Restricted intersection type assignment systems and object properties

Ugo de'Liguoro *

Università di Torino, Dipartimento di Informatica
C.so Svizzera 185, 10149 Torino (Italy)
e-mail: deligu@di.unito.it

December 2002

Abstract

In this note we consider a restricted version of the intersection types for a λ-calculus with records as presented in [5, 6] w.r.t. principal typing property and expressivity. We sketch how the classical approach to principal typing for intersection type assignment system can be adapted to cope with record types. We then exemplify typings in our system of self-application and recursive record interpretations of objects.

1 Introduction

When object-oriented languages are concerned, types are always meant to be polymorphic. Here the basic source of polymorphism is subtyping, either linked to inheritance or not. Further the presence of the self parameter naturally leads to some form of quantification over types which shows up in the use of recursive types, universal and existential types, several forms of bounded quantifications.

Polymorphic types are expressive and powerful tools in the theoretical investigation, but are quite cumbersome to use in practice. On the other hand they are not feasible for an automatic treatment as they do not enjoy relevant properties as the existence of principal types/principal typings and of related algorithms. This motivates the search for systems of polymorphic types strong enough to catch relevant properties of programs, but still allowing for automatic search and reconstruction of types.

The approach we follow in this note is a development of what we propose in [5, 6], which is basically to reconstruct typings from the semantics of object-calculi, which is given in terms of interpretations in a type free λ-calculus with

*Partially supported by MURST Cofin'01 COMETA Project, IST-2001-33477 DART Project and IST-2001-322222 MIKADO Project. The funding bodies are not responsible for any use that might be made of the results presented here.

records. More precisely types are seen as term properties, so that they are formalized using extensions of Curry assignment systems known as intersection types (see e.g. [7] and references there).

2 Intersection type assignment systems for a λ-calculus with records

Define a type-free λ-calculus with records as follows:

the set of terms Λ_R is inductively defined by:

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

where I is a finite set of indexes, $\langle \ell_i = M_i \ ^{i\in I} \rangle$ is a record, $M\cdot \ell_i$ is field selection, $M\cdot \ell_i := N$ either adds a new field ℓ_i initializing its value to N, or updates to N the value of an existing field ℓ_i in M;

the intended meaning of record operators is formalized by adding to the definition of β-reduction the following rules:

$(R1)$ $\langle \ell_i = M_i \ ^{i\in I} \rangle \cdot \ell_j \longrightarrow M_j$, if $j \in I$,
$(R2)$ $M \longrightarrow M' \ \Rightarrow \ M\cdot l_i \longrightarrow M'\cdot l_i$,
$(R3)$ $\langle \ell_i = M_i \ ^{i\in I} \rangle \cdot \ell_j := N \longrightarrow \langle \ell_i = M_i \ ^{i\in I\setminus\{j\}}, \ell_j = N \rangle$,
$(R4)$ $M \longrightarrow M' \ \Rightarrow \ M\cdot \ell_i := N \longrightarrow M'\cdot \ell_i := N$.

Intersection types plus record types are defined according to the grammar:

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

where α ranges over a denumerable set of type variables, ω is a constant, I ranges over finite sets of indexes. Then it is easy to extend intersection type systems to the λ-calculus with records. Here is the basic system:

$$\frac{}{\Gamma \vdash M : \omega}\ (\omega) \qquad \frac{}{\Gamma, x : \sigma \vdash x : \sigma}\ (Var)$$

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

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

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

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

This is the system considered in [5] but for rule ($\langle\rangle U$), which has there the restriction $j \in I$ because the meaning of $M \cdot \ell_j := N$ was field update only. With slight changes in the proof we can establish the same result as in [5], namely that a term is typeable by any σ essentially different from ω if and only if it has a (weak) normal form under lazy reduction:

Theorem 2.1 *Any closed term $M \in \Lambda_R$ reduces to a (weak) normal form if and only if it is typable by some non trivial type; moreover M reduces to an abstraction if and only if M is typable by $\omega \to \omega$; M reduces to a record if and only if M is typable by $\langle \ell_i : \omega \ ^{i\in I}\rangle$, for some I.*

