $\bar{M} \in Val^0$ such that $M[O^r/x_1, ..., O^r/x_n] \to_v^* \bar{M}$.

Proof. By induction on $step_i(M)$. The case $step_i(M) = 1$ is trivial.

In the case the last applied rule is (var) the result is obvious.
Let the last applied rule be

$$\frac{Q \Downarrow_i R \quad P[R/z]M_1\ldots M_m \Downarrow_i N}{(\lambda z.P)QM_1\ldots M_m \Downarrow_i N}(\text{head}).$$

Let $q = step_i(Q)$. By induction $\forall r \geq q$, $Q[O^r/x_1, ..., O^r/x_n] \to_v^* \bar{Q} \in Val^0$, and by Lemma 4.7, $R[O^r/x_1, ..., O^r/x_n] \to_v^* \bar{Q}$.
Let $p = step_i(P[R/z]M_1\ldots M_m)$. By induction, $\forall h \geq p$:

$$(P[R/z]M_1\ldots M_m)[O^h/x_1, ..., O^h/x_n] \to_v^* \bar{N} \in Val.$$

In particular, since $step_i(M) = 1 + q + p$, for all $\rho \geq step_i(M)$, both

$$Q' \equiv Q[O^\rho/x_1, ..., O^\rho/x_n] \to_v^* \bar{Q'} \in Val^0$$

and

$$R' \equiv R[O^\rho/x_1, ..., O^\rho/x_n] \to_v^* \bar{Q'} \in Val^0.$$

Let $P' \equiv P[O^\rho/x_1, ..., O^\rho/x_n]$ and $M'_i \equiv M_i[O^\rho/x_1, ..., O^\rho/x_n]$;

$$(P[R/z]M_1\ldots M_m)[O^\rho/x_1, ..., O^\rho/x_n] \equiv P'[R'/z]M'_1\ldots M'_m \to_v^*$$

$$\to_v^* P'[\bar{Q'}/z]M'_1\ldots M'_m \to_v^* \bar{M},$$

for some $\bar{M} \in Val^0$. So

$$((\lambda z.P)QM_1\ldots M_m)[O^\rho/x_1, ..., O^\rho/x_n] \equiv (\lambda z.P')Q'M'_1\ldots M'_m \to_v^*$$

$$(\lambda z.P')\bar{Q}'M'_1 \ldots M'_m \to_v P'[\bar{Q}'/z]M'_1 \ldots M'_m \to_v^* \bar{M} \in Val^0.$$

□

Now we are able to prove the characterization theorem.

Theorem 4.9 (inner property). *$M \Downarrow_i$ if and only if M is potentially valuable.*

Proof. ($\Leftarrow$) The proof follows directly from Lemma 4.8. ($\Rightarrow$) By definition, M potentially valuable means that there is a substitution s, replacing variables by closed values, such that $s(M)$ is valuable. By Proposition 2.5, this implies $s(M) \to_v^* N \in Val^0$, and, by Lemma 4.7, this implies $M \Downarrow_i$. □

The *inner*-machine induces an operational equivalence on terms, defined in the usual way as follows.

Definition 4.10. Let $M, N \in \Lambda$. M and N are *i-operationally equivalent* ($M \approx_i N$), if and only if for all context $C[\ ]$ such that $C[M], C[N] \in \Lambda^0$, $C[M] \Downarrow_i \Leftrightarrow C[N] \Downarrow_i$.

By Proposition 4.3, $\approx_s$ and $\approx_i$ coincide.

It can be interesting to consider an extension of the operational equivalence, by dropping the restriction that contexts must be closing. Let define:

Definition 4.11 (i_{open}-equivalence). The term M and N are *i-open-operationally equivalent* ($M \approx_i^o N$) if and only if for all context $C[\ ]$, $C[M] \Downarrow_i \Leftrightarrow C[N] \Downarrow_i$.

Proposition 4.12. *$M \approx_i N$ if and only if $M \approx_i^o N$.*

Proof. Both directions can be proved by contrapposition.

($\Leftarrow$): Obvious.

($\Rightarrow$): Let $C[M] \Downarrow_i$ and $C[N] \Uparrow_i$, for some $C[.]$ not necessarely closing. $C[M]$ is potentially valuable, so there is a sequence of closed values $P_1, \ldots, P_n$ such that, if $FV(C[M]) \subseteq \{x_1, \ldots, x_n\}$, then $C[M][P_1/x_1, \ldots, P_n/x_n] \Downarrow_i$. Then the closing context, separating M and N is

$$C'[\ ] \equiv (\lambda x_1 \ldots x_m.C[\ ])P_1 \ldots P_n \underbrace{I \ldots I}_{m-n}$$

where $FV(C[N]) \cup FV(C[M]) = \{x_1, \ldots, x_m\}$.

□

Note that in general the equivalence induced by closing contexts does not coicide with that one induced by all context. For example, let us consider the machine which takes a λ-term as input, performs at every step the leftmost outermost β_v-redex not inside the scope of a λ-abstraction and stops on the lazy β_v-normal form. For closed terms this machine is equivalent to the secd-machine, so it induces the same equivalence. Consider the terms $P_0 \equiv \lambda y.(\lambda x.\Delta\Delta)(yI)$ and $P_1 \equiv \lambda y.\Delta\Delta$,

which are $\approx_s$. Let $C[\ ] \equiv [\ ](\lambda x.x(zI))$. Then the previous described machine stops on $C[P_0]$, while does not stop on $C[P_1]$.

It is important to notice that the behaviour of the *inner*-machine is in some sense anomalous, since $M \Downarrow_i N$ does not necessarily imply $M \approx_i N$. A counter example is the term $(\lambda yz.I)(xI)$: it is immediate to check that $(\lambda yz.I)(xI) \Downarrow_i \lambda z.I$, while the context $(\lambda x.[\ ])(\lambda x.\Delta\Delta)$ separates the two terms.

5. Operational characterization of v-solvability

In this section the operational characterization of the v-solvability is given, through a reduction machine, the ahead machine.

Such a reduction machine performs the β-reduction and uses the inner-machine as submachine.

Definition 5.1. i) A term M is a *v-head normal form* (v.h.n.f) if and only if it has the following shape:

$$\lambda x_1 \ldots x_n.xM_1 \ldots M_m$$

where $M_i \in \mathcal{C}$ $(1 \leq i \leq m)$. Let $\mathcal{VH}$ be the set of v-head normal forms.

ii) The *ahead-machine* is a set of rules proving statements of the shape:

$$M \Downarrow_a^0 N$$

where $M \in \Lambda$ and $N \in \mathcal{VH}$. The rules define an auxiliary machine too, proving statements of the shape $M \Downarrow_a^1 N$. The set of rules is the following, where $k \in \{0, 1\}$:

$$\frac{m \geq 0 \quad M_i \Downarrow_a^1 N_{i(1 \leq i \leq m}}{xM_1 \ldots M_m \Downarrow_a^k xN_1 \ldots N_m}(\text{var})$$

$$\frac{Q \Downarrow_a^1 R \quad P[R/z]M_1 \ldots M_m \Downarrow_a^k N}{(\lambda z.P)QM_1 \ldots M_m \Downarrow_a^k N}(\text{head})$$

$$\frac{}{\lambda x.Q \Downarrow_a^1 \lambda x.Q}(\text{lazy})$$

$$\frac{P \Downarrow_a^0 Q}{\lambda x.P \Downarrow_a^0 \lambda x.Q}(\lambda 0).$$

Let $M \Downarrow_a^k$ be an abbreviation for $M \Downarrow_a^k N$, for some N.

