Characterizing convergent terms in object calculi via intersection types

Ugo de'Liguoro

Dipartimento di Informatica, Università di Torino,
Corso Svizzera 185, 10149 Torino, Italy
deligu@di.unito.it
http://www.di.unito.it/~deligu

Abstract. We give a simple characterization of convergent terms in Abadi and Cardelli untyped Object Calculus (ς-calculus) via intersection types. We consider a λ-calculus with records and its intersection type assignment system. We prove that convergent λ-terms are characterized by their types. The characterization is then inherited by the object calculus via self-application interpretation.

1 Introduction

Concerning type systems for object oriented languages, theoretical research over the last decades has focused on subtyping, having as correctness criterion that typed programs will never rice "message not understood" exception at run time. Undoubtedly these are central issues in the field; nevertheless there are questions for which a different understanding of typing could be useful.

We move from the remark that, at least in case of major theoretical models, like the Objects Calculus of [1], or the Lambda Calculus of Objects of [18], typed terms do not normalize, in general. This is not surprising, since objects are essentially recursive, and these calculi are Turing complete; but this has the unpleasant consequence that types are completely insensitive with respect to termination. For example, the ς-term $[l = \varsigma(x)x.l].l$, which diverges, has type $[]$ in the basic first order type system OB_1; but an inspection of the derivation shows that it has any type A, since the object term $[l = \varsigma(x)x.l]$ may be typed by $[l{:}A]$, for any A (more precisely, for any A there exists a typed version $[l = \varsigma(x{:}[l{:}A])x.l].l$ of the diverging term, which has type A). Exactly the same remark applies to the Lambda Calculus of Objects, where a coding of the paradoxical combinator (following notation of [18]) $\mathbf{Y} \stackrel{\Delta}{=} \lambda f.\langle rec = \lambda x.f(x \Leftarrow rec)\rangle \Leftarrow rec$ is typable by $(\sigma \to \sigma) \to \sigma$, for any σ: then $\mathbf{Y}(\lambda x.x) : \sigma$ for all σ.

Should we care about termination in OO systems? If one's focus is on event driven systems, modularization, encapsulation or software reusability, as much as these features are supported by OO languages, probably not. But, after all, object orientation is a programming paradigm: if a program enters some infinite loop, without exhibiting any communication capability, it is true that no system inconsistency is responsible for this fact; it might also be clear that, due to some

clever typing, one knows in advance that no object will receive some unexpected message; nevertheless such a program is hardly useful.

In the present paper we show that a simple characterization of converging terms in the ς-calculus is achievable via a combination of (untyped) interpretations of object calculi and type assignment systems. We consider untyped λ-calculus with records as target language in which object terms from ς-calculus are translated, according to interpretations which have been proposed in the literature. We restrict attention to self-application interpretation (see [1], chapter 18), since it is easily proved that convergency in the ς-calculus (see [1], chapter 6) and in the λ-calculus with records are equivalent under such interpretation. We provide a characterization of convergency in the λ-calculus with records via an intersection type assignment system; the characterization is then inherited by the (untyped) ς-calculus.

Intersection types are better understood as "functional characters" (namely computational properties), than as sets of values: this is in accordance with the fact that in such systems any term has a (possibly trivial) type, and, moreover, that each term is typable by infinitely many types. On the other hand the set of types that can be given to a term describes its functional behaviour, that is its meaning (see e.g. [12, 6, 22, 14]). That convergency is characterized by typability within the system by types of some specific shape is basic with respect to the construction of denotational models using types (see [6, 3, 4]).

As a matter of fact, we consider the study of reduction properties via type assignment systems a preliminary step toward a theory of equivalence and of models for object calculi, based on domain logic and type assignment, which is left for further research.

1.1 Related work

The use of intersection types as a tool for the study of computational properties of the untyped λ-calculus begins with [25] and [12], and it is an established theory: see [22] for an exposition. These technique has been recently applied to the study of lazy λ-calculus [4], of parallel extension of λ-calculus [7, 16], and of call-by-value λ-calculus [17, 20]. Further studies of reduction properties via intersection types are reported in [15].

Intersection types have been also used by Reynolds in the design of his FORSYTHE language (see among many others [26, 24]). Although intersection semantics is the same as that in the literature quoted above, the meaning of types is close to the standard interpretation of polymorphism, and does not provide a tool for characterizing properties of reduction. However, the typing of records is strikingly similar to that one we have used here.

The source of the λ-calculus of records is [1], chapter 8, where it is contrasted to the ς-calculus. Interpretations have a long story, both as formalizations of the informal notion of object, and as translation of formal calculi: in particular the self-application interpretation originates from Kamin work [21]. Sources of further information on the subject of interpretations are [8, 2, 9], as well as [1], chapter 18. More recently Crary [10] advocated a use of intersection types for

object encoding, together with existential and recursive types. The paper discusses encoding into typed calculi, and is in the line of Reynolds and Pierce understanding of intersection types.

2 A λ-calculus with records

The syntax of terms is obtained from that of untyped λ-terms by adding records, equipped with selection and update operators:

$$M ::= x \mid \lambda x.M \mid MN \mid \langle l_i = M_i \ ^{i\in I}\rangle \mid M\cdot l \mid M\cdot l := N,$$

where I varies over finite sets of indexes, and $l_i \neq l_j$ if $i \neq j$. Note that $M\cdot l$ is not a subterm of $M\cdot l := N$ (much as $a.l$ is not a subterm of $a.l \Leftarrow \varsigma(x)b$ in the ς-calculus). We call Λ_R the resulting set of terms. The set of closed terms in Λ_R is denoted by Λ_R^0.

To give semantics to the calculus we first define a notion of reduction, and then choose a suitable subset of normal forms to be considered as values.

