Intersection Types for the Computational λ-Calculus

Ugo de'Liguoro, Riccardo Treglia

University of Turin, Italy

INRIA - Sophia Antipolis 16-12-2019

Why effects in functional languages

Why effects in functional languages

- Semantic theories are an established field providing foundations to software analysis and verification

Why effects in functional languages

- Semantic theories are an established field providing foundations to software analysis and verification
- We have nice theories of the semantics for functional programming languages, but procedural languages commonly used for both sequential and concurrent programming have non-functional features, collectively called **effects**

Why effects in functional languages

- Semantic theories are an established field providing foundations to software analysis and verification
- We have nice theories of the semantics for functional programming languages, but procedural languages commonly used for both sequential and concurrent programming have non-functional features, collectively called **effects**
- Since Algol'60 project various λ-calculi have been used as metalanguages providing semantics to programming languages in general, and a successful approach among others is based on the notion of **computational monad** [Moggi89, 91]

Why effects in functional languages

- Semantic theories are an established field providing foundations to software analysis and verification
- We have nice theories of the semantics for functional programming languages, but procedural languages commonly used for both sequential and concurrent programming have non-functional features, collectively called **effects**
- Since Algol'60 project various λ-calculi have been used as metalanguages providing semantics to programming languages in general, and a successful approach among others is based on the notion of **computational monad** [Moggi89, 91]
- Monads have been studied and implemented mainly for typed languages; here we address the issue of modelling effects in the untyped case.

Computational Monads (after Wadler)

A *monad* is a triple $(T, unit, \star)$

Types

D is the type of values;

TD is the type of computations (with **effects**) over D.

Operators

$unit_D : D \to TD$ (Haskell: **return**);

$\star_{D,E} : TD \to (D \to TE) \to TE$ (Haskell: >>=).

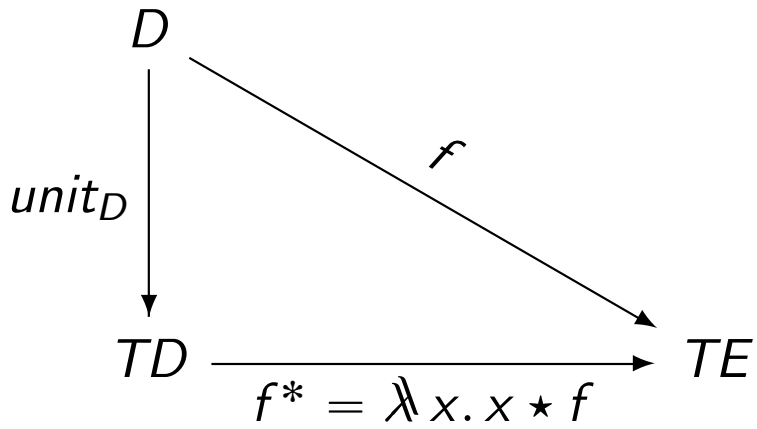
Axioms

$(unit\ d) \star f = f\ d$

$a \star unit = a$

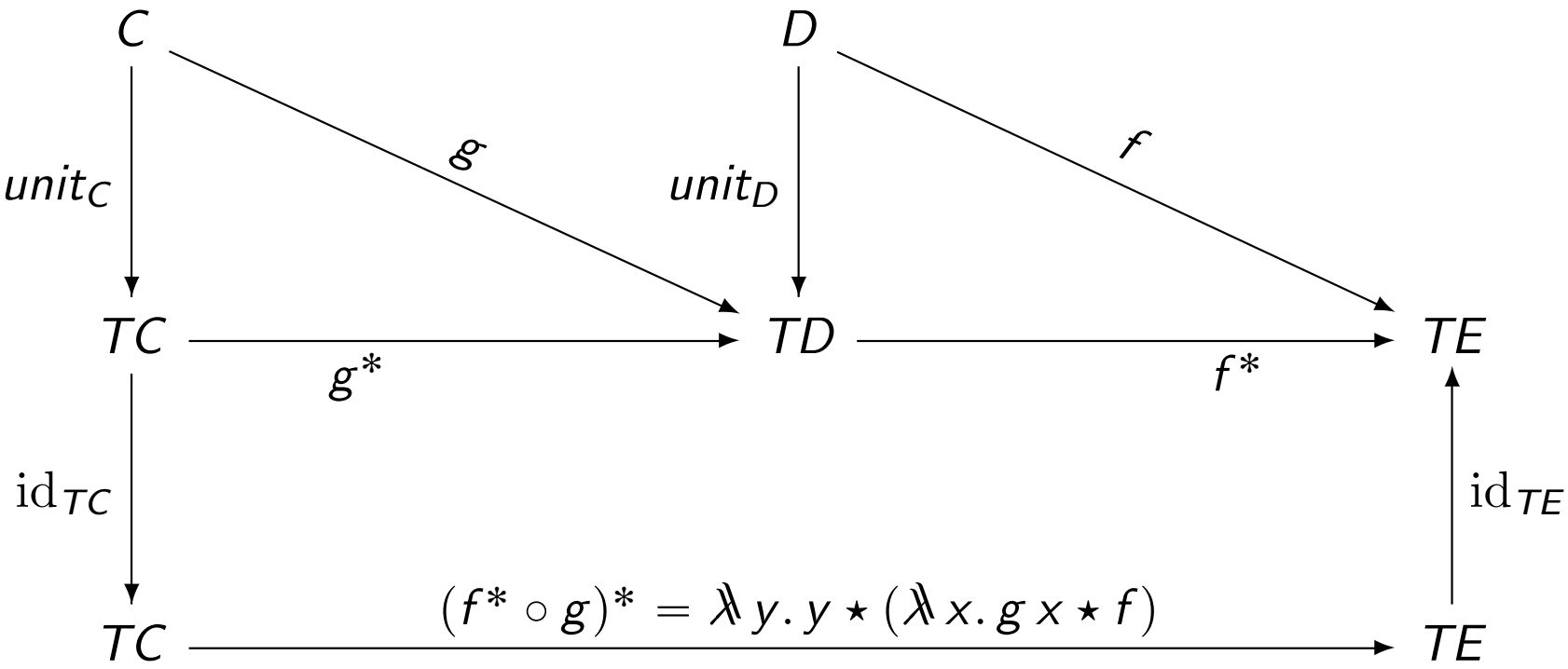
$(a \star f) \star g = a \star \lambda d.(f\ d \star g)$