It is easy to check that the definition is correct, *i.e.*, $M \Downarrow_a^0 N$ implies $N \in \mathcal{VH}$. Furthermore, note that the machine of level 1 is the inner-machine, *i.e.* $M \Downarrow_a^1 N$ if and only if $M \Downarrow_i N$. The behaviour of the ahead-machine is not completely lazy: it enters under the external abstraction (if any) and then it works exactly as the *inner*-machine. In order to give a precise characterization, in terms of reductions, of the behaviour of the ahead-machine, we need to introduce a new reduction rule. Let $\rightarrow_I$, $\rightarrow_I^*$ and $=_I$ be the not lazy version of $\rightarrow_i$, $\rightarrow_i^*$ and $=_i$ respectively; namely $\rightarrow_I$, $\rightarrow_I^*$ and $=_I$ denote respectively the contextual closure of $\rightarrow_{inner}$, the reflexive and transitive closure of $\rightarrow_I$ and the reflexive, symmetric and transitive closure of $\rightarrow_I$.

Proposition 5.2. *$M \Downarrow_a^0 N$ if and only if N is of the shape $\lambda x_1 \ldots x_n.xN_1 \ldots N_m$, and $M \rightarrow_I^* \lambda x_1 \ldots x_n.xM_1 \ldots M_m$, and $M_j \Downarrow_i N_j$ $(1 \leq j \leq m)$.*

Note that the fact that the $\rightarrow_I$ reduction is not Church-Rosser does not create any problem, since the ahead-machine performs a particular strategy on it. So this new machine is deterministic, and if $M \Downarrow_a^k$, then $step_a^k(M)$ is the numbers of steps of the derivation proving $M \Downarrow_a^k N$.

For proving the desired characterization, we need two lemmas.

Lemma 5.3. *Let $M \in \Lambda$ and $FV(M) \subseteq \{x_1, \ldots, x_n\}$. If there are $P_1, \ldots, P_k \in Val^0$ such that $(\lambda x_1 \ldots x_n.M)P_1 \ldots P_k \rightarrow_v^* I$ then $M \Downarrow_a^0 N$ and $(\lambda x_1 \ldots x_n.N)P_1 \ldots P_k \rightarrow_v^* I$.*

Proof. As showed in the proof of Lemma 3.3, it is always possible to assume $k \geq n$.

Let $S \equiv (\lambda x_1 \ldots x_n.M)P_1 \ldots P_k$ and, for every $R \in \Lambda$, let R' be $R[P_1/x_1, .., P_n/x_n]$. The proof is given by induction on the following pair: ($\langle S \rangle$, number of symbols in M), ordered by the lexicographical order.

$(0, s)$: Then $S \in Val$ and the proof is trivial by rule $(\lambda 0)$.

$(h+1, s)$: Let analyze all possible shapes of the term M.

- $M \equiv x_j M_1 \ldots M_m$ $(m \geq 1)$. By hypothesis there are $P_1, \ldots, P_k \in Val^0$ such that

$$(\lambda x_1 \ldots x_n.x_j M_1 \ldots M_m)P_1 \ldots P_k \rightarrow_v^* P_j M_1' \ldots M_m' P_{n+1} \ldots P_k \rightarrow_v^* I$$

  where, by Lemma 4.6 and Proposition 2.5, $M_i' \rightarrow_v^* \bar{M}_i \in Val$. So by Lemma 4.7 (using the fact that $\Downarrow_a^1$ coincides with $\Downarrow_i$) we can state $M_i \Downarrow_a^1 N_i$ and

$$N_i' \equiv N_i[P_1/x_1, ..., P_n/x_n] \rightarrow_v^* \bar{M}_i.$$

  So $x_j M_1 \ldots M_m \Downarrow_a^0 x_j N_1 \ldots N_m$, by rule (var), and

$$(\lambda x_1 \ldots x_n.x_j N_1 \ldots N_m)P_1 \ldots P_k \rightarrow_v^* P_j N_1' \ldots N_m' P_{n+1} \ldots P_k \rightarrow_v^*$$

$$P_j \bar{M}_1 \ldots \bar{M}_m P_{n+1} \ldots P_k \rightarrow_v^* I.$$

- $M \equiv (\lambda z.P)QM_1 \ldots M_m$ $(m \geq 0)$. Then

$$S \equiv (\lambda x_1 \ldots x_n.(\lambda z.P)QM_1 \ldots M_m)P_1 \ldots P_k \rightarrow_v^*$$

$\rightarrow_v^* (\lambda z.P')Q'M_1'\ldots M_m'P_{n+1}\ldots P_k \rightarrow_v^* (\lambda z.P')\bar{Q}M_1'\ldots M_m'P_{n+1}\ldots P_k \rightarrow_v^* I$

where $Q' \rightarrow_v^* \bar{Q} \in Val$. Since $Q' \rightarrow_v^* \bar{Q} \in Val$, by Lemma 4.7 $Q \Downarrow_i R$ and

$$R' \rightarrow_v^* \bar{Q} \in Val.$$

Let us remind that $Q \Downarrow_i R$ coincides with $Q \Downarrow_a^1 R$ $(\star)$. Observe that

$$(\lambda z.P')Q'M_1'\ldots M_m'P_{n+1}\ldots P_k \rightarrow_\beta P'[Q'/z]M_1'\ldots M_m'P_{n+1}\ldots P_k \rightarrow_\beta^*$$

$$\rightarrow_\beta^* P'[R'/z]M_1'\ldots M_m'P_{n+1}\ldots P_k \rightarrow_v^* I \quad (\star\star).$$

Since

$$\langle S\rangle = n + \langle P_1\rangle + \ldots\ldots + \langle P_n\rangle + \langle(\lambda z.P')Q'M_1'\ldots M_m'P_{n+1}\ldots P_k\rangle$$

$$= n + \langle P_1\rangle + \ldots\ldots + \langle P_n\rangle + 1 + \langle Q'\rangle + \langle P'[Q'/z]M_1'\ldots M_m'P_{n+1}\ldots P_k\rangle$$

= (by Prop. 4.5.ii))

$$\langle P'[Q'/z]M_1'\ldots M_m'P_{n+1}\ldots P_k\rangle \geq \langle P'[R'/z]M_1'\ldots M_m'P_{n+1}\ldots P_k\rangle$$

we can apply the induction hypothesis $(\star\star)$ obtaining $P[R/z]M_1\ldots M_m \Downarrow_a^0 N$ $(\star\star\star)$ and $(\lambda x_1\ldots x_n.N)P_1\ldots P_k \rightarrow_v^* I$.
$(\star)$ and $(\star\star\star)$ together imply $(\lambda x.P)QM_1\ldots M_m \Downarrow_a^0 N$, and the proof is given.

- $M \equiv \lambda x.P$. This case is straighforward by induction on s.

□

Lemma 5.4. *Let $M \in \Lambda$ and let $FV(M) \subseteq \{x_1,\ldots,x_n\}$. If $M \Downarrow_a^0$ then $\forall r > \max\{n, step_a^0(M)\}$, $\exists h \geq 0$ such that*

$$(\lambda x_1\ldots x_n.M)\underbrace{O^r.....O^r}_{r} \rightarrow_v^* O^h$$

where $O^k \equiv \lambda x_1 ... x_{k+1}.x_{k+1}$.