Definition 1. *The* weak reduction *relation, $\longrightarrow_w$, over Λ_R is defined by the following axioms and rules:*

(β) $(\lambda x.M)N \longrightarrow_w M[N/x]$,
(ν) $M \longrightarrow_w M' \ \Rightarrow \ MN \longrightarrow_w M'N$,
$(R1)$ $\langle l_i = M_i \ ^{i\in I}\rangle\cdot l_j \longrightarrow_w M_j$, *if* $j \in I$,
$(R2)$ $M \longrightarrow_w M' \ \Rightarrow \ M\cdot l \longrightarrow_w M'\cdot l$,
$(R3)$ $\langle l_i = M_i \ ^{i\in I}\rangle\cdot l_j := N \longrightarrow_w \langle l_i = M_i \ ^{i\in I\setminus\{j\}}, l_j = N\rangle$, *if* $j \in I$,
$(R4)$ $M \longrightarrow_w M' \ \Rightarrow \ M\cdot l := N \longrightarrow_w M'\cdot l := N$.

This is lazy reduction of the λ-calculus plus extra reduction rules for record terms.

Definition 2. *The set of* values $\mathcal{V}$ *is the union of the set* $\mathcal{R} = \{\langle l_i = M_i \ ^{i\in I}\rangle \mid \forall i \in I.\ M_i \in \Lambda_R^0\}$ *and the set $\mathcal{F}$ of closed abstractions. Then we define a convergency predicate w.r.t. weak-reduction by:*

i) $M \Downarrow V \stackrel{\Delta}{\Leftrightarrow} V \in \mathcal{V} \wedge M \stackrel{*}{\longrightarrow}_w V$,
ii) $M \Downarrow \stackrel{\Delta}{\Leftrightarrow} \exists V.\ M \Downarrow V$.

Definition 2 rules out closed normal forms like $(\lambda x.x)\cdot l := (\lambda x.x)$. Any value is a normal form w.r.t. $\longrightarrow_w$, but not viceversa: in particular any term representing a selection or an update over a label which is not defined in its operand in normal form, results in a blocked term which is not a value.

Terms:

$$a ::= x \mid [l_i = \varsigma(x_i)b_i \ ^{i\in I}] \mid a.l \mid a.l \Leftarrow \varsigma(x)b$$

Values:

$$v ::= [l_i : \varsigma(x_i)b_i \ ^{i\in I}]$$

Evaluation Rules:

$$\frac{}{v \downarrow v} \qquad \frac{a \downarrow [l_i : \varsigma(x_i)b_i \ ^{i\in I}] \quad b_j\{[l_i : \varsigma(x_i)b_i \ ^{i\in I}]/x_j\} \downarrow v \quad j \in I}{a.l_j \downarrow v}$$

$$\frac{a \downarrow [l_i : \varsigma(x_i)b_i \ ^{i\in I}] \quad j \in I}{a.l_j \Leftarrow \varsigma(y)b \downarrow [l_i : \varsigma(x_i)b_i \ ^{i\in I\setminus\{j\}}, l_j = \varsigma(y)b]}$$

Fig. 1. The untyped ς-calculus and its operational semantics.

3 Self-application interpretation and convergency

Syntax and operational semantics of the untyped ς-calculus are from [1], chapter 6; they are reported in figure 1. Note that substitution is written $a\{b/x\}$, instead of using square braces, to avoid confusion with notation of object terms.

We then introduce the self-application interpretation $[\![\_]\!]^S$, which is a mapping sending ς-terms into Λ_R. We take this definition from [1], chapter 18.

Definition 3. *Under self-application interpretation, ς-terms are translated according to the following rules:*

$$\begin{array}{ll} [\![x]\!]^S & \stackrel{\Delta}{=} x \\ [\![[l_i = \varsigma(x_i)b_i \ ^{i\in I}]]\!]^S & \stackrel{\Delta}{=} \langle l_i = \lambda x_i.[\![b_i]\!]^S \ ^{i\in I}\rangle \\ [\![a.l]\!]^S & \stackrel{\Delta}{=} ([\![a]\!]^S \cdot l)[\![a]\!]^S \\ [\![a.l \Leftarrow \varsigma(y)b]\!]^S & \stackrel{\Delta}{=} [\![a]\!]^S \cdot l := \lambda y.[\![b]\!]^S \end{array}$$

For the sake of relating convergency predicates via self-application interpretation, we give a one-step operational semantics of the ς-calculus which is equivalent to the big-step one.

Definition 4. *The reduction relation $\longrightarrow_\varsigma$ over ς-terms is defined by :*

i) $[l_i = \varsigma(x_i)b_i \ ^{i\in I}].l_j \longrightarrow_\varsigma b_j\{[l_i = \varsigma(x_i)b_i \ ^{i\in I}]/x_j\}$, *if* $j \in I$,
ii) $[l_i = \varsigma(x_i)b_i \ ^{i\in I}].l_j \Leftarrow \varsigma(y)b \longrightarrow_\varsigma [l_i = \varsigma(x_i)b_i \ ^{i\in I\setminus\{j\}}, l_j = \varsigma(y)b]$, *if* $j \in I$,
iii) $a \longrightarrow_\varsigma b \Rightarrow a.l \longrightarrow_\varsigma b.l$,
iv) $a \longrightarrow_\varsigma b \Rightarrow a.l \Leftarrow \varsigma(y)c \longrightarrow_\varsigma b.l \Leftarrow \varsigma(y)c$.

This reduction relation is weaker than the one-step reduction defined in [1] 6.2-1; on the other hand the subsequent lemma does not hold for that relation.

Lemma 1. *For any ς-term a and value v:*

$$a \downarrow v \Leftrightarrow a \stackrel{*}{\longrightarrow}_\varsigma v.$$

Proof. By induction over the definition of $a \xrightarrow{*}_\varsigma v$ (if part), and over the definition of $a \downarrow v$ (only if part).

□

The next theorem states that operational semantics is faithfully mirrored under self-application interpretation. We write $a \downarrow$ if there exists some v such that $a \downarrow v$.

Theorem 1. *For any ς-term a, $a \downarrow$ if and only if $[\![a]\!]^S \Downarrow$, that is the self-application interpretation preserves and respects the convergency predicate.*

Proof. We observe that:

a) if $a \longrightarrow_\varsigma b$, then $[\![a]\!]^S \xrightarrow{+}_w [\![b]\!]^S$ (proof by induction over $\longrightarrow_\varsigma$);
b) if $[\![a]\!]^S \longrightarrow_w M$, then, for some N and b,

$$M \xrightarrow{*}_w N \equiv [\![b]\!]^S \quad \text{and} \quad a \longrightarrow_\varsigma b,$$

(proof by induction over a);
c) v is a value in the ς-calculus if and only if $[\![v]\!]^S$ is such in Λ_R (immediate from the shape of $[\![v]\!]^S$);
d) if $V \in \mathcal{V}$ and $V \equiv [\![a]\!]^S$ for some a, then a is a value (by inspection of the definition of $[\![\cdot]\!]^S$).

