Intersection Types for the Computational λ-Calculus

Extended Abstract

Ugo de'Liguoro and Riccardo Treglia

Dipartimento di Informatica, Università degli Studi di Torino, Corso Svizzera 185, 10149 Torino, Italy
ugo.deliguoro@unito.it riccardo.treglia@unito.it

The computational λ-calculus was introduced by Moggi [5,6] as a meta-language to describe non functional effects in programming languages via an incremental approach. The basic idea is to distinguish among values of some type D and computations over such values, the latter having type TD. Semantically T is a monad, endowing D with a richer structure such that operations over computations can be seen as algebras of T. Any D is embedded into TD and there is a universal way to extend any morphism in $D \to TE$ to a morphism in $TD \to TE$.

In Wadler's formulation [7], at the ground of Haskell implementation, a monad is a triple $(T, \mathit{unit}, \star)$ where T is a type constructor, and for all types D, E, $\mathit{unit}_D : D \to TD$ and $\star_{D,E} : TD \times (D \to TE) \to TE$ are such that (omitting subscripts and writing $\star$ as an infix operator):

$$(\mathit{unit}\ d) \star f = f\,d, \qquad a \star \mathit{unit} = a, \qquad (a \star f) \star g = a \star \lambda d.(f\,d \star g).$$

Instances of monads are partiality, exceptions, input/output, store, non determinism, continuations.

Aim of our work is to investigate the monadic approach to effectfull functional languages in the untyped case. Much as the untyped λ-calculus can be seen as a calculus with a single type $D \triangleleft D \to D$, which is interpreted by a reflexive object in a suitable category, the untyped computational λ-calculus λ_c^u has two types: the type of values D and the type of computations TD. The type D is a retract of $D \to TD$, which is the call-by-value analogous of the reflexive object (see [5], sec. 5). This leads to the following definition:

Definition 1 (The untyped computational λ-calculus). *The* untyped computational λ-calculus*, shortly λ_c^u, is a calculus of two sorts of expressions:*

$$\begin{array}{llll} \mathit{Val}: & V, W ::= x \mid \lambda x.M & & \text{(values)} \\ \mathit{Com}: & M, N ::= \mathit{unit}\,V \mid M \star V & & \text{(computations)} \end{array}$$

where x ranges over a denumerable set Var of variables.

A reduction relation $\longrightarrow \subseteq$ *Com* $\times$ *Com is defined as follows:*

$$\begin{array}{rc} (\beta_c) & \mathit{unit}\, V \star (\lambda x.M) \to M[V/x] \\ (\star - \mathit{red}) & M \longrightarrow M' \Rightarrow M \star V \longrightarrow M' \star V \end{array}$$

where $M[V/x]$ *denotes the capture avoiding substitution of* V *for all free occurrences of* x *in* M.

Terms of the calculus can be interpreted into any $D \simeq D \to TD$ (where we restrict to extensional models for simplicity) via the mappings $[\![V]\!]^D_\rho \in D$ and $[\![M]\!]^{TD}_\rho \in TD$, where $\rho \in \mathit{Env}_D = \mathit{Var} \to D$ by:

$$\begin{array}{rlrl} [\![x]\!]^D_\rho &= \rho(x) & [\![\mathit{unit}\, V]\!]^{TD}_\rho &= \mathit{unit}\, [\![V]\!]^D_\rho \\ [\![\lambda x.M]\!]^D_\rho &= \lambda\!\!\!\lambda\, d \in D.\, [\![M]\!]^{TD}_{\rho[x\mapsto d]} & [\![M \star V]\!]^{TD}_\rho &= [\![M]\!]^{TD}_\rho \star [\![V]\!]^D_\rho \end{array}$$

where $\rho[x \mapsto d](y) = \rho(y)$ if $y \not\equiv x$, it is equal to d otherwise. We therefore dub (extensional) T-*model* in a cartesian closed category $\mathcal{D}$ a tuple (D, T, Φ, Ψ) such that T is a monad over $\mathcal{D}$ and $D \simeq D \to TD$ via the morphisms $\Phi, \Psi = \Phi^{-1}$.

Proposition 1. *If* $M \longrightarrow N$ *then* $[\![M]\!]^{TD}_\rho = [\![N]\!]^{TD}_\rho$ *for any* T*-model* D *and* $\rho \in \mathit{Env}_D$.