Proof. First of all, observe that $P \Downarrow_i Q$ and $FV(P) \subseteq \{x_1,\ldots,x_n\}$ imply (by Lem. 4.8) that $\forall r \geq step_i(P)$, $\exists \bar{P} \in Val$ such that $P[O^r/x_1,...,O^r/x_n] \rightarrow_v^* \bar{P}$, and $Q[O^r/x_1,...,O^r/x_n] \rightarrow_v^* \bar{P}$ (by Lem. 4.7). Furthermore, $P \Downarrow_a^1$ if and only if $P \Downarrow_i$ and $step_i[P] = step_a^1[P]$.

The proof is carried out by induction on the derivation of $M \Downarrow_a^0 N$.

λ0: The proof follows immediately from the induction hypothesis.

var: Let

$$\frac{m \geq 0 \quad M_i \Downarrow_a^1 N_i}{xM_1\ldots M_m \Downarrow_a^0 xN_1\ldots N_m}(\text{var}).$$

By Lemma 4.8 $\forall r_j \geq step_i(M_j)$ $M_j[O^{r_j}/x_1, ..., O^{r_j}/x_n] \rightarrow_v^* \bar{M}_j \in Val$ $(1 \leq j \leq m)$. Let $\pi = step_a^0(M) = 1 + step_a^1(M_1) + \ ..... \ + step_a^1(M_m)$. Let $\rho > \max\{n, \pi\}$. Since $\pi > m$, then

$$(\lambda x_1 \ldots x_n . x_i M_1 \ldots M_m) \underbrace{O^\rho .....O^\rho}_{\rho} \rightarrow_v^* O^\rho M_1' \ldots M_m' \underbrace{O^\rho .....O^\rho}_{\rho - n} \rightarrow_v^* O^h$$

for some $h \geq 0$, where $M_i' \equiv M_i[O^\rho/x_1, ..., O^\rho/x_n]$.

head: Let the last used rule be

$$\frac{Q \Downarrow_a^1 R \quad P[R/z]M_1 \ldots M_m \Downarrow_a^0 N}{(\lambda z.P)RM_1 \ldots M_m \Downarrow_a^0 N}(\text{head}).$$

If $\pi = step_a^0(P[R/z]M_1 \ldots M_m)$, then by induction, $\forall r > \max\{n, \pi\}$, $\exists h' \geq 0$:

$$(\lambda x_1 \ldots x_n . P[R/z]M_1 \ldots M_m) \underbrace{O^r .....O^r}_{r} \rightarrow_v^* O^{h'}.$$

Let $\rho > \max\{n, 1 + step_a^1(Q) + \pi\}$. So by Lemma 4.8 $\exists \bar{Q}' \in Val^0$ such that both

$$Q' \equiv Q[O^\rho/x_1, ..., O^\rho/x_n] \rightarrow_v^* \bar{Q}'$$

and

$$R' \equiv R[O^\rho/x_1, ..., O^\rho/x_n] \rightarrow_v^* \bar{Q}'.$$

Let $P' \equiv P[O^\rho/x_1, ..., O^\rho/x_n]$ and $M_i' \equiv M_i[O^\rho/x_1, ..., O^\rho/x_n]$. Then

$$(\lambda x_1 \ldots x_n . P[R/z]M_1 \ldots M_m) \underbrace{O^\rho .....O^\rho}_{\rho} \equiv P'[R'/z]M_1' \ldots M_m' \underbrace{O^\rho .....O^\rho}_{\rho - n} \rightarrow_v^*$$

$$\rightarrow_v^* P'[\bar{Q}'/z]M_1' \ldots M_m' \underbrace{O^\rho .....O^\rho}_{\rho - n} \rightarrow_v^* O^{h''} \quad (h'' \geq 0)$$

by Church-Rosser, and finally

$$(\lambda x_1 \ldots x_n . (\lambda z.P)QM_1 \ldots M_m) \underbrace{O^\rho .....O^\rho}_{\rho} \rightarrow_v^* (\lambda z.P')Q'M_1' \ldots M_m' \underbrace{O^\rho .....O^\rho}_{\rho - n} \rightarrow_v^*$$

$$\rightarrow_v^* (\lambda z.P')\bar{Q}'M_1' \ldots M_m' \underbrace{O^\rho .....O^\rho}_{\rho - n} \rightarrow_v^* P'[\bar{Q}'/z]M_1' \ldots M_m' \underbrace{O^\rho .....O^\rho}_{\rho - n} \rightarrow_v^* O^{h''}.$$

□

Now we are able to prove our result.

Theorem 5.5 (v- solvability). *$M \Downarrow_a^0$ if and only if M is v-solvable.*

Proof. Let $FV(M) = \{x_1, \ldots, x_n\}$.

$\Rightarrow$: By Lemma 5.4 there are r and h such that:

$$(\lambda x_1 \ldots x_n.M)\underbrace{O^r.....O^r}_{r} \rightarrow_v^* O^h \quad (h \geq 0).$$

So $(\lambda x_1 \ldots x_n.M)\underbrace{O^r.....O^r}_{r} R_1 \ldots R_h \rightarrow_v^* I$, for all $R_1, \ldots, R_h \in Val$.

$\Leftarrow$: By Lemma 5.3.

□

It can be interesting to compare the notions of β_v-normal form, valuable term and v-solvable term. $\lambda x.(\lambda y.\Delta)(xI)\Delta$ and $\lambda x.\Delta\Delta$ are respectively a β_v-normal-form and a value, and are both v-unsolvable.

We can classify the v-unsolvable terms as follows.

Definition 5.6. Let P be v-unsolvable. P is of order 0 if and only if there is no Q such that $P =_i \lambda x.Q$. P is of order $k+1$ if $P =_i \lambda x.Q$ and k is the maximum integer such that Q is v-unsolvable of order k, while it is of infinite order if this integer does not exists.

All the v-unsolvable terms of order 0 can be consistently equated (see [4]). Moreover the relation between potentially valuable and v-solvable terms can be now stated as follows.

Proposition 5.7. *A term is not potentially valuable if and only if it is v-unsolvable of order* 0.

A $\lambda\beta_v$-theory is a conguence relation on terms closed under the β_v-equality. Let us recall that the λ-theories can be classified into sensible and semi-sensible, the former being these equating all unsolvable terms, and the latter these never equating a solvable term to an unsolvable one. We will introduce a similar classification for the $\lambda\beta_v$-theories.

Definition 5.8. i) A $\lambda\beta_v$-theory is *v-sensible* if and only if it equates all the v-unsolvable terms.

ii) A $\lambda\beta_v$-theory is *v-semi-sensible* if and only if it never equates a v-solvable term to a v-unsolvable one.

The $\lambda\beta_v$-theory induced by the secd operational equivalence is $T_{secd} = \{(M,N) | M \approx_s N\}$. It immediate to see that:

Proposition 5.9. *T_{secd} is not v-sensible.*

Proof. Consider the two terms $\Delta\Delta$ and $\lambda x.\Delta\Delta$. They are both v-unsolvable but the former is not valuable while the latter is a value. □

Now we will prove that T_{secd} is not v-semi-sensible.

Let $Y_v \equiv (\lambda xf.f(\lambda z.xxfz))(\lambda xf.f(\lambda z.xxfz))$, $A \equiv \lambda xyz.(\lambda uv.xuv)(yz)$ and $R \equiv Y_v AI$.

It is immediate to check that, for all $M \in Val$, $Y_v M \rightarrow_v^* M(\lambda z.Y_v Mz)$. The combinator Y_v is a recursion combinator in the call-by-value setting. We will prove

that R and I have the same operational behaviour. The next lemma will allow us to consider just contexts of a particular shape.