If $a \downarrow$ then $a \xrightarrow{*}_\varsigma v$ for some value v, by lemma 1; hence $[\![a]\!]^S \xrightarrow{*}_w [\![v]\!]^S$, by (a), and $[\![v]\!]^S \in \mathcal{V}$ by (c); therefore $[\![a]\!]^S \Downarrow$.

Viceversa, if $[\![a]\!]^S \Downarrow$, then $[\![a]\!]^S \xrightarrow{*}_w V$ for some $V \in \mathcal{V}$; since V is a normal form, by (b) we know that $V \equiv [\![b]\!]^S$, with $a \xrightarrow{*}_\varsigma b$; it follows that b is a value, by (d), so that $a \downarrow$ by lemma 1.

□

4 A type assignment system

In this section we introduce the basic tool we use for analyzing the computational behaviour of λ-terms. It is a type assignment system, which is an extension of system CDV_ω (see [12, 5, 14]), also called $\mathcal{D}\Omega$ in [22]. To arrow and intersection type constructors we add a constructor for record types, which is from [1].

Types are defined according to the grammar:

$$\sigma ::= \alpha \mid \omega \mid \sigma \to \tau \mid \sigma \wedge \tau \mid \langle l_i : \sigma_i \,^{i \in I} \rangle,$$

where α ranges over a denumerable set of type variables, ω is a constant, σ and τ are types, I ranges over finite sets of indexes.

Definition 5. *The type assignment system CDV^R_ω is defined in figure 2. Judgments take the usual form $\Gamma \vdash M : \tau$, with $M \in \Lambda_R$. The context Γ is a finite set of assumptions $x : \sigma$, where each variable occurs at most once; the writing $\Gamma, x : \sigma$ is an abbreviation of $\Gamma \cup \{x : \sigma\}$, for $x \notin \Gamma$. We write $\Gamma \vdash_{CDV^R_\omega} M : \tau$ to abbreviate "$\Gamma \vdash M : \tau$ is derivable in system CDV^R_ω".*

$$\frac{}{\Gamma, x:\sigma \vdash x:\sigma}\ (Var) \quad \frac{\Gamma, x:\sigma \vdash M:\tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}\ (\to I) \quad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash MN : \tau}\ (\to E)$$

$$\frac{}{\Gamma \vdash M : \omega}\ (\omega) \qquad \frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash M : \tau}{\Gamma \vdash M : \sigma \wedge \tau}\ (\wedge I) \quad \frac{\Gamma \vdash M : \sigma_1 \wedge \sigma_2 \quad i = 1,2}{\Gamma \vdash M : \sigma_i}\ (\wedge E)$$

$$\frac{\Gamma \vdash M_i : \sigma_i \quad \forall i \in I \quad J \subseteq I}{\Gamma \vdash \langle l_i = M_i\ ^{i\in I}\rangle : \langle l_j : \sigma_j\ ^{i\in J}\rangle}\ (\langle\rangle I) \quad \frac{\Gamma \vdash M : \langle l_i : \sigma_i\ ^{i\in I}\rangle \quad j \in I}{\Gamma \vdash M \cdot l_j : \sigma_j}\ (\langle\rangle E)$$

$$\frac{\Gamma \vdash M : \langle l_i : \sigma_i\ ^{i\in I}\rangle \quad \Gamma \vdash N : \tau \quad j \in I}{\Gamma \vdash M \cdot l_j := N : \langle l_i : \sigma_i\ ^{i\in I\setminus\{j\}}, l_j : \tau\rangle}\ (\langle\rangle U)$$

Fig. 2. CDV^R_ω, intersection type assignment system for Λ_R.

Adding rules (ω), $(\wedge I)$ and $(\wedge E)$ to Curry rules (Var), $(\to I)$ and $(\to E)$ yields system CDV_ω. Rules $(\langle\rangle E)$ and $(\langle\rangle U)$ are from [1]; rule $(\langle\rangle I)$ is slightly more liberal than usual record type introduction rule: it is however a good example of the distinctive feature of intersection types. In fact, in ordinary typed systems, records are elements of some cartesian product; with respect to such interpretation rule $(\langle\rangle I)$ is unsound. But the meaning of the record type $\langle l_i : \sigma_i\ ^{i\in I}\rangle$ in our system (as it is formally defined below) is the property to reduce to some record such that, if some l_i is selected, for $i \in I$, then something having property σ_i is returned. Hence the extension of the property $\langle l_1 : \sigma_1, l_2 : \sigma_2\rangle$ is included in the extension of $\langle l_i : \sigma_i\rangle$, for any $i = 1, 2$. Finally, rule $(\langle\rangle U)$ is sound both w.r.t. the standard interpretation of record types and w.r.t. our interpretation: in fact, if the component types have to express properties of record components, and some of these is changed by an update, then it is reasonable that its type changes too. It would be unsound, instead, in system $\mathrm{OB}_{1<:}$ (see figure 4), as this rule immediately conflicts with the self type.

The essential reason for using intersection types and ω is type invariance under subject reduction and expansion, as stated in the next theorem; its proof follows a standard pattern and it is omitted.

Theorem 2 (Subject reduction and expansion). *Let $M, N \in \Lambda_R$ and σ be any type:*

i) if $\Gamma \vdash_{CDV^R_\omega} M : \sigma$ and $M \longrightarrow_w N$ then $\Gamma \vdash_{CDV^R_\omega} N : \sigma$;
ii) if $\Gamma \vdash_{CDV^R_\omega} N : \sigma$ and $M \longrightarrow_w N$ then $\Gamma \vdash_{CDV^R_\omega} M : \sigma$.