An intersection type system for λ^u_c

To study T-models we use intersection types, because they are at the same time a formal system to reason on terms and a tool to bridge reduction and operational semantics of the calculus to its models. As shown in [3] reasoning over generic monads is challenging, and indeed a major issue of the present work is to complement Dal Lago's and others contributions by Coppo-Dezani approach to the study of Scott's D_∞ models of the untyped λ-calculus.

Let *TypeVar* be a countable set of type variables, ranged over by α; then we define the following languages of types via the grammar:

$$\begin{array}{rll} \mathit{ValType}: & \delta ::= \alpha \mid \delta \to \tau \mid \delta \wedge \delta \mid \omega_{\mathsf{V}} & (\textit{value types}) \\ \mathit{ComType}: & \tau ::= T\delta \mid \tau \wedge \tau \mid \omega_{\mathsf{C}} & (\textit{computation types}) \end{array}$$

Over types we consider the preorders $\leq_{\mathsf{V}}$ and $\leq_{\mathsf{C}}$ making $\wedge$ into a meet operator and such that:

$$\delta \leq_{\mathsf{V}} \omega_{\mathsf{V}} \qquad (\delta \to \tau) \wedge (\delta \to \tau') \leq_{\mathsf{V}} \delta \to (\tau \wedge \tau') \qquad \frac{\delta' \leq_{\mathsf{V}} \delta \quad \tau \leq_{\mathsf{C}} \tau'}{\delta \to \tau \leq_{\mathsf{V}} \delta' \to \tau'}$$

$$\tau \leq_{\mathsf{C}} \omega_{\mathsf{C}} \qquad T\delta \wedge T\delta' \leq_{\mathsf{C}} T(\delta \wedge \delta') \qquad \frac{\delta \leq_{\mathsf{V}} \delta'}{T\delta \leq_{\mathsf{C}} T\delta'}$$

$$\omega_{\mathsf{V}} \leq_{\mathsf{V}} \omega_{\mathsf{V}} \to \omega_{\mathsf{C}}$$

Now we are ready to define the intersection type assignment for λ^u_c and the generic monad T:

Definition 2 (Type assignment). *A* basis *is a finite set of typings* $\Gamma = \{x_1 : \delta_1, \dots x_n : \delta_n\}$ *with pairwise distinct variables* x_i*, whose* domain *is the set* $\mathrm{dom}\,(\Gamma) = \{x_1, \dots, x_n\}$*. A basis determines a function from variables to types such that* $\Gamma(x) = \delta$ *if* $x : \delta \in \Gamma$*,* $\Gamma(x) = \omega_{\mathsf{V}}$ *otherwise.*

A judgment *is an expression of either shapes* $\Gamma \vdash V : \delta$ *or* $\Gamma \vdash M : \tau$*. It is* derivable *if it is the conclusion of a derivation according to the rules:*

$$\frac{x : \delta \in \Gamma}{\Gamma \vdash x : \delta} \qquad \frac{\Gamma, x : \delta \vdash M : \tau}{\Gamma \vdash \lambda x.M : \delta \to \tau} \qquad \frac{\Gamma \vdash V : \delta}{\Gamma \vdash \mathit{unit}\, V : T\delta} \qquad \frac{\Gamma \vdash M : T\delta \quad \Gamma \vdash V : \delta \to \tau}{\Gamma \vdash M \star V : \tau}$$

where $\Gamma, x : \delta = \Gamma \cup \{x : \delta\}$ *with* $x : \delta \notin \Gamma$*, and the rules:*

$$\frac{}{\Gamma \vdash P : \omega} \qquad \frac{\Gamma \vdash P : \sigma \quad \Gamma \vdash P : \sigma'}{\Gamma \vdash P : \sigma \wedge \sigma'} \qquad \frac{\Gamma \vdash P : \sigma \quad \sigma \leq \sigma'}{\Gamma \vdash P : \sigma'}$$

where either $P \in \mathit{Val}$*,* $\omega \equiv \omega_{\mathsf{V}}$*,* $\sigma, \sigma' \in \mathit{ValType}$ *and* $\leq = \leq_{\mathsf{V}}$ *or* $P \in \mathit{Com}$*,* $\omega \equiv \omega_{\mathsf{C}}$*,* $\sigma, \sigma' \in \mathit{ComType}$ *and* $\leq = \leq_{\mathsf{C}}$*.*

Then by a standard technique, that is by proving suitable Generation and Substitution Lemmas, we establish:

Theorem 1 (Subject reduction). $\Gamma \vdash M : \tau \;\;\&\;\; M \longrightarrow N \Rightarrow \Gamma \vdash N : \tau$*.*