Lemma 5.10. *Let $M, N \in \Lambda$ and let $FV(M) \cup FV(N) \subseteq \{x_1, \ldots, x_n\}$.*
$M \not\approx_s N$ if and only if $\exists C[\ ] \equiv (\lambda x_1 \ldots x_n.[\ ])M_1 \ldots M_m$ such that $C[M], C[N] \in \Lambda^0$ and $C[M] \Downarrow_s$ and $C[M] \Uparrow_s$ or vice versa, for some $M_1, \ldots, M_m \in \Lambda$.

Proof. See [4] (Th. 33). □

Lemma 5.11. *Let $P_1, \ldots, P_k \in \Lambda^0 (k \geq 1)$.*
$IP_1 \ldots P_k \Downarrow_s M$ if and only if $RP_1 \ldots P_k \Downarrow_s \lambda v.(\lambda z.Y_v Az)Mv$.

Proof. By induction on k.

$k = 1$: $IP_1 \Downarrow_s M$ if and only if $P_1 \Downarrow_s M$ if and only if $RP_1 \Downarrow_s \lambda v.(\lambda z.Y_v Az)Mv$, by the secd-rules and $R \Downarrow_s \lambda x.(\lambda uv.(\lambda z.Y_v Az)uv)(Ix)$.

$k > 1$: By induction $IP_1 \ldots P_{k-1} \Downarrow_s N$ if and only if

$$RP_1 \ldots P_{k-1} \Downarrow_s \lambda v.(\lambda z.Y_v Az)Nv.$$

$IP_1 \ldots P_k \Downarrow_s M$ if and only if $N \equiv \lambda y.N'$ and $P_k \Downarrow_s M'$ and $N'[M'/y] \Downarrow_s M$ if and only if $((\lambda z.Y_v Az)Nv)[M_k/v] \Downarrow_s \lambda v.(\lambda z.Y_v Az)Mv$ (by secd-rules) if and only if $RP_1 \ldots P_k \Downarrow_s \lambda v.(\lambda z.Y_v Az)Mv$.

□

Theorem 5.12. *T_{secd} is not v-semi-sensible.*

Proof. We will prove that $R \approx_s I$. Since $R, I \in \Lambda^0$, by Lemma 5.10 we can consider just contexts of the shape $C[\ ] \equiv [\ ]M_1 \ldots M_m$. If $m = 0$ then the secd machine stops for both I and R. Otherwise the proof follows from Lemma 5.11. □

6. Logical characterization

In this section we will present a type assignment system which allows a complete characterization of the v-solvable terms.

Definition 6.1. Let ν and α be two *type constants*. Let T be the set of types σ built out from the following grammar:

$$\sigma ::= \nu | \alpha | \sigma_1 \cap \ldots \cap \sigma_n \to \sigma \qquad (n \geq 1).$$

T will be ranged over by $\sigma, \tau, \pi, \rho, \mu \ldots$.

The $\to$ type-constructor is associative on the right and the intersection type-constructor $\cap$ binds stronger than $\to$. The types are considered modulo permutations of types bound by intersection costructor.

All types have the following shape:

$$\sigma_1^1 \cap \ldots \cap \sigma_{n_1}^1 \to \sigma_1^2 \cap \ldots \cap \sigma_{n_2}^2 \to \ldots\ldots \to \sigma_1^m \cap \ldots \cap \sigma_{n_m}^m \to \rho$$

for some m, n where ρ is either ν or α. In the latter case the type is named *proper*. Let a proper type be denoted by σ_p and the subset of proper type by T_p. In the rest of the paper, we will use $\equiv$ for denoting the synctactical identity both on terms and types.

Definition 6.2.

i) Let a basis be a finite set of assignments of the shape $x : \sigma$, where x is a variable and σ is a type. If B is a basis, let $dom(B) = \{x \mid x : \sigma \in B\}$.

ii) The following type assignment system proves statements of the shape: $B \vdash M : \sigma$ where B is a basis, $M \in \Lambda$ and $\sigma \in T$. The rules are:

$$\frac{}{\{x : \sigma\} \vdash x : \sigma}(var)$$

$$\frac{}{\emptyset \vdash \lambda x.M : \nu}(\nu)$$

$$\frac{B \vdash M : \tau \quad x \notin dom(B)}{B \vdash \lambda x.M : \nu \to \tau}(\to_\nu I)$$

$$\frac{B \cup \{x : \sigma_1, \ldots, x : \sigma_n\} \vdash M : \tau \quad x \notin dom(B) \quad (n \geq 0)}{B \vdash \lambda x.M : \sigma_1 \cap \ldots \cap \sigma_n \to \tau}(\to I)$$

$$\frac{B \vdash M : \sigma_1 \cap \ldots \cap \sigma_n \to \tau \quad (B_i \vdash N : \sigma_i)_{1 \leq i \leq n}}{B \cup_{i=1}^{n} B_i \vdash MN : \tau}(\to E).$$

We will denote by $\mathcal{D} : B \vdash M : \sigma$ a derivation $\mathcal{D}$ proving $B \vdash M : \sigma$; and by $\mathcal{D}' \subseteq \mathcal{D}$ the fact that $\mathcal{D}'$ is a subderivation of $\mathcal{D}$.

Proposition 6.3.

i) *Let* $M \in \mathcal{C}$. *Then* $\exists B, \sigma.\ B \vdash M : \sigma$. *In particular,* $\exists B.\ B \vdash M : \nu$.

ii) *Let* $M \in \mathcal{VH}$. *Then* $\exists B, \sigma_p.\ B \vdash M : \sigma_p$.

iii) *Let* $M \in Val$. *Then* $\exists B.\ B \vdash M : \nu$.

Proof. i) By induction on M. If M is an abstraction apply directly the rule (ν). If $M \equiv xM_1 \ldots M_m$, by induction $\exists B_i, \sigma_i.\ B_i \vdash M_i : \sigma_i \quad (1 \leq i \leq m)$; then for every σ, there is a derivation using one application of rule (var) followed by m applications of rule $(\to E)$, proving:

$$\{x : \sigma_1 \to \ldots \to \sigma_m \to \sigma\} \cup_i B_i \vdash M : \sigma.$$

Note that in particular it can be $\sigma \equiv \nu$.

ii) By induction on M. If $M \equiv xM_1 \ldots M_m$, take the proof of point i) and choose σ to be proper. The case M is an abstraction is trivial by induction.

iii) Trivial by (var) and (ν) rules.

□

We will prove that the typability in the above type assignment system is preserved by β-reduction and by a particular case of I-expansion. In particular, since $\to_i$ implies $\to_\beta$, and the i-expansion implies the expansion we considered, it turns out that the system is closed under $=_i$.

Lemma 6.4 (Subject reduction). *$B \vdash M : \sigma$ and $M \to_\beta^* M'$ imply $\exists B'$ such that $B' \vdash M' : \sigma$.*

Proof. The proof is by induction on the number m of β-reduction steps. The case $m = 0$ is obvious. Otherwise, let $\mathcal{D} : B \vdash M : \sigma$. We will build a new derivation $\mathcal{D}'$ proving $B' \vdash M' : \sigma$, where $M \to_\beta M' \to_\beta^* M^*$. Then the result follows by induction.