□

We observe that a restrictive rule for update, closer to rule (*Val Update*) of system $\mathrm{OB}_{1<:}$, such as

$$\frac{\Gamma \vdash M : \langle l_i : \sigma_i\ ^{i\in I}\rangle \quad \Gamma \vdash N : \sigma_j \quad j \in I}{\Gamma \vdash M \cdot l_j := N : \langle l_i : \sigma_i\ ^{i\in I}\rangle}\ (\langle\rangle U')$$

would break (ii) of theorem 2. Indeed, if $\mathbf{\Omega} \stackrel{\Delta}{=} (\lambda x.xx)(\lambda x.xx)$ and $\mathbf{I} \stackrel{\Delta}{=} \lambda x.x$, then $\langle l = \mathbf{\Omega}\rangle \cdot l := \mathbf{I} \longrightarrow_w \langle l = \mathbf{I}\rangle$; now $\langle l = \mathbf{I}\rangle$ has type $\langle l : \sigma \to \sigma\rangle$, for any σ, but, with $(\langle\rangle U')$ in place of $(\langle\rangle U)$, $\langle l = \mathbf{\Omega}\rangle \cdot l := \mathbf{I}$ has type $\langle l : \omega\rangle$ at best.

That types are "functional characters" is formalized by the following type interpretation, which associates to each type its extension. Note that it is a closed interpretation, namely extensions of types are subsets of Λ^0_R.

For $X, Y, X_i \subseteq \Lambda^0_R$ let us set (by overloading $\to$ and $\langle\rangle$):

$$\begin{array}{ll} X \to Y & = \{M \in \Lambda^0_R \mid \exists F \in \mathcal{F}.\, M \Downarrow F \wedge\ \forall N \in X.\, FN \in Y\}, \\ \langle l_i : X_i\rangle^{i\in I} & = \{M \in \Lambda^0_R \mid \exists R \in \mathcal{R}.\, M \Downarrow R \wedge\ \forall i \in I.\, R\cdot l_i \in X_i\}. \end{array}$$

We are now in place to define type interpretation, by associating to each type a subset of Λ^0_R, given an interpretation of type variables.

Definition 6 (Type interpretation). *A* closed interpretation $\mathcal{I}$ *is any map from type variables to subsets of* Λ^0_R*. It extends to a* closed type interpretation *(type interpretation for short)* $[\![\sigma]\!]_{\mathcal{I}} \subseteq \Lambda^0_R$ *as follows:*

$$\begin{array}{ll} [\![\alpha]\!]_{\mathcal{I}} & = \mathcal{I}(\alpha), \\ [\![\omega]\!]_{\mathcal{I}} & = \Lambda^0_R, \\ [\![\sigma \to \tau]\!]_{\mathcal{I}} & = [\![\sigma]\!]_{\mathcal{I}} \to [\![\tau]\!]_{\mathcal{I}}, \\ [\![\langle l_i : \sigma_i\ ^{i\in I}\rangle]\!]_{\mathcal{I}} & = \langle l_i : [\![\sigma_i]\!]^{i\in I}_{\mathcal{I}}\rangle, \\ [\![\sigma \wedge \tau]\!]_{\mathcal{I}} & = [\![\sigma]\!]_{\mathcal{I}} \cap [\![\tau]\!]_{\mathcal{I}}. \end{array}$$

With this definition of type interpretation there may be empty types, and even types which are empty under any interpretation $\mathcal{I}$, e.g. $(\omega \to \omega) \wedge \langle l : \omega\rangle$, which also shows that the problem would not be solved neither by some clever definition of interpretation of type variables, nor by eliminating type variables at all.

On the other hand one could weaken the definition of arrow and record type interpretations, by asking only that $M \Downarrow V$, for some V, and observing that both $(\lambda x.M)\cdot l$ and $\langle l = M\rangle N$ are in the interpretation of ω, for any closed M, N. But all this results into unnecessary weakening of the theorem below, and contradicts the philosophy of types as functional characters.

As a final remark about type interpretation let us define:

$$\sigma \leq \tau \stackrel{\Delta}{=} \forall \mathcal{I}.\, [\![\sigma]\!]_{\mathcal{I}} \subseteq [\![\tau]\!]_{\mathcal{I}}, \quad \text{and} \quad \sigma = \tau \stackrel{\Delta}{=} \sigma \leq \tau \leq \sigma.$$

Then we have some expected inequations: among them note (i) and (ii), which are subtyping in width and in depth respectively.

Proposition 1. *The following (in)equations hold:*

i) $I \supseteq J \Rightarrow \langle l_i : \sigma_i\ ^{i\in I}\rangle \leq \langle l_j : \sigma_j\ ^{j\in J}\rangle$;
ii) $\forall i \in I.\ \sigma_i \leq \tau_i \Rightarrow \langle l_i : \sigma_i\ ^{i\in I}\rangle \leq \langle l_i : \tau_i\ ^{i\in I}\rangle$;
iii) $\langle l_i : \sigma_i\ ^{i\in I}\rangle = \bigwedge_{i\in I}\langle l_i : \sigma_i\rangle$;
iv) $\langle l : \sigma\rangle \wedge \langle l : \tau\rangle = \langle l : \sigma \wedge \tau\rangle$.

□

Types:

$$A ::= \alpha \mid \mathtt{Top} \mid [l_i : A_i \ ^{i \in I}]$$

Subtyping rules:

$$\frac{}{E \vdash A <: \mathtt{Top}} \text{ (Sub Top)} \qquad \frac{I \supseteq J}{E \vdash [l_i : B_i \ ^{i \in I}] <: [l_i : B_i \ ^{i \in J}]} \text{ (Sub Object)}$$

$$\frac{}{E \vdash A <: A} \text{ (Sub Refl)} \qquad \frac{E \vdash A <: B \quad E \vdash B <: C}{E \vdash A <: C} \text{ (Sub Trans)}$$

Fig. 3. Subtyping rules for system $\mathrm{OB}_{1<:}$.

5 Typing interpretations of ς-calculus

We now turn to untyped interpretations of ς-terms into Λ_R. Since we are also interested in appreciating the closeness or the distance between our typing of the interpretations and what can be deduced for typed versions of the same ς-terms, we consider a type assignment version of Abadi and Cardelli system $\mathrm{OB}_{1<:}$, which still we call $\mathrm{OB}_{1<:}$.