Type assignment and T-models

As a first step we interpret types as certain subsets of D and TD, according to the sorts *ValType* and *ComType* respectively. Let (D, T, Φ, Ψ) be a T-model and $d, d' \in D$; we abbreviate $d \cdot d' = \Phi(d)(d')$. Let $\xi \in \mathit{TypeEnv}_D = \mathit{TypeVar} \to 2^D$; then the followings are natural requirements for the type interpretation mappings $[\![\cdot]\!]^D : \mathit{ValType} \times \mathit{TypeEnv}_D \to 2^D$ and $[\![\cdot]\!]^{TD} : \mathit{ComType} \times \mathit{TypeEnv}_D \to 2^{TD}$:

$$\begin{array}{ll} [\![\alpha]\!]^D_\xi = \xi(\alpha) & [\![\delta \to \tau]\!]^D_\xi = \{d \in D \mid \forall d' \in [\![\delta]\!]^D_\xi \;\; d \cdot d' \in [\![\tau]\!]^{TD}_\xi\} \\ [\![\omega_{\mathsf{V}}]\!]^D_\xi = D & [\![\delta \wedge \delta']\!]^D_\xi = [\![\delta]\!]^D_\xi \cap [\![\delta']\!]^D_\xi \\ [\![\omega_{\mathsf{C}}]\!]^{TD}_\xi = TD & [\![\tau \wedge \tau']\!]^{TD}_\xi = [\![\tau]\!]^{TD}_\xi \cap [\![\tau']\!]^{TD}_\xi \end{array}$$

Further we call these interpretations *monadic* if $[\![T\delta]\!]^{TD}_\xi$ satisfies:

1. $d \in [\![\delta]\!]^D_\xi \Rightarrow \mathit{unit}\, d \in [\![T\delta]\!]^{TD}_\xi$
2. $d \in [\![\delta' \to T\delta]\!]^D_\xi \;\;\&\;\; a \in [\![T\delta']\!]^{TD}_\xi \Rightarrow a \star d \in [\![T\delta]\!]^{TD}_\xi$

The main problem with monadic interpretations is that the clauses above are not inductive, as they would be if we had types $\omega_{\mathsf{V}} =_{\mathsf{V}} \omega_{\mathsf{V}} \to T\omega_{\mathsf{V}}$ and $T\omega_{\mathsf{V}}$ only. However, working in a category of domains and with an ω-continuous monad T we can build a T-model $D_\infty = \lim_{\leftarrow} D_n$, where D_0 is some fixed domain, and $D_{n+1} = [D_n \to TD_n]$ is such that for all n, $D_n \lhd D_{n+1}$ is an embedding. As a consequence we have $D_\infty \simeq [D_\infty \to TD_\infty]$. We say that D_∞ is a *limit* T-model.

More importantly with such a T-model we can stratify the above clauses by means of approximate type interpretations $[\![\delta]\!]^{D_n}_\xi \subseteq D_n$ and $[\![\tau]\!]^{TD_n}_\xi \subseteq TD_n$, that now can be defined by induction over $n \in \mathbb{N}$.

Theorem 2. *The mappings* $[\![\delta]\!]^{D_\infty}_\xi = \lim_{\leftarrow}[\![\delta]\!]^{D_n}_\xi$ *and* $[\![\tau]\!]^{TD_\infty}_\xi = \lim_{\leftarrow}[\![\tau]\!]^{TD_n}_\xi$ *are monadic type interpretations. In particular for any* $\xi \in Env_{D_\infty}$*:*

1. $[\![\delta \to \tau]\!]^{D_\infty}_\xi = \{d \in D_\infty \mid \forall d' \in [\![\delta]\!]^{D_\infty}_\xi \ \ d(d') \in [\![\tau]\!]^{TD_\infty}_\xi\}$
2. $[\![T\delta]\!]^{TD_\infty}_\xi = \begin{array}{l}\{unit\, d \in TD_\infty \mid d \in [\![\delta]\!]^{D_\infty}_\xi\} \cup \\ \{a \star d \in TD_\infty \mid \exists \delta'.\, d \in [\![\delta' \to T\delta]\!]^{D_\infty}_\xi \ \ \& \ \ a \in [\![T\delta']\!]^{TD_\infty}_\xi\}\end{array}$