The present system is an extension of that named CDV in [11]. Let $\leq$ be the least preorder over types including the subtyping relation of [3] and such that (writing $\sigma = \tau$ for $\sigma \leq \tau \leq \sigma$):

i) $I \supseteq J \Rightarrow \langle \ell_i : \sigma_i \ ^{i\in I}\rangle \leq \langle \ell_j : \sigma_j \ ^{j\in J}\rangle$;

ii) $\forall i \in I.\ \sigma_i \leq \tau_i \Rightarrow \langle \ell_i : \sigma_i \ ^{i\in I}\rangle \leq \langle \ell_i : \tau_i \ ^{i\in I}\rangle$;

iii) $\langle \ell_i : \sigma_i \ ^{i\in I}\rangle = \bigwedge_{i\in I} \langle \ell_i : \sigma_i\rangle$;

iv) $\langle \ell_i : \sigma\rangle \wedge \langle \ell_i : \tau\rangle = \langle \ell_i : \sigma \wedge \tau\rangle$.

Then the type assignment system for Λ_R can be presented in the same style as the BCD system [3, 11] by adding the subsumption rule:

$$\frac{\Gamma \vdash M : \sigma \quad \sigma \leq \tau}{\Gamma \vdash M : \tau}\ (Sub)$$

(Now rule ($\wedge I$) becomes redundant). We call CDV^R and BCD^R the extensions with record types and record subtyping of CDV and of BCD systems respectively.

3 Principal typings

According to [12], the problem of *principal typings* is the following: given a term M find a pair (Γ, σ) (if any) such that the judgment $\Gamma \vdash M : \sigma$ represents all possible typings for M in a given system. A system $\mathcal{S}$ has the *principal typings property* if a principal typing exists for any term w.r.t. $\mathcal{S}$.

Terms of the shape $\langle \ell_i = M_i \ ^{i\in I}\rangle$ are finite sets of fields; this causes technical difficulties in the search of a typing so that we adopt a slightly different syntax (see [10]):

$$P, Q ::= x \mid \lambda x.P \mid PQ \mid emp \mid (sel_i\ P) \mid (lcond_i\ P\ Q)$$

where i ranges over the same set of indexes used for labels ℓ_i. The intended meaning of the new constants (which is the empty record, selection and field add/update respectively) can be expressed via the rewriting rules:

a) $(sel_i\ (lcond_i\ P\ Q)) \longrightarrow Q$

b) $(sel_i\ (lcond_j\ P\ Q)) \longrightarrow (sel_i\ P)$, if $i \neq j$.

There is an obvious translation $[\cdot]^*$ from Λ_R to the new syntax, which is the compatible closure of the clauses:

1. $[\langle\rangle]^* \equiv emp$ (where $\langle\rangle \equiv \langle \ell_i = M_i\ ^{i\in\emptyset}\rangle$);
2. $[\langle \ell_i = M_i\ ^{i\in I}\rangle]^* \equiv (lcond_{i_k}(\ldots(lcond_{i_0} emp\ [M]^*_{i_0})\ldots)[M]^*_{i_k})$, where $I = \{i_0, \ldots, i_k\}$ (we suppose that the set of all indexes is linearly ordered);
3. $[M \cdot \ell_i]^* \equiv (sel_i\ [M]^*)$;
4. $[M \cdot \ell_i := N]^* \equiv (lcond_i\ [M]^*\ [N]^*)$.

The basic system CDV^R is then adapted to the new syntax by replacing rules $(\langle\rangle I)$, $(\langle\rangle E)$ and $(\langle\rangle U)$ by the following axiom schemata:

$$\frac{}{\Gamma \vdash emp : \langle\rangle}\ (emp)$$

$$\frac{\Gamma \vdash P : \langle \ell_j : \sigma_j\ ^{j\in J}\rangle \quad i \in J}{\Gamma \vdash (sel_i\ P) : \sigma_i}\ (sel)$$

$$\frac{\Gamma \vdash P : \langle \ell_j : \sigma_j\ ^{j\in J}\rangle \quad \Gamma \vdash Q : \tau}{\Gamma \vdash (lcond_i\ P\ Q) : \langle \ell_j : \sigma_j\ ^{j\in J\setminus\{i\}}, \ell_i : \tau\rangle}\ (lcond)$$

We call $\mathrm{CDV}^{\mathsf{C}}$ the resulting assignment system. The correspondence between the systems is established in the following lemma, whose proof is a simple induction (only if part) and relies on a generation lemma for system $\mathrm{CDV}^{\mathsf{C}}$ which we omit (if part).