To make reading more comfortable, we report the definition of system $\mathrm{OB}_{1<:}$ in figures 3 and 4 (we omit both rules for kinds and premises concerning the assumption that types and contexts are well formed, being first order types easily defined by a grammar and supposing that in a context each term variable occurs at most once). In the examples below we add term constants to make reading easier: they are typed according to some obvious rules, which we collectively name (*Const*) in both systems ($\mathrm{OB}_{1<:}$ and ours).

Self-application interpretation has been introduced in definition 3 of section 3. In [1] the criticism to this interpretation is that it is unsuitable w.r.t. subtyping, because the abstractions in front of method bodies makes the type of the interpretation of an object term contravariant in the self type.

Let us consider the ς-term:

$$a_2 \stackrel{\Delta}{=} [l_1 = \varsigma(x)3, l_2 = \varsigma(x)x.l_1 \Leftarrow \varsigma(y)x.l_1 + 1].$$

In system $\mathrm{OB}_{1<:}$ it can be typed by $\sigma \stackrel{\Delta}{=} [l_1 : int, l_2 : []]$:

$$\dfrac{\dfrac{}{x : \sigma \vdash 3 : int} \text{ (Const)} \qquad \dfrac{\dfrac{}{x : \sigma \vdash x : \sigma} \text{ (Var)} \qquad \dfrac{\dfrac{\dfrac{}{x : \sigma, y : \sigma \vdash x : \sigma} \text{ (Var)}}{x : \sigma, y : \sigma \vdash x.l_1 : int} \text{ (Val Select)} \qquad \dfrac{}{x : \sigma, y : \sigma \vdash 1 : int} \text{ (Const)}}{x : \sigma, y : \sigma \vdash x.l_1 + 1 : int} \text{ (Val Update)}}{x : \sigma \vdash x.l_1 \Leftarrow \varsigma(y)x.l_1 + 1 : \sigma} \text{ (Val Sub)}}{x : \sigma \vdash x.l_1 \Leftarrow \varsigma(y)x.l_1 + 1 : []} \text{ (Type Object)}$$

$$\vdash a_2 : \sigma$$

$$\frac{}{E, x : A \vdash x : A}\ (\text{Var}) \qquad \frac{E, x_i : [l_i : B_i\ ^{i\in I}] \vdash b_i : B_i \quad \forall i \in I}{E \vdash [l_i = \varsigma(x_i)b_i\ ^{i\in I}] : [l_i : B_i\ ^{i\in I}]}\ (\text{Type Object})$$

$$\frac{E \vdash a : [l_i : B_i\ ^{i\in I}] \quad j \in I}{E \vdash a.l_j : B_j}\ (\text{Val Select}) \qquad \frac{\begin{array}{c}A \equiv [l_i : B_i\ ^{i\in I}] \\ E \vdash a : A \quad E, y : A \vdash b : B_j \quad j \in I\end{array}}{E \vdash a.l_j \Leftarrow \varsigma(y)b : A}\ (\text{Val Update})$$

$$\frac{E \vdash a : A \quad E \vdash A <: B}{E \vdash a : B}\ (\text{Val Sub})$$

Fig. 4. Typing rules of type assignment system $\mathrm{OB}_{1<}$.

Its interpretation is

$$[\![a_2]\!]^S \triangleq \langle l_1 = \lambda x.3, l_2 = \lambda x.x \cdot l_1 := \lambda y.(x \cdot l_1)x + 1\rangle$$

In CDV^R_ω we may assign to $[\![a_2]\!]^S$ the type $\sigma_1 \triangleq \langle l_1 : \omega \to int, l_2 : \omega\rangle$, which is close to the original type of a_2 in $\mathrm{OB}_{1<}$; but we can also deduce

$$\sigma_2 \triangleq \langle l_1 : \sigma_1 \to int, l_2 : \sigma_1 \to \sigma_1\rangle.$$

In fact:

$$\dfrac{\dfrac{\dfrac{}{x : \sigma_1 \vdash 3 : int}\ (\text{Const})}{\vdash \lambda x.3 : \sigma_1 \to int}\ (\to\text{I}) \quad \dfrac{\dfrac{x : \sigma_1 \vdash \lambda y.(x \cdot l_1)x + 1 : \omega \to int}{x : \sigma_1 \vdash x \cdot l_1 := \lambda y.(x \cdot l_1)x + 1 : \sigma_1}\ (\langle\rangle\text{U})}{\vdash \lambda x.x \cdot l_1 := \lambda y.(x \cdot l_1)x + 1 : \sigma_1 \to \sigma_1}\ (\to\text{I})}{\vdash [\![a_2]\!]^S : \sigma_2}\ (\langle\rangle\text{I})$$

since

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x : \sigma_1, y : \omega \vdash x : \sigma}\ (\text{Var})}{x : \sigma_1, y : \omega \vdash x \cdot l_1 : \omega \to int}\ (\langle\rangle\text{E}) \quad \dfrac{}{x : \sigma_1, y : \omega \vdash x : \omega}\ (\omega)}{x : \sigma_1, y : \omega \vdash (x \cdot l_1)x : int}\ (\to\text{E})}{x : \sigma_1, y : \omega \vdash (x \cdot l_1)x + 1 : int}\ (\text{Const})}{x : \sigma_1 \vdash \lambda y.(x \cdot l_1)x + 1 : \omega \to int}\ (\to\text{I})$$