Let $M \equiv C[(\lambda x.P)Q]$ and $M' \equiv C[P[Q/x]]$. If $(\lambda x.P)Q$ occurs in a subterm of M typed using the rule (ν), then $\mathcal{D}'$ can be obtained by replacing $(\lambda x.P)Q$ by $P[Q/x]$ in all subjects of $\mathcal{D}$. Otherwise there are two cases.

1) There is a subderivation $\mathcal{S} \subseteq \mathcal{D}$ of the shape:

$$\dfrac{\dfrac{\dfrac{\overline{\{x:\sigma_1\}\vdash x:\sigma_1}\,(\mathrm{var}) \quad \overline{\{x:\sigma_n\}\vdash x:\sigma_n}\,(\mathrm{var})}{\vdots}}{B \cup \{x:\sigma_1,\ldots,x:\sigma_n\} \vdash P : \tau}}{B \vdash \lambda x.P : \sigma_1 \cap \ldots \cap \sigma_n \to \tau}\,(\to I) \qquad \dfrac{\vdots\ \mathcal{D}_i}{B_i \vdash Q : \sigma_{i(1\le i\le n)}}$$

$$\dfrac{B \vdash \lambda x.P : \sigma_1 \cap \ldots \cap \sigma_n \to \tau \qquad B_i \vdash Q : \sigma_{i(1\le i\le n)}}{B \cup_{i=1}^n B_i \vdash (\lambda x.P)Q : \tau}\,(\to E)$$

Then by:

a) replacing the *i-th* application of the (*var*)-rule typing x by $\mathcal{D}_i$ $(1 \le i \le n)$;
b) replacing every occurrence of x in the subjects by Q;
c) replacing every assignment $x : \sigma_i$ by the assignments in B_i;
d) erasing the rules $(\to I)$ and $(\to E)$;

the following subderivation $\mathcal{S}'$ can be built.

$$\dfrac{\dfrac{\mathcal{D}_1}{B_1 \vdash Q : \sigma_1}\ \vdots \qquad \dfrac{\mathcal{D}_n}{B_n \vdash Q : \sigma_n}\ \vdots}{B' \cup_{i=1}^n B_i \vdash P[Q/x] : \tau}$$

Note that $B' = B$. The desired $\mathcal{D}'$ is then obtained by replacing in $\mathcal{D}$ the subderivation $\mathcal{S}$ by $\mathcal{S}'$ and finally, by replacing in the rest of derivation $(\lambda x.P)Q$ by $P[Q/x]$.

2) Let the redex be introduced by an application of the rule $(\to_\nu I)$ followed by an application of the rule $(\to E)$. In this case the proof is similar, but it is possible

that $B' \neq B$ since either x does not occur in P or it occurs just in subterms of P typed by ν . □

Lemma 6.5. *Let $\mathcal{D} : B \vdash P[Q/x] : \sigma$. If $Q \in \mathcal{C}$ then $\exists \bar{B}$ such that $\bar{B} \vdash (\lambda x.P)Q : \sigma$.*

Proof. The occurrences of Q considered for the expansion in P can be divided in two groups: let $Q_1,\ldots,Q_q$ $(q \geq 0)$ be those occurrences of Q such that there is $\mathcal{D}_i : B_i \vdash Q_i : \sigma_i$ and $\mathcal{D}_i \subseteq \mathcal{D}$ $(1 \leq i \leq q)$ and let $Q_{q+1},\ldots,Q_{q+p}$ $(p \geq 0)$ be those occurrences of Q which are not typed by subderivations of $\mathcal{D}$ (*i.e.* these occurrences are in subterms of P typed by the constant ν). We will consider two cases, according to $q = 0$ or $q > 0$.

$q = 0$ Every occurrence of Q in $P[Q/x]$ occurs in a subterm of P typed by a rule (ν). Since, by hypothesis, $Q \in \mathcal{C}$, by Property 6.3, there is $B^\star$ and a derivation $\mathcal{S}^\star$ proving $B^\star \vdash Q : \nu$.

Let replace in $\mathcal{D}$ every such occurrence of Q by x (note that x is not typed); the result is a derivation $\mathcal{D}' : B \vdash P : \sigma$, where $x \notin dom(B)$.

Thus, $\bar{\mathcal{D}}$ is the following subderivation:

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{D}' \\ \vdots \\ B \vdash P : \sigma \quad x \notin dom(B)\end{array}}{B \vdash \lambda x.P : \nu \to \sigma}(\to_\nu I) \qquad \begin{array}{c}\mathcal{S}^\star \\ \vdots \\ B^\star \vdash Q : \nu\end{array}}{B \cup B^\star \vdash (\lambda x.P)Q : \sigma}(\to E)$$

$q > 0$ Let $\mathcal{S}$ be the subderivation obtained from $\mathcal{D}$ by:

a) replacing every $\mathcal{D}_i : B_i \vdash Q : \sigma_i$ by the rules

$$\dfrac{}{\{x : \sigma_i\} \vdash x : \sigma_i}(var) \qquad (1 \leq i \leq q)$$

b) replacing every rule $\dfrac{}{\vdash P_i[Q/x] : \nu}(\nu)$ by $\dfrac{}{\vdash P_i : \nu}(\nu)$, where P_i are the subterms of P containing the i-th untyped occurrence of Q $(q < i \leq q + p)$

c) replacing every occurrence of Q in the subjects by x and adjusting elsewhere the basis.

The result is $\mathcal{S} : B' \cup \{x : \sigma_1,\ldots,x : \sigma_q\} \vdash P : \tau$ where

$$B' = \Big\{ z : \mu \mid\ z : \mu \in B, z \in FV(M) \text{ and}$$

$$\text{the rule } \dfrac{}{\{z : \mu\} \vdash z : \mu}(var) \text{ occurs out of the } \mathcal{D}_i\ (1 \leq i \leq q) \Big\}.$$

The subderivation $\bar{\mathcal{D}}$ is obtained by adjoining an application of $(\to E)$ and $(\to I)$ rules to $\mathcal{S}$, in the following way:

$$\dfrac{\dfrac{\begin{array}{c}\mathcal{S}\\ \vdots\\ B' \cup \{x:\sigma_1,\ldots,x:\sigma_n\} \vdash P : \tau\end{array}}{B' \vdash \lambda x.P : \sigma_1 \cap \ldots \cap \sigma_q \to \tau}(\to I) \qquad \begin{array}{c}\mathcal{D}_i\\ \vdots\\ B_i \vdash Q : \sigma_{i(1\le i\le n)}\end{array}}{B' \cup_{i=1}^{q} B_i \vdash (\lambda x.P)Q : \tau}(\to E)$$

Note that $B' \cup_{i=0}^{q} B_i = B$.

□

Lemma 6.6 (Subject expansion). *$B \vdash M : \sigma$ and $\bar{M} \to_i^* M$ imply $\exists \bar{B}$ such that $\bar{B} \vdash \bar{M} : \sigma$.*

Proof. By induction on the number m of i-reduction steps. The case $m = 0$ is obvious. Let consider the case $m = 1$; then the general case follows directly from the induction hypothesis.