Computational Monads (after Wadler)



$$(f^* \circ unit)\, d = unit\, d \star f = f\, d \qquad d \in D$$

Computational Monads (after Wadler)



$$(f^* \circ g)^*\, m = m \star (\lambda x.\, g\, x \star f) = (m \star g) \star f \qquad m \in TC$$

Moggi's type-free λ_c

Moggi considered a type-free version of his λ_c, obtained by type erasure:

$$\begin{array}{rrcl} \text{Terms}: & e, e' & ::= & v \mid n \\ \text{Values}: & v & ::= & x \mid \lambda x.e \\ \text{NonValues}: & n & ::= & \mathit{let}\ x = e\ \mathit{in}\ e' \mid e\, e' \end{array}$$

Problems:

- a quite rich syntax with complex equational theory
- a reduction $e \rightsquigarrow e'$ is defined with seven rules (five about *let*)
- no clear model interpreting non values like: $x\,x$ and $x\,x\,x$

Untyped computational λ-calculus: λ_c^u

We consider Moggi's call-by-value reflexive object:

$$D = D \to TD$$

hence there are two types, D and TD, and two kinds of terms:

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

having types:

$$x : D \vdash x : D \qquad \frac{x : D \vdash M : TD}{\vdash \lambda x.M : D \to TD = D}$$

$$\frac{\vdash V : D}{\vdash \mathit{unit}\ V : TD} \qquad \frac{\vdash M : TD \quad \vdash V : D = D \to TD}{\vdash M \star V : TD}$$

Untyped computational λ-calculus: λ_c^u

We consider Moggi's call-by-value reflexive object:

$$D = D \to TD$$

hence there are two types, D and TD, and two kinds of terms:

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

and a natural interpretation:

$$[\![\cdot]\!]^D : Val \to Env_D \to D \qquad [\![\cdot]\!]^{TD} : Com \to Env_D \to TD$$

where $\rho \in Env_D = Var \to D$ and

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

Reduction

Orienting monad axioms we get $\longrightarrow \subseteq \mathit{Com} \times \mathit{Com}$ as the compatible closure of:

$$\begin{array}{lrcll} (\beta_c) & \mathit{unit}\ V \star (\lambda x.M) & \longrightarrow & M[V/x] & \\ (\mathrm{id}) & M \star \lambda x.\mathit{unit}\ x & \longrightarrow & M & \\ (\mathrm{ass}) & (L \star \lambda x.M) \star \lambda y.N & \longrightarrow & L \star \lambda x.(M \star \lambda y.N) & \text{for } x \notin FV(N) \end{array}$$