Now, writing $\rho, \xi \models^D \Gamma$ if $\rho(x) \in [\![\Gamma(x)]\!]^D_\xi$ for all $x \in \mathrm{dom}\,(\Gamma)$, we may set $\Gamma \models^D V : \delta$ ($\Gamma \models^D M : \tau$) if $\rho, \xi \models^D \Gamma$ implies $[\![V]\!]^D_\rho \in [\![\delta]\!]^D_\xi$ ($[\![M]\!]^{TD}_\rho \in [\![\tau]\!]^{TD}_\xi$). Also for any class $\mathcal{C}$ of T-models we write $\Gamma \models^{\mathcal{C}} V : \delta$ ($\Gamma \models M : \tau$) if $\Gamma \models^D V : \delta$ ($\Gamma \models^D M : \tau$) for all $D \in \mathcal{C}$.

Theorem 3 (Soundness). *If* $[\![\delta]\!]^D_\xi$ *and* $[\![\tau]\!]^{TD}_\xi$ *are monadic w.r.t. any* T*-model* $D \in \mathcal{C}$ *then*

$$\Gamma \vdash V : \delta \ \Rightarrow\ \Gamma \models^{\mathcal{C}} V : \delta \quad \textit{and} \quad \Gamma \vdash M : \tau \ \Rightarrow\ \Gamma \models^{\mathcal{C}} M : \tau.$$

In particular, by Theorem 2, we may take $\mathcal{C}$ as the set of limit T-models.

Completeness and computational adequacy

Toward completeness, we first concentrate on the category $\mathcal{D}$ of ω-algebraic lattices, whose objects are known to be presentable as the poset of filters over a meet-semilattice, or equivalently over a preorder whose quotient is such; the ω in the name means that the Scott topology of a domain in $\mathcal{D}$ has a countable basis, formed by the upward cones of compact points. Then any axiomatization $Th = (\mathcal{T}, \leq_{Th})$ of a preorder over a language $\mathcal{T}$ of intersection types making $\wedge$ into the meet and ω the top, will generate such a domain, and vice versa: we call $D_{Th} = \mathcal{F}(Th)$ the domain of filters w.r.t. $\leq_{Th}$ ordered by subset inclusion, and Th_D the theory of the restriction of the order in D to the compacts $\mathcal{K}(D)$. Therefore $D_{Th_D} = \mathcal{F}(Th_D) \simeq D$ which we abbreviate by $\mathcal{F}_D$ and identify with D itself.

Let $Th_{\mathsf{V}} = (\mathit{ValType}, \leq_{\mathsf{V}})$ and $Th_{\mathsf{C}} = (\mathit{ComType}, \leq_{\mathsf{C}})$ and set $D_* = D_{Th_{\mathsf{V}}}$ and $TD_* = D_{Th_{\mathsf{C}}}$: then Th_{V} is a continuous EATS (see e.g. [1] ch. 3, where continuity is expressed by condition ($\mathcal{F}$*refl*) of Prop. 3.3.18), hence the space of continuous functions $D_* \to TD_*$ is representable in D_*, and actually isomorphic to it. On the other hand the theory Th_{C} is parametric in Th_{V}. More precisely given a type theory Th we can use the axioms of Th_{C} to form a new theory we call $T(Th)$; then we can define a mapping $\mathbf{T}$ among objects of $\mathcal{D}$ by $\mathbf{T}D = D_{T(Th)}$ where $Th = Th_D$.

Theorem 4. *Define* $unit^{\mathcal{F}}_D : \mathcal{F}_D \to \mathcal{F}_{TD}$ *and* $\star^{\mathcal{F}}_{D,E} : \mathcal{F}_{TD} \times \mathcal{F}_{D \to \mathbf{T}E} \to \mathcal{F}_{\mathbf{T}E}$ *by:*

$$unit^{\mathcal{F}}_D\, d = \uparrow\{T\delta \in \mathcal{T}_{\mathbf{T}D} \mid \delta \in d\} \qquad t \star^{\mathcal{F}}_{D,E} e = \uparrow\{\tau \in \mathcal{T}_{\mathbf{T}E} \mid \exists\, \delta \to \tau \in e.\, T\delta \in t\}$$

Then $(\mathbf{T}, unit^{\mathcal{F}}, \star^{\mathcal{F}})$ *is a monad over* $\mathcal{D}$*. Hence* D_* *is a* T*-model.*

Strictly speaking to enforce extensionality of the filter model, Th_{V} must be extended to the theory Th^η_{V} by adding suitable axioms: see [4] for the precise treatment.

