Characterizing convergent terms in object calculi via intersection types

Ugo de'Liguoro
Università di Torino
TLCA'01

Toward logic semantics of objects

- logic semantics amounts to type characterization of relevant computational properties;
- it is a tool for the study of syntactical properties (normalization, strong normalization, standardization);
- it can be used to achieve a concrete description of denotational models (filter models, domain logic);
- it could be a tool to analyze notions of subtyping (like matching) which otherwise have only a syntactical definition (this for the future).

The type-free ς-calculus [AC96]

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

Define the one-step reduction relation by

i) $[l_i = \varsigma(x_i)b_i\ ^{i\in I}].l_j \to_\varsigma b_j\{[l_i = \varsigma(x_i)b_i\ ^{i\in I}]\ /x_j\}$

ii) $[l_i = \varsigma(x_i)a\ ^{i\in I}].l_j \Leftarrow \varsigma(x)b \to_\varsigma [l_i = \varsigma(x_i)a\ ^{i\in I\setminus\{j\}}, l_j = \varsigma(x)b]$

iii) $a \to_\varsigma a' \Rightarrow a.l \to_\varsigma a'.l$

iv) $a \to_\varsigma a' \Rightarrow a\ .l \Leftarrow \varsigma(x)b \to_\varsigma a'.l \Leftarrow \varsigma(x)b$

Defining values as $v ::= [l_i = \varsigma(x_i)b_i\ ^{i\in I}]$ we have

$$a{\downarrow}v \Leftrightarrow a \overset{*}{\to}_\varsigma v$$

Remark: $\to_\varsigma$ is weaker than one-step reduction in [AC96]

Existing type systems in object calculi do not catch termination

In Abadi and Cardelli "Theory of Objects", for any type A, the diverging term:

$$[l = \varsigma(x{:}[l{:}A])x.l].l$$

has type A (so any type has a fully undefined inhabitant).

In terms of type assignment systems this can be restated as:

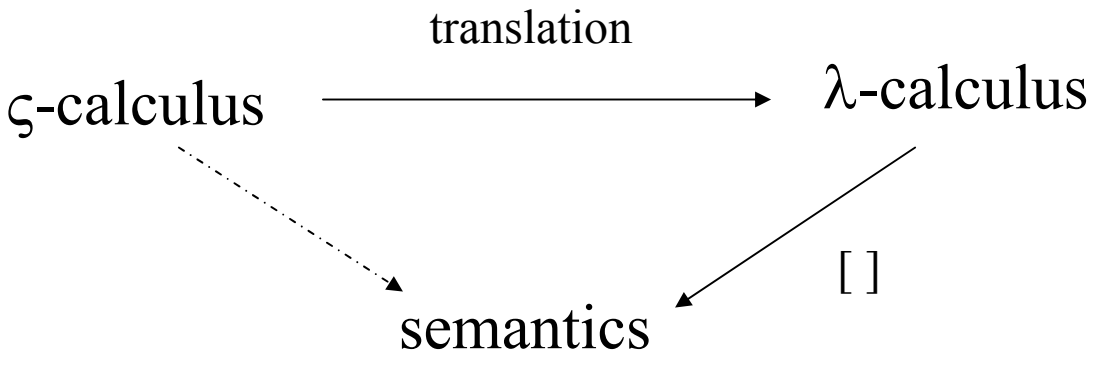
$$[l = \varsigma(x)x.l].l \text{ has any type.}$$

Remark: this is not a criticism, as the same holds for PCF (e.g. fix x.x) or Pascal.

The present approach

• We consider a type-free λ-calculus and an intepretation of (type-free) ς-calculus respecting and preserving convergency;

• we then provide a characterization of convergent λ-terms via an intersection type assignment system.



Λ_R: a λ-calculus with records

record

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

selection

update

(β) $(\lambda x.M)N \to M[N/x]$

(ν) $M \to M' \Rightarrow MN \to M'N$

(R1) $\langle l_i = M_i{}^{i \in I} \rangle \cdot l_j \to M_j$, if $j \in I$

(R2) $M \to M' \Rightarrow M \cdot l \to M' \cdot l$

(R3) $\langle l_i = M_i{}^{i \in I} \rangle \cdot l_j := N \to \langle l_i = M_i{}^{i \in I \setminus \{j\}}, l_j = N \rangle$, if $j \in I$

(R4) $M \to M' \Rightarrow M \cdot l := N \to M' \cdot l := N$

The self-application interpretation

Define (after Kamin 1988 and Abadi Cardelli 1996) a map:

$$[\ ]^S : \varsigma\text{-terms} \to \Lambda_R$$

$$[x]^S = x$$

$$[[l_i = \varsigma(x_i)b_i\ ^{i\in I}]\ ]^S = \langle l_i = \lambda x_i\,.[b_i]^{S\ i\in I}\rangle$$

$$[a.l\ ]^S = ([a]^S \cdot l\ )\ [a]^S$$

$$[a.l \Leftarrow \varsigma(x)b]^S = [a]^S \cdot l := \lambda x.[b]^S$$

Convergency

Let $V ::= \lambda x.M \mid \langle l_i = M_i \,^{i \in I} \rangle$ be the definition of values in the λ-calculus with records; then the *convergency* predicate is defined by:

$$M\Downarrow \Leftrightarrow \exists\, V.\, M \stackrel{*}{\rightarrow} V$$

An irreducible term like $(\lambda x.x) \cdot l$ is a normal form but not a value (hence it "diverges").