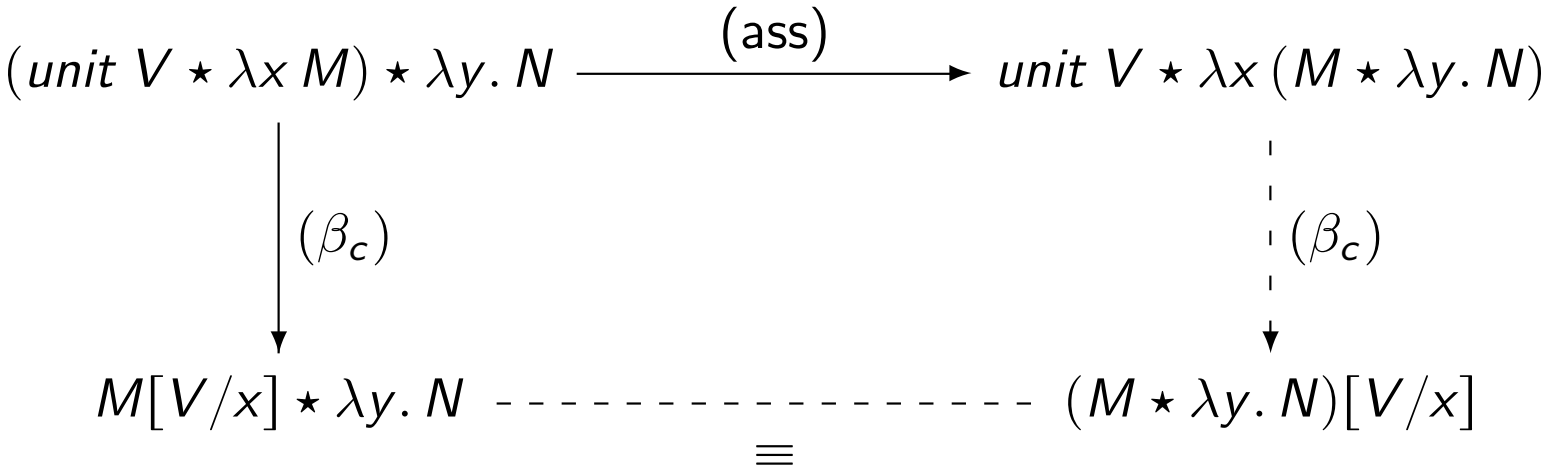
where $M[V/x]$ denotes the capture avoiding substitution of V for all free occurrences of x in M.

To have extensionality we add:

$$(\eta_c) \quad \mathit{unit}\ \lambda x.(\mathit{unit}\ x \star V) \longrightarrow \mathit{unit}\ V \quad \text{if } x \notin FV(V)$$

Confluence

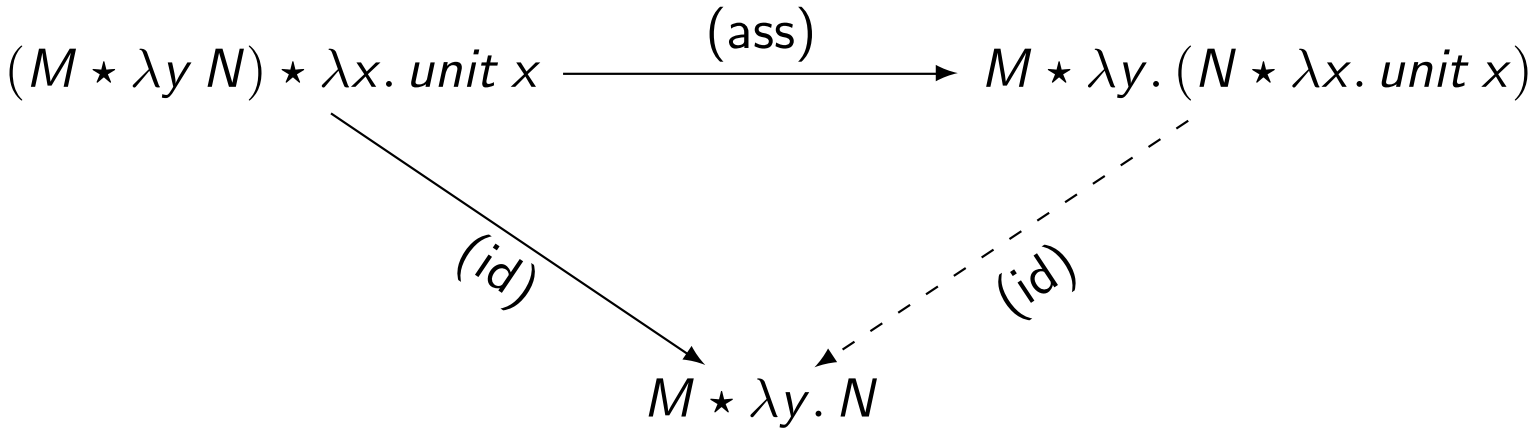
Several critical pairs:



where $x \notin FV(N)$

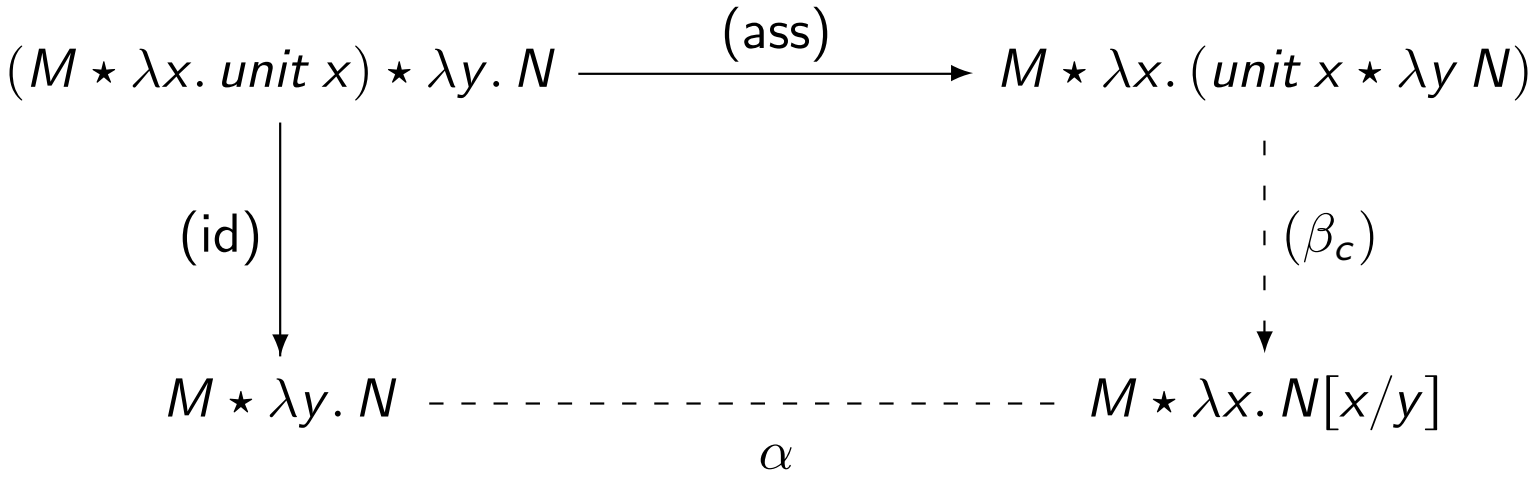
Confluence

Several critical pairs:



Confluence

Several critical pairs:



where $x \notin FV(N)$

Confluence

Theorem

If $\longrightarrow$ *is either* $\longrightarrow_{\beta_c}$ *or* $\longrightarrow_{\beta_c\eta_c}$ *then*

$$M \xrightarrow{*} N \qquad \text{and} \qquad M \xrightarrow{*} L$$

implies

$$N \xrightarrow{*} M' \qquad \text{and} \qquad L \xrightarrow{*} M'$$

for some $M' \in Com$.

Proof: by the parallel reduction technique.

Also the reduction relation $\rightsquigarrow$ is confluent; this has been shown by Hamana in 2018, by an implicit proof.

λ_c^u versus type free λ_c

let is definable:

$$let\ x = M\ in\ N \equiv M \star \lambda x.N$$

No primitive functional application but

$$(\lambda x.M)V \equiv (unit\ V) \star (\lambda x.M)$$

$$VM \equiv M \star V$$

$$MN \equiv M \star \lambda z.(N \star z) \text{ for some fresh } z$$

By this we retrieve ordinary call-by-value reduction rule:

$$(\beta_v) \qquad (\lambda x.M)V \equiv (unit\ V) \star (\lambda x.M) \longrightarrow_{\beta_c} M[V/x]$$

λ_c^u versus type free λ_c

There exist the mappings

$$\lfloor\cdot\rfloor^{\mathsf{V}} : \mathrm{Values} \to \mathit{Val} \qquad \lfloor\cdot\rfloor^{\mathsf{C}} : \mathrm{NonValues} \to \mathit{Com}$$

where e.g.

- $\lfloor\lambda x.v\rfloor^{\mathsf{V}} = \lambda x.\mathit{unit}\,\lfloor v\rfloor^{\mathsf{V}}$ and $\lfloor\lambda x.n\rfloor^{\mathsf{V}} = \lambda x.\lfloor n\rfloor^{\mathsf{C}}$
- $\lfloor nv\rfloor^{\mathsf{C}} = \lfloor n\rfloor^{\mathsf{C}} \star (\lambda z.\mathit{unit}\,\lfloor v\rfloor^{\mathsf{V}} \star z)$ and
  $\lfloor nn'\rfloor^{\mathsf{C}} = \lfloor n\rfloor^{\mathsf{C}} \star (\lambda z.\lfloor n'\rfloor^{\mathsf{C}} \star z)$ for fresh z
- $\lfloor \mathit{let}\ x = n\ \mathit{in}\ v\rfloor^{\mathsf{C}} = \lfloor n\rfloor^{\mathsf{C}} \star \lambda x.\mathit{unit}\,\lfloor v\rfloor^{\mathsf{V}}$ and
  $\lfloor \mathit{let}\ x = v\ \mathit{in}\ n'\rfloor^{\mathsf{C}} = \mathit{unit}\,\lfloor v\rfloor^{\mathsf{V}} \star \lambda x.\lfloor n'\rfloor^{\mathsf{C}}$

