Parametric Polymorphism and Operational Equivalence

Andrew M. Pitts

Cambridge University Computer Laboratory

- Impredicative parametric polymorphism from a programming language perspective.
- Operationally-based relational parametricity as a tool for proving contextual equivalence of programs.

Draft paper:

`ftp://ftp.cl.cam.ac.uk/papers/ap/parpoe.ps`

Girard-Reynolds Polymorphic Lambda Calculus (PLC)

Types: $\tau ::= \alpha \mid \tau \to \tau \mid \forall\, \alpha\, (\tau)$

Terms: $M ::= x \mid \lambda\, x : \tau\, (M) \mid M\, M \mid \Lambda\, \alpha\, (M) \mid M\, \tau$

Typing: $\alpha_1, \ldots, \alpha_m, x_1 : \tau_1, \ldots, x_n : \tau_n \vdash M : \tau$

Reduction:
$$\begin{array}{rcl} (\lambda\, x : \tau\, (M_1))\, M_2 & \to & M_1[M_2/x] \\ (\Lambda\, \alpha\, (M))\, \tau & \to & M[\tau/\alpha] \end{array}$$

Famous SN theorem of Girard: *there are no infinite chains of reduction starting from a well-typed PLC term.*

Examples of polymorphic types

$$
\begin{aligned}
1 &\sim \forall\,\alpha\,(\alpha \to \alpha)\\
\alpha_1 \times \alpha_2 &\sim \forall\,\alpha\,((\alpha_1 \to \alpha_2 \to \alpha) \to \alpha)\\
0 &\sim \forall\,\alpha\,(\alpha)\\
\alpha_1 + \alpha_2 &\sim \forall\,\alpha\,((\alpha_1 \to \alpha) \to (\alpha_2 \to \alpha) \to \alpha)\\
\mathit{bool} &\sim \forall\,\alpha\,(\alpha \to \alpha \to \alpha)\\
\mathit{nat} &\sim \forall\,\alpha\,(\alpha \to (\alpha \to \alpha) \to \alpha)\\
\alpha\ \mathit{list} &\sim \forall\,\alpha'\,(\alpha' \to (\alpha \to \alpha' \to \alpha') \to \alpha')\\
\mu\,\alpha\,(\tau) &\sim \forall\,\alpha\,((\tau \to \alpha) \to \alpha) \qquad \text{(1,2)}\\
\exists\,\alpha\,(\tau) &\sim \forall\,\alpha'\,(\forall\,\alpha\,(\tau \to \alpha') \to \alpha') \qquad \text{(1)}\\
\nu\,\alpha\,(\tau) &\sim \forall\,\alpha'\,(\forall\,\alpha\,((\alpha \to \tau) \to \alpha \to \alpha') \to \alpha') \qquad \text{(1,2)}
\end{aligned}
$$

(1) α' not free in τ. (2) only occurrences of α in τ are positive.

Commonly held view:

Virtually any datatype of interest can be encoded within PLC.
I.e. PLC is to functional programming statics (type systems) as untyped lambda calculus is to its dynamics.

But PLC is a calculus of *total* functions.

(Cf. Girard's SN theorem.)

What happens when we combine PLC with features of functional programming that involve possible non-termination?

(E.g. recursive definitions, potentially infinite ('lazy') datatypes, etc.)

In this case, what should 'can be encoded' mean, and what in fact can be encoded?

PLC^{+} = cbn PLC + lazy lists + fixpoints

Types: $\tau ::= \alpha \mid \tau \to \tau \mid \forall\, \alpha\, (\tau) \mid \tau\, \mathit{list}$

Terms: function abstraction and application, type generalisation and specialisation, and constants:

$$
\begin{aligned}
\mathbf{fix} &: \forall\, \alpha\, ((\alpha \to \alpha) \to \alpha) \\
\mathbf{nil} &: \forall\, \alpha\, (\alpha\, \mathit{list}) \\
\mathbf{cons} &: \forall\, \alpha\, (\alpha \to \alpha\, \mathit{list} \to \alpha\, \mathit{list}) \\
\mathbf{case} &: \forall\, \alpha, \alpha'\, (\alpha\, \mathit{list} \to \alpha' \to (\alpha \to \alpha\, \mathit{list} \to \alpha') \to \alpha')
\end{aligned}
$$