These types are not that different from those which are derivable for a_2 in $\mathrm{OB}_{1<}$; moreover the occurrence of ω seems to be connected to their recursive nature. But $[\![a_2]\!]^S$ is a normal form (and a value): by analogy with untyped λ-calculus and the characterization of strongly normalizing terms in system CDV (see [14, 22]), we expect that it should be typable without any occurrence of ω, both in the conclusion and in the derivation. This is actually the case. Let τ, ρ be any types (possibly type variables) without occurrences of ω; define

$$\sigma_3 \triangleq \langle l_1 : \tau \to int, l_2 : \rho\rangle.$$

Then:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x:\sigma_3\wedge\tau, y:\tau\vdash x:\sigma_3\wedge\tau}\,(\mathrm{Var})}{x:\sigma_3\wedge\tau, y:\tau\vdash x:\sigma_3}\,(\wedge\mathrm{E})}{x:\sigma_3\wedge\tau, y:\tau\vdash x\cdot l_1:\tau\to int}\,(\langle\rangle\mathrm{E}) \quad \dfrac{\dfrac{}{x:\sigma_3\wedge\tau, y:\tau\vdash x:\sigma_3\wedge\tau}\,(\mathrm{Var})}{x:\sigma_3\wedge\tau, y:\tau\vdash x:\tau}\,(\wedge\mathrm{E})}{x:\sigma_3\wedge\tau, y:\tau\vdash (x\cdot l_1)x:int}\,(\to\mathrm{E})}{x:\sigma_3\wedge\tau, y:\tau\vdash (x\cdot l_1)x+1:int}\,(\mathrm{Const})}{x:\sigma_3\wedge\tau\vdash \lambda y.(x\cdot l_1)x+1:\tau\to int}\,(\to\mathrm{I})$$

Therefore, writing $N \stackrel{\Delta}{=} (x\cdot l_1)x+1$, we have

$$\dfrac{\dfrac{\dfrac{}{x:\tau\vdash 3:int}\,(\mathrm{Const})}{\vdash \lambda x.3:\tau\to int}\,(\to\mathrm{I}) \quad \dfrac{\dfrac{\dfrac{\dfrac{}{x:\sigma_3\wedge\tau\vdash x:\sigma_3\wedge\tau}\,(\mathrm{Var})}{x:\sigma_3\wedge\tau\vdash x:\sigma_3}\,(\wedge\mathrm{E}) \quad x:\sigma_3\wedge\tau\vdash \lambda y.N:\tau\to int}{x:\sigma_3\wedge\tau\vdash x\cdot l_1:=\lambda y.N:\sigma_3}\,(\langle\rangle\mathrm{U})}{\vdash \lambda x.x\cdot l_1:=\lambda y.N:\sigma_3\wedge\tau\to\sigma_3}\,(\to\mathrm{I})}{\vdash [\![a_2]\!]^S:\langle l_1:\tau\to int, l_2:\sigma_3\wedge\tau\to\sigma_3\rangle}\,(\langle\rangle\mathrm{I})$$

As a matter of fact we conjecture the stronger statement: if CDV^R is obtained from CDV^R_ω by deleting ω from the type definition, and eliminating rule (ω) from the system, then $M\in\Lambda_R$ is typable in CDV^R if and only if it is strongly normalizing. If the full reduction of ς-calculus is considered, we also conjecture that any term $[\![a]\!]^S$, such that a is typable in $\mathrm{OB}_{1<}$, is typable in system CDV^R if and only if a is strongly normalizable in the ς-calculus.

As the last example shows, the interpretation of the self-variable can be typed in a non uniform way. Indeed $[\![[l_i=\varsigma(x_i)b_i\ ^{i\in I}]]\!]^S$ is typed (among many other possibilities) by $\langle l_i:\sigma_i\to\tau_i\ ^{i\in I}\rangle$, for some σ_i,τ_i, where the σ_i are not necessarily equal.

This, which surely sounds odd to those familiar with typings of object calculi, is sound in our perspective: in fact in the derivation of the type of object interpretations the judgment $x_i:\sigma_i$ does not mean "the type of this object is σ_i", being the type we derive just a predicate of records. It is indeed clear that the notion of self is not immediately translated into the interpretation of object terms, rather it is implicit in the translation of method invocation.

We only observe that it is possible to collect all the assumptions made about the self-variable into a uniform typing: indeed any derivation of $[\![[l_i=\varsigma(x_i)b_i\ ^{i\in I}]]\!]^S:\langle l_i:\sigma_i\to\tau_i\ ^{i\in I}\rangle$ can be transformed into a derivation of $[\![[l_i=\varsigma(x_i)b_i\ ^{i\in I}]]\!]^S:\langle l_i:\bigwedge_{j\in I}\sigma_j\to\tau_i\ ^{i\in I}\rangle$.

6 The characterization theorem

In this section we provide a characterization of convergent λ-terms with records using the type assignment system of section 4. Combining this with theorem 1, we obtain a characterization of convergent ς-terms, which is the main result of

the paper. Henceforth by types we mean intersection types for Λ_R. This characterization has a strict analogy with the characterization of those terms from the (classical) λ-calculus which are reducible to some head normal form (see e.g. [22]).

A type is *trivial* if its interpretation is Λ_R^0 for any $\mathcal{I}$: then a trivial type is either ω or an intersection of trivial types. A subset $X \subseteq \Lambda_R^0$ is *saturated* if it is closed under closed expansions (M is a closed expansion of N if $M \longrightarrow_w N$ and $M \in \Lambda_R^0$); we also say that $\mathcal{I}$ is a *saturated interpretation* if $\mathcal{I}(\alpha)$ is a saturated set, for all type variable α. A straightforward induction shows that, if $\mathcal{I}$ is saturated then $[\![\sigma]\!]_{\mathcal{I}}$ is saturated, for all σ.

A *closed substitution* is some mapping $\vartheta : TermVar \to \Lambda_R^0$; $M\vartheta$ denotes the result of substituting all free occurrences of x in M by $\vartheta(x)$. We say that ϑ *respects* $\Gamma,\mathcal{I}$ if for all $x : \sigma \in \Gamma$ it is the case that $\vartheta(x) \in [\![\sigma]\!]_{\mathcal{I}}$. Observe that, if ϑ respects $\Gamma,\mathcal{I}$ then $[\![\sigma]\!]_{\mathcal{I}} \neq \emptyset$, for all σ occurring in Γ.

Lemma 2 (Soundness of Type Interpretation). *Let $\Gamma \vdash_{CDV^R_\omega} M : \tau$ and suppose that $\mathcal{I}$ is a saturated interpretation. If ϑ is some closed substitution respecting $\Gamma,\mathcal{I}$, then $M\vartheta \in [\![\tau]\!]_{\mathcal{I}}$.*

Proof. By induction over the derivation of $\Gamma \vdash M : \tau$. Cases *(Var)* and (ω) are immediate by the hypothesis. Cases $(\wedge I)$, $(\wedge E)$ and $(\to E)$ follow by induction hypothesis. The fact that the interpretation of some types may be empty is relevant just in case the derivation ends with an application of rule $(\to I)$.