Then putting

$$\lfloor n\rfloor = \lfloor n\rfloor^{\mathsf{C}} \qquad \lfloor v\rfloor = \mathit{unit}\,\lfloor v\rfloor^{\mathsf{V}}$$

we have

$$e \rightsquigarrow e' \Rightarrow \lfloor e\rfloor \xrightarrow{*}_{\beta_c\eta_c} \lfloor e'\rfloor$$

Types and assignment system

Definition (Intersection types for values and computations)

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

Types and assignment system

Definition (Intersection types for values and computations)

ValType :	δ	$::=$	$\alpha \mid \delta \to \tau \mid \delta \wedge \delta \mid \omega_{\mathsf{V}}$	refine D	(*value types*)
ComType :	τ	$::=$	$T\delta \mid \tau \wedge \tau \mid \omega_{\mathsf{C}}$	refine TD	(*computation types*)

$$x : D \vdash x : D \qquad \frac{x : D \vdash M : TD}{\vdash \lambda x.M : D \to TD = D}$$

Types and assignment system

Definition (Intersection types for values and computations)

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

$$\frac{x : \delta \in \Gamma}{\Gamma \vdash x : \delta}\,(\mathsf{Ax}) \qquad \frac{\Gamma, x : \delta \vdash M : \tau}{\Gamma \vdash \lambda x.M : \delta \to \tau}\,(\to \mathsf{I})$$

Types and assignment system

Definition (Intersection types for values and computations)

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

$$\frac{x : \delta \in \Gamma}{\Gamma \vdash x : \delta}\,(\mathsf{Ax}) \qquad \frac{\Gamma, x : \delta \vdash M : \tau}{\Gamma \vdash \lambda x.M : \delta \to \tau}\,(\to \mathsf{I})$$

$$\frac{\vdash V : D}{\vdash \mathit{unit}\ V : TD} \qquad \frac{\vdash M : TD \quad \vdash V : D = D \to TD}{\vdash M \star V : TD}$$

Types and assignment system

Definition (Intersection types for values and computations)

ValType :	δ	$::=$	$\alpha \mid \delta \to \tau \mid \delta \wedge \delta \mid \omega_{\mathsf{V}}$	refine *D*	(*value types*)
ComType :	τ	$::=$	$T\delta \mid \tau \wedge \tau \mid \omega_{\mathsf{C}}$	refine *TD*	(*computation types*)

$$\frac{x : \delta \in \Gamma}{\Gamma \vdash x : \delta}\,(\mathsf{Ax}) \qquad \frac{\Gamma, x : \delta \vdash M : \tau}{\Gamma \vdash \lambda x.M : \delta \to \tau}\,(\to \mathsf{I})$$

$$\frac{\Gamma \vdash V : \delta}{\Gamma \vdash \mathit{unit}\ V : T\delta}\,(\mathit{unit}\ \mathsf{I}) \qquad \frac{\Gamma \vdash M : T\delta \quad \Gamma \vdash V : \delta \to \tau}{\Gamma \vdash M \star V : \tau}\,(\to \mathsf{E})$$

where $\Gamma, x : \delta = \Gamma \cup \{x : \delta\}$ with $x : \delta \notin \Gamma$.

Subtyping

Definition

A *subtyping relation* is defined as the smallest relation on types such that:

$$\delta \leqslant_{\mathsf{V}} \omega_{\mathsf{V}} \qquad \omega_{\mathsf{V}} \leqslant_{\mathsf{V}} \omega_{\mathsf{V}} \to \omega_{\mathsf{C}}$$

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

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

So we add pertinent rules:

$$\frac{}{\Gamma \vdash P : \omega}\ (\omega) \qquad \frac{\Gamma \vdash P : \sigma \quad \Gamma \vdash P : \sigma'}{\Gamma \vdash P : \sigma \wedge \sigma'}\ (\wedge I) \qquad \frac{\Gamma \vdash P : \sigma \quad \sigma \leqslant \sigma'}{\Gamma \vdash P : \sigma'}\ (\mathit{Sub})$$

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

Subtyping

Definition

A *subtyping relation* is defined as the smallest relation on types such that:

$$\delta \leqslant_{\mathsf{V}} \omega_{\mathsf{V}} \qquad \omega_{\mathsf{V}} \leqslant_{\mathsf{V}} \omega_{\mathsf{V}} \to \omega_{\mathsf{C}}$$