Lemma 3.1 *Let $M \in \Lambda_R$: then $\Gamma \vdash M : \sigma$ in system CDV^R if and only if $\Gamma \vdash [M]^* : \sigma$ in system $\mathrm{CDV}^{\mathsf{C}}$.*

Systems CDV and BCD have the principal typing property, where the principal typing (Π, π) "represents" all possible typings of a given term M in the sense that for every typing $\Gamma \vdash M : \sigma$ there exists an operation O such that $O(\Pi, \pi) = (\Gamma, \sigma)$: now O is not just substitution, rather it is a sequence of expansions and substitutions in case of CDV system, a sequence of expansions, substitutions and rise in the case of BCD system (see [11] for details). A key step in the construction is a syntax directed definition of *ground typings* for approximants; the typings are then extended to the general case using the approximation theorem.

To follow this approach we first extend the notion of approximation (see [2] definition 14.3.6):

i) $\alpha(x) = x$

ii) $\alpha((\lambda x.P)Q) = \bot$

iii) $\alpha(sel_i(lcond_j\ P\ Q))) = \bot$

iv) $\alpha(\lambda x.P) = \lambda x.\alpha(P)$

v) $\alpha(PQ) = \alpha(P)\alpha(Q)$, if $P \neq \lambda x.P'$

Then $\omega(P)$ is the $\longrightarrow_{\beta,\bot}$ normal form of $\alpha(P)$, where $\longrightarrow_\bot$ is the compatible closure of the relation defined by:

- $\bot\ P \longrightarrow_\bot \bot$
- $(sel_i\ \bot) \longrightarrow_\bot \bot$
- $(lcond_i\ \bot\ Q) \longrightarrow_\bot \bot$, $(lcond_i\ P\ \bot) \longrightarrow_\bot \bot$

(note that we do not have $\lambda x.\bot \longrightarrow_\bot \bot$). Now we are in place to define inductively a mapping producing the principal typings of approximants (terms of the form $\omega(P)$ for some P):

1. $PP_{CDV^{\mathsf{C}}}(\bot) = (\emptyset, \omega)$
2. $PP_{CDV^{\mathsf{C}}}(emp) = (\emptyset, \langle\rangle)$
3. let $PP_{CDV^{\mathsf{C}}}(A) = (\Pi, \pi)$ and $A \neq \bot$:
   - if $A = x$ then $PP_{CDV^{\mathsf{C}}}((sel_i\ A)) = (\Pi + \{x : \langle \ell_i : \beta\rangle\}, \beta)$
   - else if $\pi = \langle \ell_j : \sigma_j\ ^{j\in J}\rangle$ and $i \in J$ then $PP_{CDV^{\mathsf{C}}}((sel_i\ A)) = (\Pi, \sigma_i)$
   - else $PP_{CDV^{\mathsf{C}}}((sel_i\ A)) = (\emptyset, \omega)$
4. let $PP_{CDV^{\mathsf{C}}}(A_k) = (\Pi_k, \pi_k)$ and $A_k \neq \bot$ for $k = 1, 2$:
   - if $A_1 = x$ then $PP_{CDV^{\mathsf{C}}}((lcond_i\ A_1\ A_2)) = (\Pi_1 + \Pi_2 + \{x : \langle\rangle\}, \langle \ell_i : \pi_2\rangle)$
   - else if $\pi_1 = \langle \ell_j : \sigma_j\ ^{j\in J}\rangle$ then $PP_{CDV^{\mathsf{C}}}((lcond_i\ A_1\ A_2)) = (\Pi_1 + \Pi_2, \langle \ell_j : \sigma_j\ ^{j\in J\setminus\{i\}}, \ell_i : \pi_2\rangle)$
   - else $PP_{CDV^{\mathsf{C}}}((lcond_i\ A_1\ A_2)) = (\emptyset, \omega)$
5. $PP_{CDV^{\mathsf{C}}}(x) = (\{x : \alpha\}, \alpha)$
6. let $PP_{CDV^{\mathsf{C}}}(A) = (\Pi, \pi)$:
   - if $x \in FV(A)$ then $PP_{CDV^{\mathsf{C}}}(\lambda x.A) = (\Pi \setminus \{x : \Pi(x)\}, \Pi(x) \to \pi)$
   - if $x \notin FV(A)$ then $PP_{CDV^{\mathsf{C}}}(\lambda x.A) = (\Pi, \omega \to \pi)$
7. let $PP_{CDV^{\mathsf{C}}}(A_i) = (\Pi_i, \pi_i)$ for $1 \leq i \leq n$ (supposed to be variants disjoint in pairs w.r.t. names of type variables) and α a fresh type variable, then $PP_{CDV^{\mathsf{C}}}(xA_1 \ldots A_n) = (\Pi_1 + \cdots + \Pi_n + \{x : \pi_1 \to \cdots \to \pi_n \to \alpha\}, \alpha)$.