PLC^+ transition relation (à la Felleisen)

$$(\lambda\, x : \tau\, (M_1))\, M_2 \to M_1[M_2/x]$$

$$(\Lambda\, \alpha\, (M))\, \tau \to M[\tau/\alpha]$$

$$\mathbf{fix}\, \tau\, M \to M(\mathbf{fix}\, \tau\, M)$$

$$\mathbf{case}\, \tau\, \tau'(\mathbf{nil}\, \tau)\, M\, M' \to M$$

$$\mathbf{case}\, \tau\, \tau'(\mathbf{cons}\, \tau\, M_1\, M_2)\, M\, M' \to M'\, M_1\, M_2$$

$$\frac{M \to M'}{\mathcal{E}[M] \to \mathcal{E}[M']}$$

where $\mathcal{E}$ ranges over

evaluation contexts:

$$\mathcal{E} ::= - \mid \mathcal{E}\, M \mid \mathcal{E}\, \tau \mid \mathbf{case}\, \tau\, \tau\, \mathcal{E}$$

Contextual equivalence of PLC^+ terms

We choose only to observe results at list types.

$$M_1 \cong_{\mathrm{ctx}} M_2 : \tau \;\stackrel{\mathrm{def}}{\Leftrightarrow}$$

$$\forall\, \tau',\ \forall \text{ contexts } \mathcal{M}[- : \tau] : \tau'\ \mathit{list}$$

$$\mathcal{M}[M_1] \to^* \mathbf{nil}\,\tau' \;\Leftrightarrow\; \mathcal{M}[M_2] \to^* \mathbf{nil}\,\tau'$$

(For simplicity, we'll restrict attention here to the case where τ is a *closed* type and M_1, M_2 are *closed* terms.)

A typical definability question

Letting $L(\alpha) \stackrel{\text{def}}{=} \forall\,\alpha'\,(\alpha' \to (\alpha \to \alpha' \to \alpha') \to \alpha')$ be the usual polymorphic type of α-lists

are there PLC^{+} terms

$$I : \forall\,\alpha\,(\alpha\,\mathit{list} \to L(\alpha)) \text{ and } J : \forall\,\alpha\,(L(\alpha) \to \alpha\,\mathit{list})$$

such that for all closed τ and $M : \tau$

$$\begin{aligned} J\,\tau(I\,\tau\,M) &\cong_{\mathrm{ctx}} M : \tau\,\mathit{list} \\ I\,\tau(J\,\tau\,M') &\cong_{\mathrm{ctx}} M' : L(\tau) \end{aligned}$$

In other words, up to contextual equivalence, is $\tau\,\mathit{list}$ isomorphic to the pure PLC type $L(\tau)$ (polymorphically in τ)?

Aim

Characterise $\cong_{\mathrm{ctx}}$ in terms of a logical relation, Δ_τ:

$$M_1 \cong_{\mathrm{ctx}} M_2 : \tau \;\Leftrightarrow\; (M_1, M_2) \in \Delta_\tau \quad (\tau \textit{ closed})$$

defined by induction on structure of τ.

In general, for an *open* type $\tau(\vec{\alpha})$, Δ_τ will be a *function mapping tuples of term relations to term relations*:

$$\vec{r} \in \mathrm{Rel}(\vec{\sigma}, \vec{\sigma}') \mapsto \Delta_\tau(\vec{r}) \in \mathrm{Rel}(\tau(\vec{\sigma}), \tau(\vec{\sigma}'))$$

where, given closed PLC^+ types σ and σ', $\boxed{\mathrm{Rel}(\sigma, \sigma')}$ denotes the set of binary relations between closed terms of type σ and closed terms of type σ'.

Ingredients of the logical relation Δ

Operational semantics of PLC^{+}.

Extensionality properties of $\to$ and $list$ types:

- $$\begin{aligned}\Delta_{\tau \to \tau'} = \{(F, F') \mid \ & \forall\, (A, A') \in \Delta_{\tau} \\ & (F\, A, F'\, A') \in \Delta_{\tau'}\}.\end{aligned}$$

- $\Delta_{\tau\ list}$ is coinductively defined—it's the greatest 'Δ_{τ}-bisimulation'.

Relational parametricity at $\forall$-types ...