Let $M \equiv C[P[Q/x]]$ and $\bar{M} \equiv C[(\lambda x.P)Q]$. Since $\bar{M} \to_i M$, then $P[Q/x]$ does not occur under an abstraction. This means that, if $\mathcal{D} : B \vdash M : \sigma$, then $P[Q/x]$ cannot occur in a subterm of M typed by the rule (ν), so there is a subderivation $\mathcal{S} \subseteq \mathcal{D}$, such that $\mathcal{S} : B^* \vdash P[Q/x] : \tau$, for some B^* and τ. Then, by Lemma 6.5, there is $\mathcal{S}' : B' \vdash P[(\lambda x.P)Q] : \tau$. The conclusion is obtained from $\mathcal{D}$ by replacing $\mathcal{S}$ by $\mathcal{S}'$, by adjusting the basis and by replacing every occurrence of $P[Q/x]$ in the subjects in $\mathcal{D}$ by $(\lambda x.P)Q$. □

Theorem 6.7 (Characterization of potentially solvable terms). *$M \Downarrow_i$ if and only if $\exists B.\ B \vdash M : \sigma$.*

Proof. **($\Leftarrow$):** $M \Downarrow_i N$ if and only if $M \to_i^* N$ and $N \in \mathcal{C}$. Then the proof follows by induction on the lenght of the reduction from M to N, using Proposition 6.3.1 and Lemma 6.4 (since $M \to_i^* N$ implies $M \to_\beta^* N$).

($\Rightarrow$): See the Appendix B.

□

Theorem 6.8 (Characterization of v-solvable terms). *$M \Downarrow_a^0$ if and only if $\exists B.$ $B \vdash M : \sigma_p \in T_p$.*

Proof. **($\Leftarrow$):** By Proposition 5.2, $M \Downarrow_a^0$ implies $M \to_I^* N$, for some N. Then the proof can be carried out by induction on the number of steps of the reduction from M to N, using Proposition 6.3.1 and Lemma 6.4 (since $M \to_I^* N$ implies $M \to_\beta^* N$).

($\Rightarrow$): See the Appendix B.

□

The type assignment system $\vdash$, presented here, is strongly related to the system presented in [4] for reasoning on the denotational semantics of the $\lambda\beta_v$-calculus (let call it $\vdash_*$). Indeed, $\vdash$ can be obtained from $\vdash_*$ by restricting both the syntax of types and the rules of types formation, and by dropping the weakening and subtyping rules. The two systems have the same typability power. We could use directly system $\vdash_*$ for characterizing the v-solvability, introducing in it a suitable notion of proper type. But the characterization would have been less simple and clear.

7. Appendix A

In order to prove the proposition we need some additional lemmas.

Lemma 7.1. *$M \to_v N$ and $\langle N\rangle$ is defined imply $\langle M\rangle$ is defined.*

Proof. $M \to_v N$ means that $M \equiv C[(\lambda x.P)Q]$ and $N \equiv C[P[Q/x]]$. The proof is given by induction on $\langle N\rangle$. $\langle C[P[Q/x]]\rangle$ is defined implies that $C[\ ]$ must have one of the following shapes:

1. $C[\ ] \equiv [\ ]P_1 \ldots P_n$;
2. $C[\ ] \equiv (\lambda y.C'[\ ])P_1 \ldots P_n$;
3. $C[\ ] \equiv (\lambda y.P_0)P_1 \ldots P_{i-1}C'[\ ]P_{i+1} \ldots P_n$ $(n > 0,\ 1 \leq i \leq n)$.

Let recall that $(\lambda x.P)Q \to_v P[Q/x]$ implies Q is a value and so $\langle Q\rangle = 0$. We can assume, without loss of generality, that $x \notin FV(P_i)$, for all i $(1 \leq i \leq n)$; otherwise we can rename the bound variable x with a fresh variable.

Let $C[\ ]$ be of the shape 1. Then $\langle C[P[Q/x]]\rangle$ defined implies that $P[Q/x]$ is of the shape $M_0M_1 \ldots M_m$ $(m \geq 0)$, with $M_0 \equiv (\lambda z.M_0')$ or $M_0 \equiv z$ and $m = 0$.

- Let $m = 0$ and $n = 0$. Then $C[P[Q/x]] \equiv M_0 \in \mathrm{Val}$. Then $\langle C[P[Q/x]]\rangle = 0$, and $\langle C[(\lambda x.P)Q]\rangle = 1 + \langle Q\rangle + \langle P_1\rangle = 1 + 0 + 0 = 1$.
- Let $m \geq 0$, and $n \geq 1$. Then $C[P[Q/x]] \equiv M_0M_1 \ldots M_mP_1 \ldots P_n$ and $M_0 \equiv (\lambda z.M_0')$, so by hypothesis $\langle M_0M_1 \ldots M_mP_1 \ldots P_n\rangle$ is defined. Furthermore, since $\langle Q\rangle = 0$, $\langle N\rangle = \langle(\lambda x.((\lambda z.M_0)M_1 \ldots M_m)QP_1 \ldots P_n\rangle = 1 + \langle Q\rangle + \langle(\lambda z.M_0)M_1 \ldots M_mP_1 \ldots P_n\rangle$ is defined.
- The case $m = 0$ and $n \geq 1$ is similar to the previous one, but simpler.

Let $C[\ ]$ be of the shape 2. $\langle C[P[Q/x]]\rangle = 1 + \langle P_1\rangle + \langle C'[P[Q/x]][P_1/y]P_2 \ldots P_n\rangle$. By induction $\langle C'[(\lambda x.P)Q][P_1/y]P_2 \ldots P_n\rangle$ is defined, and we are done, since by definition $\langle C[(\lambda x.P)Q]\rangle = 1 + \langle P_1\rangle + \langle C'[(\lambda x.P)Q][P_1/y]P_2 \ldots P_n\rangle$.

Let $C[\ ]$ be of the shape 3.

- Let $i = 1$. So $\langle C[P[Q/x]]\rangle = 1 + \langle C'[P[Q/x]]\rangle + \langle P_0[C'[P[Q/x]]/y]P_2 \ldots P_n\rangle$. By induction $\langle C'[(\lambda x.P)Q]\rangle$ and $\langle P_0[C'[(\lambda x.P)Q]/y]P_2 \ldots P_n\rangle$ are both defined and we have done, since $\langle C[(\lambda x.P)Q]\rangle = 1 + \langle C'[(\lambda x.P)Q]\rangle + \langle P_0[C'[(\lambda x.P)Q]/y]P_2 \ldots P_n\rangle$.

– Let $i > 1$. Then

$$\langle C[P[Q/x]]\rangle = 1 + \langle P_1\rangle + \langle P_0[P_1/y]P_2 \ldots P_{i-1}[C'[P[Q/x]]/y]P_{i+1}\ldots P_n\rangle,$$

and by induction $\langle P_0[P_1/y]P_2\ldots P_{i-1}[C'[(\lambda x.P)Q]]/y]P_{i+1}\ldots P_n\rangle$ is defined. So we have done, since by definition

$$\langle C[(\lambda x.P)Q]\rangle = 1 + \langle P_1\rangle + \langle P_0[P_1/y]P_2\ldots P_{i-1}[C'[(\lambda x.P)Q]]/y]P_{i+1}\ldots P_n\rangle.$$

□

Lemma 7.2. *$\langle M\rangle$ is defined and $M \to_\beta^* N$ imply $\langle M\rangle \geq \langle N\rangle$.*

Proof. Let $\langle M\rangle = k$. The proof is given by induction on the following pair: (k, p), where p is the numbers of steps of the reduction $M \to_\beta^* N$, ordered according to the lexicographical order. The cases where either $\langle M\rangle = 0$ or $p = 0$ are trivial. Let the reduction path be: $M \to_\beta R_1 \to_\beta \ldots\ldots \to_\beta R_p \equiv N$ $(p \geq 0)$. Clearly $M \equiv (\lambda x.M_0)M_1\ldots M_m$, so let $h' = \langle M_1\rangle$, $h'' = \langle M_0[M_1/x]M_2\ldots M_m\rangle$ and so $k = 1 + h' + h''$. Let $p = 1$. There are three cases:

1. If $R_1 \equiv M_0[M_1/x]M_2\ldots M_m$ then $\langle R_1\rangle = h'' < k$. Thus the proof is trivial.
2. Let $R_1 \equiv (\lambda x.N_0)M_1N_2\ldots N_m$ where $\exists j$ (unique) $M_j \to_\beta N_j$, while for $i \neq j$ $M_i \equiv N_i$ $(0 \leq i, j \leq m$ and $i, j \neq 1)$. Clearly $M_0[M_1/x]M_2\ldots M_m \to_\beta N_0[M_1/x]N_2\ldots N_m$; thus $h'' < k$ implies, by induction $\langle N_0[M_1/x]N_2\ldots N_m\rangle \leq h''$. Finally $\langle R_1\rangle = 1 + \langle M_1\rangle + \langle N_0[M_1/x]N_2\ldots N_m\rangle \leq k$ and the proof is done.
3. Let $R_1 \equiv (\lambda x.M_0)N_1M_2\ldots M_m$, where $M_1 \to_\beta N_1$. By induction $\langle M_1\rangle \geq \langle N_1\rangle$. Furthermore $h'' < k$ and $M_0[M_1/x]M_2\ldots M_m \to_\beta^* M_0[N_1/x]M_2\ldots M_m$ imply, by induction, $\langle M_0[M_1/x]M_2\ldots M_m\rangle \leq h''$. Thus the conclusion follows, trivially, by definition of weight.

Since $(\langle M\rangle, p)$ is greater than $(\langle R_1\rangle, p-1)$, the complete proof follows by induction.

□

Now we are able to prove the Proposition 4.5.

Proof. i) By induction on the number of steps of the reduction $M \to_v^* \bar{M} \in Val$ using Lemma 7.1.

ii) By i) and by Lemma 7.2.

iii) By ii), since the β_v-reduction is a special case of the β-reduction.

□

8. Appendix B

Proof. $(\Rightarrow)$ *of Theorem 6.7.*

The proof will be given by a computability argument.

Let define the following predicate: $\mathcal{P}(\sigma, M) \Leftrightarrow$ there is a basis B such that $B \vdash M : \sigma$ and $M \Downarrow_i$.

Let $\vec{M}$ denote a sequence of terms $M_1\ldots M_m$, for some $m \geq 0$.

Proposition 8.1.

i) $\mathcal{P}(\sigma_1 \cap ... \cap \sigma_n \to \tau, x\vec{M})$ *and* $\mathcal{P}(\sigma_i, N)$ *imply* $\mathcal{P}(\tau, x\vec{M}N)$ $(1 \leq i \leq n)$.

ii) $\mathcal{P}(\tau, Mx)$ *and* $x \notin FV(M)$ *imply* $\mathcal{P}(\sigma_1 \cap ... \cap \sigma_n \to \tau, M)$, *for some* $\sigma_1, ..., \sigma_n$.

Proof. i) Clearly $\exists B, B'$ such that $B \vdash x\vec{M} : \sigma_1 \cap ... \cap \sigma_n \to \tau$ and $B' \vdash N : \sigma_i$ imply $B \cup B' \vdash x\vec{M}N : \tau$. Let us prove that if $x\vec{M} \Downarrow_i$ and $N \Downarrow_i$ then $x\vec{M}N \Downarrow_i$.

$x\vec{M} \Downarrow_i$ implies there is a derivation whose last applied rule is:

$$\frac{M_j \Downarrow_i M'_j \quad (1 \leq i \leq m)}{xM_1 ... M_m \Downarrow_i xM'_1 ... M'_m}.$$

Since $N \Downarrow_i N'$ the following derivation can be built:

$$\frac{M_j \Downarrow_i M'_j \ (1 \leq i \leq m) \quad N \Downarrow_i N'}{xM_1 ... M_m N \Downarrow_i xM'_1 ... M'_m N'}.$$

$\mathcal{P}(\tau, x\vec{M}N)$ follows by rule $(\to E)$.

ii) The proof is given by induction on the derivation proving $Mx \Downarrow_i$. The only not trivial case is when the last applied rule is:

$$\frac{Q \Downarrow_i R \quad P[R/z]M_1 ... M_m x \Downarrow_i N}{(\lambda z.P)QM_1 ... M_m x \Downarrow_i N}$$

where $M \equiv (\lambda z.P)QM_1 ... M_m$. By induction $P[R/z]M_1 ... M_m x \Downarrow_i N$ implies $P[R/z]M_1 ... M_m \Downarrow_i$, so $M \Downarrow_i$. Moreover, for some basis B, B_i $(1 \leq i \leq n)$, there is a derivation ending with a rule:

$$\frac{B \vdash M : \sigma_1 \cap ... \cap \sigma_n \to \tau \quad (B_i \vdash x : \sigma_i)_{1 \leq i \leq n}}{B \cup_i B_i = B \cup \{x : \sigma_1, ..., x : \sigma_n\} \vdash Mx : \tau} (\to E.)$$

Since $x \notin FV(M)$, it must be $x \notin dom(B)$. Then $\mathcal{P}(\sigma_1 \cap ... \cap \sigma_n \to \tau, M)$.

Note that the case $\sigma \equiv \nu \to \tau$, is implicitely considered.

□

Now let define the following computability predicate:

- $Comp^i(\alpha, M) \Leftrightarrow \mathcal{P}(\alpha, M)$.
- $Comp^i(\nu, M) \Leftrightarrow \mathcal{P}(\nu, M)$.
- $Comp^i(\sigma_1 \cap ... \cap \sigma_n \to \tau, M) \Leftrightarrow \forall j \ (1 \leq j \leq n)$ $(Comp^i(\sigma_j, N)$ implies $Comp^i(\tau, MN))$ $(1 \leq j \leq n)$.

Lemma 8.2. $Comp^i(\sigma, M[N/x])$ *and* $N \in \mathcal{C}$ *imply* $Comp^i(\sigma, (\lambda x.M)N)$.

Proof. By induction on σ. The basis case follows from the definition of $\mathcal{P}$ and by Lemma 6.5. The general case follows immediately from the induction hypothesis.

□

We will prove: $B \vdash M : \sigma \Rightarrow Comp^i(\sigma, M) \Rightarrow \mathcal{P}(\sigma, M) \Rightarrow M \Downarrow_i$.

Lemma 8.3. i) $\mathcal{P}(\sigma, x\vec{M}) \Rightarrow Comp^i(\sigma, x\vec{M})$.
ii) $Comp^i(\sigma, M) \Rightarrow \mathcal{P}(\sigma, M)$.

Proof. By mutual induction on σ. The only not trivial case is $\sigma \equiv \mu_1 \cap ... \cap \mu_m \to \tau$.

i) $Comp^i(\mu_j, N)$ $(\star) \Rightarrow \mathcal{P}(\mu_j, N)$ by induction on ii) $(1 \leq j \leq m)$. $\mathcal{P}(\mu_1 \cap ... \cap \mu_m \to \tau, x\vec{M})$ and $\mathcal{P}(\mu_j, N)$ $(1 \leq j \leq m) \Rightarrow \mathcal{P}(\tau, x\vec{M}N)$ by Property 8.1.1 $\Rightarrow Comp^i(\tau, x\vec{M}N)$ $(\star\star)$ by induction. Finally $(\star)$ and $(\star\star)$ togheter imply $Comp^i(\mu_1 \cap ... \cap \mu_m \to \tau, x\vec{M})$, by definition of $Comp^i$.

ii) Let $x \notin FV(M)$. Clearly $\mathcal{P}(\mu_j, x) \Rightarrow Comp^i(\mu_j, x)$ $(1 \leq j \leq m)$ by induction on i). $Comp^i(\mu_1 \cap ... \cap \mu_m \to \tau, M)$ and $Comp^i(\mu_j, x)$ $(1 \leq j \leq m) \Rightarrow Comp^i(\tau, Mx)$, by definition of $Comp^i \Rightarrow \mathcal{P}(\tau, Mx)$ by induction $\Rightarrow \mathcal{P}(\mu_1 \cap ... \cap \mu_m \to \tau, M)$ by Property 8.1.2.