Case $(\to I)$: the derivation ends by

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}\ (\to I)$$

Clearly $(\lambda x.M)\vartheta \Downarrow (\lambda x.M)\vartheta$. If $[\![\sigma]\!]_{\mathcal{I}} = \emptyset$, then $(\lambda x.M)\vartheta \in [\![\sigma \to \tau]\!]_{\mathcal{I}}$ vacuously. Else, for any $N \in [\![\sigma]\!]_{\mathcal{I}}$ let ϑ' be such that $\vartheta'(x) = N$, $\vartheta'(y) = \vartheta(y)$, if $y \not\equiv x$. Since ϑ' respects $\Gamma, x : \sigma,\mathcal{I}$, by induction we have $M\vartheta' \equiv (M[N/x])\vartheta \in [\![\tau]\!]_{\mathcal{I}}$; now $((\lambda x.M)N)\vartheta \longrightarrow_w (M[N/x])\vartheta$, and the thesis follows being $[\![\tau]\!]_{\mathcal{I}}$ a saturated set.

Case $(\langle\rangle I)$: the derivation ends by

$$\frac{\Gamma \vdash M_i : \sigma_i \quad \forall i \in I \quad J \subseteq I}{\Gamma \vdash \langle l_i = M_i\ ^{i\in I}\rangle : \langle l_j : \sigma_j\ ^{j\in J}\rangle}\ (\langle\rangle I)$$

Since $\langle l_i = M_i\ ^{i\in I}\rangle\vartheta \equiv \langle l_i = M_i\vartheta\rangle$, we have, for all $j \in J \subseteq I$,

$$\langle l_i = M_i\vartheta\ ^{i\in I}\rangle \cdot l_j \longrightarrow_w M_j\vartheta \in [\![\sigma_j]\!]_{\mathcal{I}}$$

by induction; the thesis now follows since $[\![\sigma_j]\!]_{\mathcal{I}}$ is saturated.

Case $(\langle\rangle E)$: the derivation ends by

$$\frac{\Gamma \vdash M : \langle l_i : \sigma_i\ ^{i\in I}\rangle \quad j \in I}{\Gamma \vdash M \cdot l_j : \sigma_j}\ (\langle\rangle E)$$

By induction $M\vartheta \in [\![\langle l_i : \sigma_i \ ^{i\in I}\rangle]\!]_{\mathcal{I}}$, hence for some $R \in \mathcal{R}$, $M\vartheta \Downarrow R$ and $R\cdot l_i \in [\![\sigma_i]\!]_{\mathcal{I}}$, for all $i \in I$. $M\vartheta \xrightarrow{*}_w R$ implies $M\vartheta\cdot l_i \xrightarrow{*}_w R\cdot l_i$, and therefore $M\vartheta\cdot l_i \in [\![\sigma_i]\!]_{\mathcal{I}}$, being $[\![\sigma_i]\!]_{\mathcal{I}}$ saturated.

Case $(\langle\rangle U)$: the derivation ends by

$$\frac{\Gamma \vdash M : \langle l_i : \sigma_i \ ^{i\in I}\rangle \quad \Gamma \vdash L : \sigma \quad j \in I}{\Gamma \vdash M\cdot l_j := L : \langle l_i : \sigma_i \ ^{i\in I\setminus\{j\}}, l_j : \sigma\rangle}\ (\langle\rangle U)$$

By induction there exists some $R \in \mathcal{R}$ such that $M\vartheta \xrightarrow{*}_w R$ and $R\cdot l_i \in [\![\sigma_i]\!]_{\mathcal{I}}$, for all $i \in I$. By definition R has the shape $\langle l_i : M_i \ ^{i\in I}\rangle$; therefore

$$\begin{array}{rcl}(M\cdot l_j := L)\vartheta & \equiv & M\vartheta\cdot l_j := L\vartheta \\ & \xrightarrow{*}_w & \langle l_i : M_i \ ^{i\in I}\rangle\cdot l_j := L\vartheta \\ & \longrightarrow_w & \langle l_i : M_i \ ^{i\in I\setminus\{j\}}, l_j = L\vartheta\rangle\end{array}$$

and $\langle l_i : M_i \ ^{i\in I\setminus\{j\}}, l_j = L\vartheta\rangle \in [\![\langle l_i : \sigma_i \ ^{i\in I\setminus\{j\}}, l_j : \sigma\rangle]\!]_{\mathcal{I}}$ by the above and the induction hypothesis. The thesis follows since $[\![\langle l_i : \sigma_i \ ^{i\in I\setminus\{j\}}, l_j : \sigma\rangle]\!]_{\mathcal{I}}$ is saturated.

□

Given the soundness of type interpretation we use it, together with type invariance under reduction and expansion, to characterize convergent λ-terms with records:

Theorem 3. *For any closed term M, $M \Downarrow$ if and only if it is typable by some non trivial type in CDV^R_ω; moreover $M \Downarrow F$ for some $F \in \mathcal{F}$ if and only if M is typable by $\omega \to \omega$, and $M \Downarrow R$, for some $R \in \mathcal{R}$, if and only if M is typable by $\langle l_i : \omega \ ^{i\in I}\rangle$, for some I.*

Proof. The only if part follows by theorem 2 (ii) and the fact that $\vdash_{CDV^R_\omega} \lambda x.M : \omega \to \omega$ and $\vdash_{CDV^R_\omega} \langle l_i = M_i \ ^{i\in i}\rangle : \langle l_i : \omega \ ^{i\in I}\rangle$, for all $\lambda x.M \in \mathcal{F}$ and $\langle l_i = M_i \ ^{i\in i}\rangle \in \mathcal{R}$. The if part is consequence of lemma 2.

□

A further consequence of this theorem is that terms reducing to a selection $M\cdot l$ or an update $M\cdot l := N$ over some label l which is undefined in M have only trivial types; in particular ill-formed terms like $\mathbf{I}\cdot l$ are only typable by conjunctions of ω.

We are eventually in place to state the main result of the paper.