Note that, in particular, $PP_{CDV^{\mathsf{C}}}(\lambda x.\bot) = (\emptyset, \omega \to \omega)$.

A lot of work remains: in particular a rise operation is in order since if $\Gamma \vdash M : \langle \ell_i : \sigma_i\ ^{i\in I}\rangle$ is derivable in CDV^R, then $\Gamma \vdash M : \langle \ell_i : \sigma_i\ ^{i\in I'}\rangle$ is derivable, for all $I' \subseteq I$. Eventually the approximation theorem must be established.

Types:

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

Subtyping rules:

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

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

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

4 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 1 and 2 (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).

4.1 Self-application interpretation of objects

Self-application interpretation has been introduced by [8]. 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 $\sigma \stackrel{\Delta}{=} [l_1 : \mathit{int}, l_2 : []]$:

$$\dfrac{\dfrac{}{x : \sigma \vdash 3 : \mathit{int}} \text{ (Const)} \qquad \dfrac{\dfrac{}{x : \sigma \vdash x : \sigma} \text{ (Var)} \quad \dfrac{\dfrac{\dfrac{}{x : \sigma, y : \sigma \vdash x : \sigma} \text{ (Var)}}{x : \sigma, y : \sigma \vdash x.l_1 : \mathit{int}} \text{ (Val Select)} \quad \dfrac{}{x : \sigma, y : \sigma \vdash x.l_1 + 1 : \mathit{int}} \text{ (Const)}}{x : \sigma \vdash x.l_1 \Leftarrow \varsigma(y)x.l_1 + 1 : \sigma} \text{ (Val Update)}}{x : \sigma \vdash x.l_1 \Leftarrow \varsigma(y)x.l_1 + 1 : []} \text{ (Val Sub)}}{\vdash a_2 : \sigma} \text{ (Type Object)}$$

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

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

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

Figure 2: 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 $\sigma_1 \triangleq \langle l_1 : \omega \to \mathit{int}, l_2 : \omega\rangle$, which is close to the original type of a_2 in $\mathrm{OB}_{1<:}$; but we can also deduce

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

In fact:

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

since

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

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 [7, 9]), 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

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

Then:

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

Therefore, writing $N \triangleq (x\cdot l_1)x+1$, we have

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

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 $[\![[l_i=\varsigma(x_i)b_i\ ^{i\in I}]]\!]^S:\langle l_i:\bigwedge_{j\in I}\sigma_j\to\tau_i\ ^{i\in I}\rangle$.

4.2 The recursive record interpretation of objects

Under recursive-record interpretation, ς-terms are translated as follows:

$$\begin{array}{lcl}
[\![x]\!]^R & \triangleq & x\\
[\![[l_i=\varsigma(x_i)b_i\ ^{i\in 1..k}]]\!]^R & \triangleq & \mathbf{Y}(\lambda x.\langle l_i=[\![b_i]\!]^R[x/x_i]\ ^{i\in 1..k}\rangle)\\
[\![a.l]\!]^R & \triangleq & [\![a]\!]^R\cdot l
\end{array}$$

where $\mathbf{Y} \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ is Curry paradoxical combinator. This interpretation is partial, as there is no clear way to translate overriding.

Let us consider the following untyped ς-term:

$$a_1 \triangleq [l_1 = \varsigma(x)3, l_2 = \varsigma(x)x.l_1].$$

In system OB_1 this term can be typed by $\sigma \triangleq [l_1 : \mathit{int}, l_2 : \mathit{int}]$, with derivation:

$$\dfrac{\dfrac{}{x : \sigma \vdash 3 : \mathit{int}}\ (\text{Const}) \quad \dfrac{\dfrac{}{x : \sigma \vdash x : \sigma}\ (\text{Var})}{x : \sigma \vdash x.l_1 : \mathit{int}}\ (\text{Val Select})}{\vdash a_1 : \sigma}\ (\text{Type Object})$$

Its interpretation in Λ_R is:

$$[\![a_1]\!]^R \triangleq \mathbf{Y}(\lambda x.\langle l_1 = 3, l_2 = x \cdot l_1\rangle)$$