Relational parametricity at $\forall$-types

Relationally parametric models of PLC (Reynolds 1983) suggest defining $\Delta_{\forall\,\alpha\,(\tau)} = \forall\,r\,(\Delta_\tau(r))$, where:

$$\begin{aligned}\forall\,r\,(\Delta_\tau(r))\,(\vec{r}) \;\stackrel{\text{def}}{=}\; \{(G, G') \mid\; & \forall\,r \in \mathrm{Rel}(\sigma, \sigma')\\ & (G\,\sigma, G\,\sigma') \in \Delta_\tau(r, \vec{r})\}\end{aligned}$$

But this won't do for PLC^+, because then e.g.

$$\begin{aligned}\Delta_{\forall\,\alpha\,(\alpha)} &= \forall\,r\,(r) = \emptyset \quad \text{(take } r = \emptyset\text{)}\\ &\neq \{(G, G') \mid G \cong_{\mathrm{ctx}} G' : \forall\,\alpha\,(\alpha)\} \quad \text{because} \ldots\end{aligned}$$

$$\ldots \quad \Omega \cong_{\mathrm{ctx}} \Omega \stackrel{\text{def}}{=} \Lambda\,\alpha\,(\mathbf{fix}\,\alpha(\lambda\,x : \alpha\,(x))) : \forall\,\alpha\,(\alpha).$$

A Galois connection between term-relations and evaluation-context-relations

$$\frac{r \in \mathrm{Rel}(\tau,\tau')}{r^{\top} \in \mathrm{Rel}^{\top}(\tau,\tau')} \qquad \frac{s \in \mathrm{Rel}^{\top}(\tau,\tau')}{s^{\top} \in \mathrm{Rel}(\tau,\tau')}$$

$$\boxed{\begin{array}{l} (\mathcal{E},\mathcal{E}') \in r^{\top} \overset{\text{def}}{\Leftrightarrow} \forall (M,M') \in r \ (\mathcal{E}[M] \updownarrow \mathcal{E}'[M']) \\ (M,M') \in s^{\top} \overset{\text{def}}{\Leftrightarrow} \forall (\mathcal{E},\mathcal{E}') \in s \ (\mathcal{E}[M] \updownarrow \mathcal{E}'[M']) \end{array}}$$

where $\mathcal{E}[M] \updownarrow \mathcal{E}'[M']$ means

$$\mathcal{E}[M] \to^{*} \mathbf{nil}\,\tau_1 \ \Leftrightarrow\ \mathcal{E}'[M'] \to^{*} \mathbf{nil}\,\tau_1'$$

(for the appropriate τ_1, τ_1')

and where $\boxed{\mathrm{Rel}^{\top}(\tau,\tau')}$ = set of binary relations between evaluation contexts of form $\mathcal{E}[- : \tau] : ?\ \mathit{list}$ and of form $\mathcal{E}[- : \tau'] : ?'\ \mathit{list}$.

$\top\top$-closed term-relations are *admissible* for fixpoint induction

Lemma Given $r \in \mathrm{Rel}(\tau, \tau')$ satisfying $r = r^{\top\top}$,

$$(F, F') \in r \to r \;\Rightarrow\; (\mathbf{fix}\,\tau\,F, \mathbf{fix}\,\tau'\,F') \in r$$

(Proof uses an 'unwinding' property of $\mathbf{fix}$ with respect to the transition relation.)

Definition of the logical relation for PLC^{+}

$$\Delta_{\alpha_i}(\vec{r}) \stackrel{\mathrm{def}}{=} r_i$$

$$\Delta_{\tau\to\tau'}(\vec{r}) \stackrel{\mathrm{def}}{=} \Delta_{\tau}(\vec{r}) \to \Delta_{\tau'}(\vec{r})$$

$$\Delta_{\tau\ list}(\vec{r}) \stackrel{\mathrm{def}}{=} \nu\, r\, ((1 + \Delta_{\tau}(\vec{r}) \times r)^{\top\top})$$

$$\Delta_{\forall\,\alpha\,(\tau)}(\vec{r}) \stackrel{\mathrm{def}}{=} \forall\, r\, (\Delta_{\tau}(r^{\top\top}, \vec{r}))$$

‘Lazy’ PCF

(call-by-name evaluation; termination at function types is observable)

$$\Delta_{\tau_1 \to \tau_2}(\vec{r}) \stackrel{\text{def}}{=} (\lambda\, \Delta_{\tau_1}(\vec{r})\, (\Delta_{\tau_2}(\vec{r})))^{\top\top}$$

where in general

$$\begin{aligned}
&\lambda\, r_1\, (r_2) \stackrel{\text{def}}{=} \\
&\qquad \{(\lambda\, x : \tau_1\, (M), \lambda\, x : \tau_1\, (M')) \mid \forall\, (A, A') \in r_1 \\
&\qquad\qquad\qquad\qquad (M[A/x], M'[A'/x]) \in r_2\}.
\end{aligned}$$