Theorem. For any ς-term a it is the case that

$$a\downarrow \Leftrightarrow [a]^S \Downarrow.$$

Goal: give a characterization of the convergency predicate via some type assignment system.

Known results about λ-calculus

Some relevant computational properties of type free λ-terms are characterized via intersection type assignment systems. E.g. in BCD system (with type inequalities and a subsumption rule):

- M is solvable (has hnf) iff $\Gamma \vdash M : \sigma \neq \omega$
- M is normalizable (has nf) iff $\Gamma \vdash M : \sigma$ and $\omega \notin \Gamma, \sigma$
- M is strongly normalizable iff $D: \Gamma \vdash M : \sigma, \omega \notin \Gamma, \sigma, D$

Similar results hold in system CDV_ω (called $D\Omega$ by Krivine), where no type inequalities are considered.

Intersection types for Λ_R

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

To the standard intersection type rules we add rules for records:

(*Rec*)

$$\frac{\Gamma \ \vdash M_i : \sigma_i \quad \forall i \in I \supseteq J}{\Gamma \ \vdash \langle l_i = M_i \ ^{i \in I} \rangle : \langle l_i : \sigma_i \ ^{i \in J} \rangle}$$

(*Sel*)

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

(*Upd*)

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

Comments on (*Upd*) rule

Consider the following variant of (*Upd*), closer to the typing of overriding in the Object Calculus:

same type

(*Upd'*)

$$\frac{\Gamma \mid\!\!- M : \langle l_i : \sigma_i \ ^{i \in I} \rangle \quad j \in I \quad \Gamma \mid\!\!- N : \sigma_j}{\Gamma \mid\!\!- M \cdot l_j := N : \langle l_i : \sigma_i \ ^{i \in I} \rangle}$$

In this way we lose type invariance under expansion:

$$\langle l = \Omega \rangle \cdot l := (\lambda x.x) \ \rightarrow \ \langle l = \lambda x.x \rangle$$

Now $\langle l = \lambda x.x \rangle : \langle l : \sigma \rightarrow \sigma \rangle$

but, using (*Upd'*) in place of (*Upd*), we have at best

$$\langle l = \Omega \rangle \cdot l := (\lambda x.x) : \ \langle l : \omega \rangle$$

Invariance under reduction

The main reason for using intersection types and ω,

$$\frac{\Gamma \mid\!\!- M : \sigma \qquad \Gamma \mid\!\!- M : \tau}{\Gamma \mid\!\!- M : \sigma \wedge \tau}\ (\wedge I) \qquad\qquad \frac{}{\Gamma \mid\!\!- M : \omega}\ (\omega)$$

is type invariance under subject reduction and expansion:

Subject reduction and expansion theorem. In the type system with ω, intersection, arrow and record types:

- if $\Gamma \mid\!\!- M : \sigma$ and $M \to N$ then $\Gamma \mid\!\!- N : \sigma$
- if $\Gamma \mid\!\!- N : \sigma$ and $M \to N$ then $\Gamma \mid\!\!- M : \sigma$

Some typing examples

$$\mathrm{a} = [l = \varsigma(\mathrm{x})3,\ \mathrm{m} = \varsigma(\mathrm{x})\mathrm{x}.l \Leftarrow \varsigma(\mathrm{y})\mathrm{x}.l{+}1] : [l{:}int,\ m{:}[]]$$

translates to

$$M = \langle l = \lambda \mathrm{x}.3,\ \mathrm{m} = \lambda \mathrm{x}.\mathrm{x}\cdot l := \lambda \mathrm{y}.(\mathrm{x}\cdot l)\mathrm{x}{+}1\rangle$$

which is typable by

$$\langle l : \omega \to int,\ m : \omega\rangle$$

which is close to the $\mathrm{Ob}_{1<:}$ type, but also by

$$\langle l : \alpha \to int,\ m : \sigma \wedge \alpha \to \sigma\rangle$$

where $\sigma = \langle l : \alpha \to int,\ m : \beta\rangle$ (α, β type variables), which does not contain ω, as expected being M a normal form.