Corollary 1. *For all pure (i.e. constant free) untyped ς-term a, $a \downarrow$ if and only if $[\![a]\!]^S \Downarrow$ if and only if, for some Γ and l, $\Gamma \vdash_{CDV^R_\omega} [\![a]\!]^S : \langle l : \omega\rangle$.*

Proof. By theorems 1 and 3.

□

7 Conclusion and further work

We have shown that a piece of theory of type assignment nicely yields a characterization of convergent ς-terms, up to the modest overhead of self-application interpretation. But it seems that we have just scratched the surface of a subject which deserves further investigation.

First, a suitable extension of the notion of saturated sets should give the tool to settle the conjecture in section 5 that exactly the interpretations of strongly normalizing objects (w.r.t. the full reduction relation) are typable in system CDV^R. In the same vein one may also consider the problem of characterizing other properties of reduction in object calculi that have been studied for the λ-calculus [15].

A further step is to build filter models of object calculi using Λ_R and its typings as an auxiliary tool. This opens the question of the structure of the model, namely its theory; conversely one may investigate whether, given a theory such as bisimulation theory of objects [19], a filter model can be devised such that the theory is complete w.r.t. that model.

An obvious task is investigation of subtyping: if we consider the containment induced by type interpretation in section 4, this is subtyping in depth and width; but a simple and direct correspondence with subtyping in object calculi is unlikely. If instead of containment semantics one consider the coercion semantics of subtyping (see e.g. [23], chapter 10), however, our framework looks more promising: it is also tempting to consider the retraction as types proposal by Scott [27], and see what happens.

Finally, looking for some practical application, it should not be difficult to find out a type reconstruction method based on the notion of principal types, even if, of course, the typability of normalizing objects is undecidable in our system. Also it is worthy to see whether certain abstract interpretation and static analysis techniques based on type systems (see e.g. [24, 13, 11]) carry over to object calculi using our approach.

Acknowledgment

We wish to thank Mariangiola Dezani for reading a preliminary version of this paper, and Viviana Bono for representing the point of view of object oriented people. The final version of the paper profits of the careful reading and detailed comments of two anonymous referees.

References

1. M. Abadi, L. Cardelli, *A Theory of Objects*, Springer 1996.
2. M. Abadi, L. Cardelli, R. Viswanathan, "An interpretation of objects and object types" *Proc. of of POPL'96* 1996, 396-409.
3. S. Abramsky, "Domain Theory in Logical Form", *APAL* 51, 1991, 1-77.
4. S. Abramsky, C.-H. L. Ong "Full abstraction in the lazy lambda calculus", *Inf. Comput.* 105(2), 1993, 159-267.

5. S. van Bakel, *Intersection Type Disciplines in Lambda Calculus and Applicative Term Rewriting*, Ph.D. Thesis, Mathematisch Centrum, Amsterdam 1993.
6. H.P. Barendregt, M. Coppo, M. Dezani, "A Filter Lambda Model and the Completeness of Type Assignment", *JSL* 48, 1983, 931-940.
7. G. Boudol, "A Lambda Calculus for (Strict) Parallel Functions", *Info. Comp.* 108, 1994, 51-127.
8. K.B. Bruce, "A paradigmatic Object-Oriented design, static typing and semantics", *J. of Fun. Prog.* 1(4), 1994, 127-206.
9. K.B. Bruce, L. Cardelli, B.C. Pierce, "Comparing object encodings", *Proc. of TACS'97*, *LNCS* 1281, 1997, 415-438.
10. K. Crary, "Simple, Efficient Object Encoding using Intersection Types", CMU Technical Report CMU-CS-99-100.
11. M. Coppo, F. Damiani, P. Giannini, "Inference based analyses of functional programs: dead-code and strictness", in [28], 143-176.
12. M. Coppo, M. Dezani, B. Venneri, "Functional characters of solvable terms", *Grund. der Math.*, 27, 1981, 45-58.
13. M. Coppo, A. Ferrari, "Type inference, abstract interpretation and strictness analysis", *TCS* 121, 1993, 113-144.
14. M. Dezani, E. Giovannetti, U. de' Liguoro, "Intersection types, λ-models and Böhm trees", in [28], 45-97.
15. M. Dezani, F. Honsell, Y. Motohama, "Compositional Characterizations of λ-terms using Intersection Types", *Proc. of MFCS'00*, *LNCS* 1893, 2000, 304-313.
16. M. Dezani, U. de' Liguoro, A. Piperno. "A filter model for concurrent lambda-calculus", *Siam J. Comput.* 27(5), 1998,1376-1419.
17. L. Egidi, F. Honsell, S. Ronchi della Rocca, "Operational, denotational and logical descriptions: a case study", *Fund. Inf.* 16, 1992, 149-169.
18. K. Fisher, F. Honsell, J.C. Mitchell, "A lambda calculus of objects and method specialization", *Nordic J. Comput.* 1, 1994, 3-37.
19. A. Gordon, G. Rees, "Bisimilarity for first-order calculus of objects with subtyping", *Proc. of POPL'96*, 1996, 386-395.
20. H. Ishihara, T. Kurata, "Completeness of intersection and union type assignment systems for call-by-value λ-models", to appear in *TCS*.
21. S. Kamin, "Inheritance in Smalltalk-80: a denotational definition", *Proc. of POPL'88*, 1988, 80-87.
22. J.L. Krivine, *Lambda-calcul, types et modèles*, Masson 1990.
23. J.C. Mitchell, *Foundations for Programming Languages*, MIT Press, 1996.
24. B. Pierce, *Programming with Intersection Types and Bounded Polymorphism*, Ph.D. Thesis, 1991.
25. G. Pottinger, "A Type Assignment System for Strongly Normalizable λ-terms", in R. Hindley, J. Seldin eds., *To H.B. Curry, Essays on Combinatory Logic, Lambda Calculus and Formalisms*, Academic Press, 1980, 561-527.
26. J. Reynolds, "The Coherence of Languages with Intersection Types", *LNCS* 526, 1991, 675-700.
27. D. Scott, "Data types as lattices", *SIAM J. Comput.* 5, n. 3, 1976, 522-587.
28. M. Takahashi, M. Okada, M. Dezani eds., *Theories of Types and Proofs*, Mathematical Society of Japan, vol. 2, 1998.