By stratifying types according to the rank map: $r(\alpha) = r(\omega_{\mathsf{V}}) = r(\omega_{\mathsf{C}}) = 0$, $r(\sigma \wedge \sigma') = \max(r(\sigma), r(\sigma'))$, $r(\delta \to \tau) = \max(r(\delta){+}1, r(\tau))$ and $r(T\delta) = r(\delta){+}1$, and taking $\leq_n = \leq \upharpoonright \{\sigma \mid r(\sigma) \leq n\}$ (for both $\leq_{\mathsf{V}}$ and $\leq_{\mathsf{C}}$) we obtain theories Th_n and a chain of domains $D_n = \mathcal{F}(\mathit{Th}_n)$ such that $D_* = \lim_{\leftarrow} D_n$ is a limit T-model. Consequently, we can extend the proof in [2] to our calculus obtaining:

Theorem 5 (Completeness). *Let $\mathcal{C}$ be the class of limit T-models. Then*

$$\Gamma \models^{\mathcal{C}} V : \delta \;\Rightarrow\; \Gamma \vdash V : \delta \quad \text{and} \quad \Gamma \models^{\mathcal{C}} M : \tau \;\Rightarrow\; \Gamma \vdash M : \tau.$$

Corollary 1 (Subject expansion). *If $\Gamma \vdash M : \tau$ and $N \longrightarrow M$ then $\Gamma \vdash N : \tau$.*

Finally let $\mathit{Term}^0 = \mathit{Val}^0 \cup \mathit{Com}^0$ be the set of closed terms.

Definition 3. *Let $\Downarrow \subseteq \mathit{Com}^0 \times \mathit{Val}^0$ be the smallest relation satisfying:*

$$\frac{}{\mathit{unit}\, V \Downarrow V} \qquad\qquad \frac{M \Downarrow V \qquad N[V/x] \Downarrow W}{M \star \lambda x.N \Downarrow W}$$

Then it is easily seen that $M \Downarrow V$ if and only if $M \xrightarrow{*} \mathit{unit}\, V$. We abbreviate $M \Downarrow \Leftrightarrow \exists V.\, M \Downarrow V$.

We say that $\tau \in \mathit{ComType}$ is *non trivial* if $\omega_{\mathsf{C}} \not\leq_{\mathsf{C}} \tau$. Then by adapting Tait's computability technique, we eventually have:

Theorem 6. *For all $M \in \mathit{Com}^0$ we have:*

$$M \Downarrow \;\Leftrightarrow\; \exists \tau \text{ non trivial} \,.\, \vdash M : \tau$$

Corollary 2 (Computational Adequacy). *In the model D_* we have that*

$$M \Downarrow \;\Leftrightarrow\; [\![M]\!]^{TD_*} \neq \bot_{TD_*}$$

From the proof of Theorem 6 we learn that the fact that $T\omega_{\mathsf{V}}$ is not equated to ω_{C} in Th_{C} is an essential ingredient; indeed this corresponds to the fact that the generic monad T is assumed to be non trivial (hence not the identity monad), so that $TD \not\simeq D$. This supports the intuition that a T-model equating computations to (the image of) values is not computationally adequate w.r.t. weak normal forms.

For details we refer the reader to the full paper [4].

References

1. Amadio, R., Curien, P.L.: Domains and lambda-calculi. Cambridge University Press (1998)
2. Barendregt, H., Coppo, M., Dezani-Ciancaglini, M.: A filter lambda model and the completeness of type assignment. Journal of Symbolic Logic **48**(4), 931–940 (1983)
3. Dal Lago, U., Gavazzo, F., Levy, P.B.: Effectful applicative bisimilarity: Monads, relators, and howe's method. In: Proc. of Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017. pp. 1–12 (2017)
4. de'Liguoro, U., Treglia, R.: Intersection Types for the Computational lambda-Calculus (Jul 2019), https://arxiv.org/abs/1907.05706, unpublished
5. Moggi, E.: Computational Lambda-calculus and Monads. Report ECS-LFCS-88-66, University of Edinburgh, Edinburgh, Scotland (Oct 1988)
6. Moggi, E.: Notions of Computation and Monads. Information and Computation **93**, 55–92 (1991)
7. Wadler, P.: Monads for Functional Programming. In: Advanced Functional Programming, First International Spring School on Advanced Functional Programming Techniques-Tutorial Text. Lecture Notes in Computer Science, vol. 925, pp. 24–52. Springer-Verlag (1995)