Type interpretation

Types are interpreted as sets of closed terms. Let us define

$$\mathbf{F} = \{\lambda x.M \mid \lambda x.M \in \Lambda_R{}^0\} \text{ and}$$

$$\mathbf{R} = \{\langle l_i = M_i{}^{i\in I}\rangle \mid \langle l_i = M_i{}^{i\in I}\rangle \in \Lambda_R{}^0\}.$$

Then, given some interpretation $\mathfrak{I}$ of type variables,

$$[\alpha]_{\mathfrak{I}} = \mathfrak{I}(\alpha) \quad [\omega]_{\mathfrak{I}} = \Lambda_R{}^0 \quad [\sigma \wedge \tau]_{\mathfrak{I}} = [\sigma]_{\mathfrak{I}} \cap [\tau]_{\mathfrak{I}}$$

$$[\sigma \to \tau]_{\mathfrak{I}} = \{M \in \Lambda_R{}^0 \mid \exists F \in \mathbf{F}.\ M{\Downarrow}F \text{ and } \forall N \in [\sigma]_{\mathfrak{I}}\ .\ FN \in [\tau]_{\mathfrak{I}}\}$$

$$[\langle l_i\colon \sigma_i{}^{i\in I}\rangle]_{\mathfrak{I}} = \{M \in \Lambda_R{}^0 \mid \exists R \in \mathbf{R}.\ M{\Downarrow}R \text{ and } \forall i \in I\ .\ R \cdot l_i \in [\sigma_i]_{\mathfrak{I}}\}$$

Type inclusion

Define:

$$\sigma \leq \tau \;\Leftrightarrow\; \forall\, \mathfrak{I}.\; [\sigma]_{\mathfrak{I}} \subseteq [\tau]_{\mathfrak{I}}$$

and $\sigma = \tau$ if $\sigma \leq \tau$ and $\sigma \geq \tau$; then:

- $I \supseteq J \Rightarrow \langle l_i : \sigma_i \ ^{i \in I} \rangle \leq \langle l_j : \sigma_i \ ^{j \in J} \rangle$ (sub width)
- $\forall\, i \in I\,.\, \sigma_i \leq \tau_i \Rightarrow \langle l_i : \sigma_i \ ^{i \in I} \rangle \leq \langle l_i : \tau_i \ ^{i \in I} \rangle$ (sub depth)
- $\langle l_i : \sigma_i \ ^{i \in I} \rangle = \Lambda_{i \in I} \langle l_i : \sigma_i \rangle$
- $\langle l : \sigma \rangle \wedge \langle l : \tau \rangle = \langle l : \sigma \wedge \tau \rangle$

Remark. There are *empty* types, e.g. $(\omega \to \omega) \wedge \langle l : \omega \rangle$.

The characterization theorem

If $\mathfrak{I}(\alpha)$ is closed under (closed) expansions for all α, then $[\sigma]_{\mathfrak{I}}$ is such for any σ: we say that $\mathfrak{I}$ is *saturated*.

Soundeness lemma. If $\mathfrak{I}$ is saturated, ϑ: Var $\to \Lambda_R{}^0$ a closed substitution s.t. $\vartheta(x) \in [\sigma]_{\mathfrak{I}}$ for all $x{:}\sigma \in \Gamma$, then

$$\Gamma \mid\!\!- M : \tau \Rightarrow M\vartheta \in [\tau]_{\mathfrak{I}}$$

Characterization theorem. For any $M \in \Lambda_R$:

- $M{\Downarrow}F$, for some $F \in \mathbf{F}$, iff $\Gamma \mid\!\!- M : \omega \to \omega$, for some Γ;
- $M{\Downarrow}R$, for some $R \in \mathbf{R}$, iff $\Gamma \mid\!\!- M : \langle\rangle$, for some Γ.

In particular, for any closed ς-term a:

$$a{\downarrow} \text{ iff } \mid\!\!- [a]^S : \langle\rangle.$$

Conclusions and further work

The present work suggests that, beside PER models and standard domain theoretic techniques like solving domain equations, there is a further way to build and study models of object calculi, namely via type assignment and domain logic: *filter models*.

This remained implicit since a key problem, which is under study, is a reasonable *interpretion of typed terms and equalities*, and a comparison between the (typed) theories one obtains and theories of known models as well as operational theories, like the bisimulation theory of objects by Gordon and Rees.

Further *applications* are expected, essentially extending program logic as it has been developed in the functional case, to object calculi.

Next steps

• Ignoring empty types and considering type inclusion, a filter model comes out, which is a solution of the domain equation

$$D = A + (L \to D) + (D \to D).$$

• Types from typed object calculi may be interpreted as PER over D: is there a "logical" characterization of them?

• Given type interpretation, also typed equations are interpreted, and models can be compared w.r.t. their theories. Interesting would be the study of the bisimulation theory by Gordon and Rees.