In system CDV^R_ω we give $[\![a_1]\!]^R$ essentially the same type $\langle l_1 : \mathit{int}, l_2 : \mathit{int}\rangle$. First consider the derivation:

$$\dfrac{\dfrac{\dfrac{}{x : \omega \vdash 3 : \mathit{int}}\ (\text{Const}) \quad \dfrac{}{x : \omega \vdash : x \cdot l_1 : \omega}\ (\omega)}{x : \omega \vdash \langle l_1 = 3, l_2 = x \cdot l_1\rangle : \langle l_1 : \mathit{int}, l_2 : \omega\rangle}\ (\langle\rangle\text{I})}{\vdash \lambda x.\langle l_1 = 3, l_2 = x \cdot l_1\rangle : \omega \to \langle l_1 : \mathit{int}, l_2 : \omega\rangle}\ (\to\text{I})$$

Setting $\sigma_1 \triangleq \langle l_1 : \mathit{int}, l_2 : \omega\rangle$, we have that:

$$\dfrac{\dfrac{\dfrac{}{x : \sigma_1 \vdash 3 : \mathit{int}}\ (\text{Const}) \quad \dfrac{\dfrac{}{x : \sigma_1 \vdash x : \sigma_1}\ (\text{Var})}{x : \sigma_1 \vdash x \cdot l_1 : \mathit{int}}\ (\langle\rangle\text{E})}{x : \sigma_1 \vdash \langle l_1 = 3, l_2 = x \cdot l_1\rangle : \langle l_1 : \mathit{int}, l_2 : \mathit{int}\rangle}\ (\langle\rangle\text{I})}{\vdash \lambda x.\langle l_1 = 3, l_2 = x \cdot l_1\rangle : \sigma_1 \to \langle l_1 : \mathit{int}, l_2 : \mathit{int}\rangle}\ (\to\text{I})$$

Combining these two derivations by $(\wedge I)$ rule, we get:

$$\vdash \lambda x.\langle l_1 = 3, l_2 = x \cdot l_1\rangle : (\omega \to \sigma_1) \wedge (\sigma_1 \to \sigma_2), \quad \sigma_2 \triangleq \langle l_1 : \mathit{int}, l_2 : \mathit{int}\rangle.$$

Since the combinator $\mathbf{Y}$ has type $(\omega \to \sigma_1) \wedge (\sigma_1 \to \sigma_2) \to \sigma_2$ (among infinitely many others), we conclude that, by $(\to E)$:

$$\vdash_{CDV^R_\omega} [\![a_1]\!]^R \equiv \mathbf{Y}(\lambda x.\langle l_1 = 3, l_2 = x \cdot l_1\rangle) : \langle l_1 : \mathit{int}, l_2 : \mathit{int}\rangle.$$

References

[1] M. Abadi, L. Cardelli, *A Theory of Objects*, Springer 1996.

[2] H.P. Barendregt, *The Lambda Calculus, Its Syntax and Semantics*, 2nd ed. North-Holland 1984.

[3] H.P. Barendregt, M. Coppo, M. Dezani, "A Filter Lambda Model and the Completeness of Type Assignment", *JSL* 48, 1983, 931-940.

[4] G. Boudol, "The recursive record semantics of object revised", *LNCS* 2028, 2002, 269-283

[5] U. de'Liguoro, "Characterizing convergent terms in object calculi via intersection types", *LNCS* 2044, 2001.

[6] U. de'Liguoro, "Subtyping in logical form", *Proc. of ITRS'02*, *ENTCS* n. 70.

[7] M. Dezani, E. Giovannetti, U. de' Liguoro, "Intersection types, λ-models and Böhm trees", in M. Takahashi, M. Okada, M. Dezani eds., *Theories of Types and Proofs*, Mathematical Society of Japan, vol. 2, 1998., 45-97.

[8] S. Kamin, "Inheritance in Smalltalk-80: a denotational approach", POPL' 88, 1988, 80-87.

[9] J.L. Krivine, *Lambda-calcul, types et modèles*, Masson 1990.

[10] J.C. Mitchell, *Foundations for Programming Languages*, MIT Press, 1996.

[11] S. van Bakel, *Intersection Type Disciplines in Lambda Calculus and Applicative Term Rewriting Systems*, PhD Thesis, University of Nijmegen, 1993.

[12] J. Wells, "The essence of Principal Typings", *LNCS* 2380, 2002, 913-925.