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

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

So we add pertinent rules:

$$\frac{}{\Gamma \vdash P : \omega}\ (\omega) \qquad \frac{\Gamma \vdash P : \sigma \quad \Gamma \vdash P : \sigma'}{\Gamma \vdash P : \sigma \wedge \sigma'}\ (\wedge I) \qquad \frac{\Gamma \vdash P : \sigma \quad \sigma \leqslant \sigma'}{\Gamma \vdash P : \sigma'}\ (\mathit{Sub})$$

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

Subject reduction and expansion

Theorem (Subject reduction)

If $\Gamma \vdash M : \tau$ *and* $M \longrightarrow N$ *then* $\Gamma \vdash N : \tau$.

Theorem (Subject expansion)

If $\Gamma \vdash M : \tau$ *and* $N \longrightarrow M$ *then* $\Gamma \vdash N : \tau$.

Type interpretation

Given $\xi \in \mathit{TypeEnv}_D = \mathit{TypeVar} \to 2^D$ define

$$[\![\cdot]\!]^D : \mathit{ValType} \times \mathit{TypeEnv}_D \to 2^D \qquad [\![\cdot]\!]^{TD} : \mathit{ComType} \times \mathit{TypeEnv}_D \to 2^{TD}$$

by

$$\begin{array}{rclcrcl}
[\![\alpha]\!]^D_\xi & = & \xi(\alpha) & & [\![\delta \to \tau]\!]^D_\xi & = & \{d \in D \mid \forall a \in [\![T\,\delta]\!]^{TD}_\xi .\ a \star d \in [\![\tau]\!]^{TD}_\xi\} \\
 & & & & [\![T\,\delta]\!]^{TD}_\xi & = & \{\mathit{unit}\ d \mid d \in [\![\delta]\!]_\xi\} \\
[\![\omega_V]\!]^D_\xi & = & D & & [\![\delta \wedge \delta']\!]^D_\xi & = & [\![\delta]\!]^D_\xi \cap [\![\delta']\!]^D_\xi \\
[\![\omega_C]\!]^{TD}_\xi & = & TD & & [\![\tau \wedge \tau']\!]^{TD}_\xi & = & [\![\tau]\!]^{TD}_\xi \cap [\![\tau']\!]^{TD}_\xi
\end{array}$$

Lemma

$$\begin{array}{rcl}
\delta \leqslant_V \delta' & \Rightarrow & \forall \xi \in \mathit{TypeEnv}_D .\ [\![\delta]\!]^D_\xi \subseteq [\![\delta']\!]^D_\xi \\
\tau \leqslant_C \tau' & \Rightarrow & \forall \xi \in \mathit{TypeEnv}_D .\ [\![\tau]\!]^{TD}_\xi \subseteq [\![\tau']\!]^{TD}_\xi
\end{array}$$

Soundness and completeness

A *model* of λ_c^u is a structure $\mathcal{M} = (D, T, \mathit{unit}, \star)$ such that

$$(T, \mathit{unit}, \star) \text{ is a monad and } D \simeq D \to TD$$

- $\rho, \xi \models^{\mathcal{M}} \Gamma$ if $\rho(x) \in [\![\Gamma(x)]\!]^D_\xi$ for all $x \in \mathrm{dom}(\Gamma)$
- $\Gamma \models^{\mathcal{M}} V : \delta$ ($\Gamma \models^{\mathcal{M}} M : \tau$) if $\rho, \xi \models^{\mathcal{M}} \Gamma$ implies $[\![V]\!]^D_\rho \in [\![\delta]\!]^D_\xi$ ($[\![M]\!]^{TD}_\rho \in [\![\tau]\!]^{TD}_\xi$)
- $\Gamma \models V : \delta$ ($\Gamma \models M : \tau$) if $\Gamma \models^{\mathcal{M}} V : \delta$ ($\Gamma \models^{\mathcal{M}} M : \tau$) for all $\mathcal{M}$.

Theorem

- $\Gamma \vdash V : \delta \Leftrightarrow \Gamma \models V : \delta$
- $\Gamma \vdash M : \tau \Leftrightarrow \Gamma \models M : \tau$

Convergence

Let $\Downarrow \subseteq Com^0 \times Val^0$ be the smallest relation satisfying:

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

Define

$$M \Downarrow \quad \Leftrightarrow \quad \exists V \in Val^0.\ M \Downarrow V$$

Lemma

$$M \Downarrow \quad \Leftrightarrow \quad \exists V \in Val^0.\ M \xrightarrow{*} unit\ V$$

Characterization by non trivial typing

Let $\mathcal{I} : \mathit{TypeVar} \to \mathscr{P}\mathit{Val}^0$ be a map; then define

$$|\delta|_{\mathcal{I}} \subseteq \mathit{Val}^0 \qquad |\tau|_{\mathcal{I}} \subseteq \mathit{Com}^0$$

by induction as follows:

- $|\alpha|_{\mathcal{I}} = \mathcal{I}(\alpha)$
- $|\delta \to \tau|_{\mathcal{I}} = \{V \in \mathit{Val}^0 \mid \forall M \in |T\delta|_{\mathcal{I}}.\ M \star V \in |\tau|_{\mathcal{I}}\}$
- $|T\delta|_{\mathcal{I}} = \{M \in \mathit{Com}^0 \mid \exists V \in |\delta|_{\mathcal{I}}.\ M \Downarrow V\}$
- $|\omega_{\mathsf{V}}|_{\mathcal{I}} = \mathit{Val}^0$ and $|\omega_{\mathsf{C}}|_{\mathcal{I}} = \mathit{Com}^0$
- $|\delta \wedge \delta'|_{\mathcal{I}} = |\delta|_{\mathcal{I}} \cap |\delta'|_{\mathcal{I}}$ and $|\tau \wedge \tau'|_{\mathcal{I}} = |\tau|_{\mathcal{I}} \cap |\tau'|_{\mathcal{I}}$.

Lemma

Let $\Gamma \vdash M : \tau$ where $\Gamma = \{x_1 : \delta_1, \ldots, x_k : \delta_k\}$ and $M \in \mathit{Com}$. For any $V_1, \ldots, V_k \in \mathit{Val}^0$ and $\mathcal{I}$, if $V_i \in |\delta_i|_{\mathcal{I}}$ for all $i = 1, \ldots, k$ then $M[V_1/x_1] \cdots [V_k/x_k] \in |\tau|_{\mathcal{I}}$.

Characterization by non trivial typing

Theorem

For all $M \in Com^0$ we have:

$$M \Downarrow \quad \Leftrightarrow \quad \exists \tau \neq_C \omega_C.\ \vdash M : \tau.$$

Proof. If $M \Downarrow$ then $M \xrightarrow{*} unit\ V$ for some $V \in Val^0$; since $\vdash V : \omega_V$ we have $\vdash unit\ V : T(\omega_V) \neq_C \omega_C$, so $\vdash M : \tau$ by subject expansion.

If $\vdash M : \tau$ and $M \in Com^0$ then $M \in |\tau|_{\mathcal{I}}$ and for all $\mathcal{I}$; $\tau \neq_C \omega_C$ implies $\tau \leqslant_C T(\omega_V)$ and hence $|\tau|_{\mathcal{I}} \subseteq |T(\omega_V)|_{\mathcal{I}} = \{N \in Com^0 \mid N \Downarrow\}$.

State monad

A statefull computation is a function:

$$\textit{initial state} \mapsto (\textit{computed value}, \textit{initial state})$$

State monad: $(\mathbf{S}, \textit{unit}_S, \star_S)$ where

- $\mathbf{S}D = S \to D \times S$, where say $S \subseteq \mathcal{L} \to D$ and $\mathcal{L}$ is a set of locations
- $\textit{unit}_S\, d = \lambda\!\!\lambda\, s.\, (d, s)$
- $a \star_S f = \lambda\!\!\lambda\, s.\, \textit{let}\ (d, s') = a\, s\ \textit{in}\ f\, d\, s'$

and suppose:

$$D \simeq D \to (S \to D \times S)$$

Then

$$\textit{Com}: \qquad M, N \quad ::= \quad \ldots \mid\ !\ell \mid \ell := M$$

where

$$\begin{aligned} [\![!\ell]\!]^{SD}_{\rho} &= \lambda\!\!\lambda\, s.\, (s[\ell], s) \\ [\![\ell := M]\!]^{SD}_{\rho} &= [\![M]\!]^{SD}_{\rho} \star_S \lambda\!\!\lambda\, d\, s.\, (d, s[\ell \mapsto d]) \end{aligned}$$

State monad

$$\begin{array}{rrcl} \mathit{StateType}: & \sigma & ::= & \langle \ell : \delta \rangle \mid \sigma \wedge \sigma' \mid \omega_S \\ \mathit{ValType}: & \delta & ::= & \alpha \mid \delta \to \tau \mid \delta \wedge \delta' \mid \omega_V \\ \mathit{ComType}: & \tau & ::= & \sigma \to \delta \times \sigma' \mid \tau \wedge \tau \mid \omega_C \end{array}$$

$$\frac{\Gamma \vdash V : \delta}{\Gamma \vdash \mathit{unit}\ V : \sigma \to (\delta \times \sigma)}$$

$$\frac{\Gamma \vdash M : \sigma \to (\delta \times \sigma') \quad \Gamma \vdash V : \delta \to \sigma' \to (\delta' \times \sigma'')}{\Gamma \vdash M \star V : \sigma \to (\delta' \times \sigma'')}$$

and observe that (omitting ξ):

$$[\![\delta \to \sigma' \to (\delta' \times \sigma'')]\!]^D = [\![(\delta \times \sigma') \to (\delta' \times \sigma'')]\!]^D$$