Call-by-value PCF

(call-by-value evaluation; hence termination at function types is necessarily observable)

$$\Delta_{\tau_1 \to \tau_2}(\vec{r}) \stackrel{\text{def}}{=} (\lambda_v \, \Delta_{\tau_1}(\vec{r}) \, (\Delta_{\tau_2}(\vec{r})))^{\top\top}$$

where in general

$$\begin{aligned}\lambda_v \, r_1 \, (r_2) \stackrel{\text{def}}{=} \{(\lambda \, x : \tau_1 \, (M), \lambda \, x : \tau_1 \, (M')) \mid \\ \forall \, (C, C') \in r_1 \text{ with } C, C' \text{ canonical} \\ (M[C/x], M'[C'/x]) \in r_2\}.\end{aligned}$$

Theorem For closed PLC^+ types and terms

$$M_1 \cong_{\mathrm{ctx}} M_2 : \tau \;\Leftrightarrow\; (M_1, M_2) \in \Delta_\tau$$

Consequences:

- Validity of 'Kleene equivalences' up to $\cong_{\mathrm{ctx}}$.
- Operational extensionality results for $\cong_{\mathrm{ctx}}$ (e.g. 'ciu' theorem: $\cong_{\mathrm{ctx}} = Id^{\top}$).
- Definability results, like that on Slide 8.
- Wadler-style 'theorems for free' for $\cong_{\mathrm{ctx}}$ (Wadler 1989).
- Category-theoretic properties of polymorphic types up to $\cong_{\mathrm{ctx}}$ (cf. Hasegawa 1991 and Plotkin and Abadi 1993).

A simple application of the Theorem

Up to contextual equivalence, the only element of type $\forall\,\alpha\,(\alpha)$ is 'polymorphic bottom'. In other words, for all PLC^{+} terms M

$$M : \forall\,\alpha\,(\alpha) \;\Rightarrow\; M \cong_{\mathrm{ctx}} \Omega : \forall\,\alpha\,(\alpha)$$

where

$$\Omega \stackrel{\mathrm{def}}{=} \Lambda\,\alpha\,(\mathbf{fix}\,\alpha(\lambda\,x : \alpha\,(x))).$$

Proof. If $M : \forall\,\alpha\,(\alpha)$, then $M \cong_{\mathrm{ctx}} M : \forall\,\alpha\,(\alpha)$, so by the Theorem $(M, M) \in \Delta_{\forall\,\alpha\,(\alpha)} = \forall\, r\,(r^{\top\top})$.

Hence for any closed τ have $\boxed{(M\,\tau, M\,(\tau\ \mathit{list})) \in r^{\top\top}}$ for any $r \in \mathrm{Rel}(\tau, \tau\ \mathit{list})$ and hence in particular for

$$r \stackrel{\text{def}}{=} \{(\Omega\,\tau, \Omega\,(\tau\ \mathit{list}))\}.$$

Now, given any closed evaluation context $\mathcal{E}[- : \tau] : \tau'\ \mathit{list}$, letting $\mathcal{D}[- : \tau\ \mathit{list}] : \tau\ \mathit{list}$ be an evaluation context that always diverges on any argument, one has $\boxed{(\mathcal{E}, \mathcal{D}) \in r^{\top}}$. Then $\boxed{}$+$\boxed{}$ imply $\mathcal{E}[M\,\tau] \updownarrow \mathcal{D}[M\,(\tau\ \mathit{list})]$, and hence $\mathcal{E}[M\,\tau] \not\to^{*} \mathbf{nil}\,\tau'$ (by choice of $\mathcal{D}$).

Since τ and $\mathcal{E}$ are arbitrary, it follows from this (by extensionality properties of $\cong_{\mathrm{ctx}}$ derived from Theorem) that $M \cong_{\mathrm{ctx}} \Omega : \forall\,\alpha\,(\alpha)$. □

Retro- & Pro-spect

Types of *polymorphic lambda calculus + fixpoints* have very rich properties up to $\cong_{\mathrm{ctx}}$; and operationally-based logical relations are a convenient way of proving them (compared with, e.g., domain-theoretic models).

The $(-)^{\top}$ operation is useful for defining all sorts of operationally-based logical relations (cf. Pitts-Stark contribution to HOOTS volume.) *What are* $\top\top$*-closed relations denotationally?*

Future work: properties of types up to $\cong_{\mathrm{ctx}}$ in

- $F_{<}$ + fixpoints
- (with Bierman) second order Linear Logic + fixpoints.

  (Cf. Plotkin, unpublished.)