□

Lemma 8.4. *Let* $FV(M) = \{x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+k} | n, k \geq 0\}$, $B = \{x_j : \sigma_r^j | 1 \leq j \leq n, 1 \leq r \leq m_j, 1 \leq m_j\}$ *and* $\mathcal{D} : B \vdash M : \tau$.

If $Q_j \in \mathcal{C}$ $(1 \leq j \leq n+k)$ *and* $Comp^i(\sigma_r^h, Q_h)$ $(1 \leq r \leq m_j, 1 \leq h \leq n)$ *then* $Comp^i(\tau, M[Q_1/x_1, ..., Q_{n+k}/x_{n+k}])$.

Proof. By induction on the derivation. If the last applied rule is either (var) or (ν) the proof is trivial. If the last applied rule is $(\to E)$ the proof follows by induction, the definition of computability and $(\to E)$. Let consider $(\to I)$. It must be $M \equiv \lambda y.P$ and $\tau \equiv \mu_1 \cap ... \cap \mu_p \to \tau'$, so

$$\frac{B \cup \{y : \mu_1, \ldots, y : \mu_p\} \vdash P : \tau' \quad y \notin dom(B)}{B \vdash (\lambda y.P) : \mu_1 \cap ... \cap \mu_p \to \tau'} (\to I.)$$

Let assume $N \in \mathcal{C}$ and $Comp^i(\mu_j, N)$ $(1 \leq j \leq p)$. This implies (by Lem. 8.3.2) that there are basis B_j such that $B_j \vdash N : \mu_j$ $(1 \leq j \leq p)$. By induction

$$Comp^i(\tau', P[Q_1/x_1, ..., Q_{n+k}/x_{n+k}, N/y]).$$

By Lemma 8.2:

$$Comp^i(\tau', (\lambda y.P[Q_1/x_1, ..., Q_{n+k}/x_{n+k}])N)$$

which togheter with $Comp^i(\mu_j, N)$, implies

$$Comp^i(\mu_1 \cap ... \cap \mu_p \to \tau', M[Q_1/x_1, ..., Q_{n+k}/x_{n+k}]).$$

The case $(\to_\nu I)$ is similar, using Proposition 6.3.1. □

Now we are able to conclude the proof.

Let $B \vdash M : \sigma$, $FV(M) = \{x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+k}\}$ and $B = \{x_j : \sigma_r^j | 1 \leq r \leq m_j, 1 \leq j \leq n, 1 \leq m_j\}$.

By Lemma 8.3i), $Comp^i(\sigma_r^j, x_j)$ $(1 \leq r \leq m_j, 1 \leq j \leq n)$. Then by Lemma 8.4 $Comp^i(\sigma, M)$, which implies $\mathcal{P}(\sigma, M)$ (by Lem. 8.3.2).

□

Proof. ($\Rightarrow$) *part of the Theorem 6.8.*

The proof is very similar to the previous one. Let $\sigma_p \in T_p$ and define the following predicate:

$\mathcal{R}(\sigma_p, M) \Leftrightarrow B \vdash M : \sigma_p$, for some basis B, and $M \Downarrow_a^0$.

Proposition 8.5.

1. $\mathcal{R}(\sigma_1 \cap ... \cap \sigma_n \to \tau_p, x\vec{M})$ *and* $\mathcal{R}(\sigma_i, N)$ *imply* $\mathcal{R}(\tau_p, x\vec{M}N)$ $(1 \leq i \leq n)$.
2. $\mathcal{R}(\tau_p, Mx)$ *and* $x \notin FV(M)$ *imply* $\mathcal{R}(\sigma_1 \cap ... \cap \sigma_n \to \tau_p, M)$, *for some* $\sigma_1, \ldots, \sigma_n$.

Proof. See Lemma 8.1. □

Now let us define a new computability predicate:

- $Comp^a(\alpha, M) \Leftrightarrow \mathcal{R}(\alpha, M)$.
- $Comp^a(\sigma_1 \cap ... \cap \sigma_n \to \tau_p, M) \Leftrightarrow (Comp^a(\sigma_k, N)$ implies $Comp^a(\tau_p, MN))$ $(1 \leq k \leq n)$.

The proof can be given following exactly the same lines than the proof of part $\Rightarrow$ of Theorem 6.7, *i.e.*, by proving that $B \vdash M : \sigma_p \Rightarrow Comp^a(\sigma_p, M) \Rightarrow \mathcal{R}(\sigma_p, M) \Rightarrow M \Downarrow_a^0$.

□

References

[1] S. Abramsky and C.H.L. Ong, Full Abstraction in the Lazy Lambda Calculus. *Information and Computation* **105** (1993) 159-267.

[2] H. Barendregt, The Lambda Calculus: Its syntax and semantics. North Holland (1984).

[3] H. Barendregt, M. Coppo and M. Dezani-Ciancaglini, A filter Lambda Model and the completeness of type assignment. *J. Symbolic Logic* **48** (1983) 85-116.

[4] L. Egidi, F. Honsell and S. Ronchi Della Rocca, Operational, denotational and logicals descriptions: A case study. *Fund. Inform.* (1992) 149-169.

[5] J.R. Hindley and J.P. Seldin, Introduction to Combinators and λ-Calculus. Cambridge University Press (1986).

[6] F. Honsell and S. Ronchi Della Rocca, An Approximation Theorem for topological lambda models and the topological incompleteness of Lambda Calculus. *J. Comput. Systems Sci.* **45** (1992) 49-75.

[7] M. Hyland, A syntactic characterization of the equalityin some Models for the Lambda Calculus. *J. London Math. Soc.* (1976) 361-370.

[8] P.J. Landin, The mechanical evaluation of expressions. *Comput. J.* (1964).

[9] R. Milner, M. Tofte and R. Harfen, The definition of standard ML, *M.I.T.* (1990).

[10] L. Paolini, La chiamata per valore e la valutazione pigra nel lambda calcolo. Tesi di Laurea, Universitá di Torino, Dip. Informatica (1997).

[11] G. Plotkin, Call by value, call by name and the λ-calculus. *Theoret. Comput. Sci.* (1975) 125-159.

[12] S. Ronchi Della Rocca, Basic λ-calculus. *Notes for the Summer School Proof and Types*, Chambery (1993).

[13] G.L. Steele and G.J. Sussman, The revised report on Scheme, *AI Memo 452, M.I.T.* (1978).

[14] S. Thompson, HASKELL: The craft of functional programming. *Addison-Wesley* (1996).

[15] C.P. Wadsworth, The relation between computational and denotational properties for Scott D_∞-models of the lambda calculus. *SIAM J. Comput.* **5** (1976) 488-522.

Communicated by G. Longo.
Received December, 1998. Accepted August, 1999.