State monad

Postulating

$$\delta \leqslant_{\mathsf{V}} \delta' \Rightarrow \langle \ell : \delta \rangle \leqslant_{\mathsf{S}} \langle \ell : \delta' \rangle \qquad \langle \ell : \delta \rangle \wedge \langle \ell : \delta' \rangle \leqslant_{\mathsf{S}} \langle \ell : \delta \wedge \delta' \rangle$$

we can write the typing rules:

$$\frac{\sigma \leqslant_{\mathsf{S}} \langle \ell : \delta \rangle}{\Gamma \vdash !\ell : \sigma \to \delta \times \sigma}$$

$$\frac{\Gamma \vdash M : \sigma \to \delta \times \sigma' \qquad \sigma'' = \sigma'[\langle \ell : \delta \rangle / \langle \ell : \_ \rangle]}{\Gamma \vdash \ell := M : \sigma \to \delta \times \sigma''}$$

where $\sigma'[\langle \ell : \delta \rangle / \langle \ell : \_ \rangle]$ is the replacement of $\langle \ell : \_ \rangle$ by $\langle \ell : \delta \rangle$ in σ'

$$\frac{\Gamma \vdash M : \sigma \to \delta \times \sigma' \qquad \Gamma \vdash N : \sigma' \to \delta' \times \sigma''}{\Gamma \vdash M; N \equiv M \star \lambda\_.N : \sigma \to \delta' \times \sigma''}$$

which is a derived rule, making it apparent that the value of M, that has type δ, is discarded, while the produced state of type σ' is passed to N.

State monad

Suppose $\delta \neq_V \delta'$:

$$\dfrac{\dfrac{\vdash V : \delta}{\vdash \mathit{unit}\ V : \langle \ell : \delta' \rangle \to \delta \times \langle \ell : \delta' \rangle}}{\vdash \ell := \mathit{unit}\ V : \langle \ell : \delta' \rangle \to \delta \times \langle \ell : \delta \rangle}$$

Now

$$[\![V]\!]^D \in [\![\delta]\!]^D \qquad \text{and} \qquad [\![\ell := \mathit{unit}\ V]\!]^{SD} \in [\![\langle \ell : \delta' \rangle \to \delta \times \langle \ell : \delta \rangle]\!]^{SD}$$

but

$$[\![\ell := \mathit{unit}\ V]\!]^{SD} \notin \{\mathit{unit}_S d \mid d \in [\![\delta]\!]^D\}$$

Monadic type interpretation

Definition

Type interpretations $[\![\cdot]\!]^D$ and $[\![\cdot]\!]^{TD}$ are monadic if for any $d \in D$ and $a \in TD$:

(i) $d \in [\![\delta]\!]^D_\xi \Rightarrow \mathit{unit}\, d \in [\![T\delta]\!]^{TD}_\xi$

(ii) $d \in [\![\delta' \to T\delta]\!]^D_\xi \;\&\; a \in [\![T\delta']\!]^{TD}_\xi \Rightarrow a \star d \in [\![T\delta]\!]^{TD}_\xi$

This definition is not inductive! in particular $[\![T\delta]\!]^{TD}_\xi$ depends on itself and also on $[\![T\delta']\!]^{TD}_\xi$ for any δ'.

Monadic type interpretation

Suppose that $D_\infty = \lim_{\leftarrow} D_n$ where D_0 is arbitrary and

$$D_{n+1} = [D_n \to TD_n]$$

Then we can inductively define

$$[\![\delta]\!]_\xi^{D_n} \subseteq D_n \qquad \text{and} \qquad [\![\tau]\!]_\xi^{TD_n} \subseteq TD_n$$

so that we have

Theorem

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

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

Conclusion and open issues

Results

- Moving from equation $D = D \to TD$ for a generic computational monad T we have defined a type assignment system that is sound and complete for a pure untyped computational λ-calculus.
- The type assignment system enjoys basic properties of intersection type systems: invariance under conversion, soundness and completeness w.r.t. standard domain semantics, computational adequacy.

Ongoing work

- Is there a reasonable notion of type **refinement** w.r.t. generic T-types?
- Is there a uniform way to **instantiate** the generic type system to concrete monads, preserving basic properties?
- Can we extend this approach to algebras and coalgebras of a monad?

Reference

Full paper available on arXiv.org:

Ugo de'Liguoro, Riccardo Treglia
Intersection Types for the Computational lambda-Calculus,
July 2019.

http://arxiv.org/abs/1907.05706