The recursion hierarchy for PCF is strict

John Longley

School of Informatics
University of Edinburgh
jrl@inf.ed.ac.uk

25 August 2015

The main result

PCF is simply-typed λ-calculus with base type $\mathtt{N}$ and constants

$$\widehat{0}, \widehat{1}, \ldots \;:\; \mathtt{N}\,, \qquad \mathit{suc}, \mathit{pre} \;:\; \mathtt{N} \to \mathtt{N}\,,$$
$$\mathit{ifzero} \;:\; \mathtt{N} \to \mathtt{N} \to \mathtt{N} \to \mathtt{N}\,, \qquad Y_\sigma \;:\; (\sigma \to \sigma) \to \sigma$$

Evaluation relation $M \Downarrow \widehat{n}$ defined by standard (call-by-name) operational semantics.

Question: Do we need infinitely many Y_σ to get the full power of PCF? (Berger 1999; folklore before then.)

Define sublanguages PCF_k by allowing Y_σ only when $\mathrm{lv}(\sigma) \leq k$.

Here $\mathrm{lv}(\mathtt{N}) = 0$, $\mathrm{lv}(\sigma_0 \to \cdots \to \sigma_{r-1} \to \mathtt{N}) = 1 + \max_i(\mathrm{lv}(\sigma_i))$.

Theorem: For any k, there's a closed term $M \in \mathrm{PCF}_{k+1}$ that's not observationally equivalent to any term of PCF_k.

(Makes no difference if observing contexts are drawn from PCF or just PCF_0. Can even restrict to applicative contexts $-N_0 \ldots N_{r-1}$.)

More denotational formulation

Let $\mathsf{SF}^{\mathrm{eff}}(\sigma)$ be the set of closed PCF terms $M : \sigma$ modulo $\approx_{\mathrm{obs}}$. Then we have well-defined application operations

$$\cdot \;:\; \mathsf{SF}^{\mathrm{eff}}(\sigma \to \tau) \times \mathsf{SF}^{\mathrm{eff}}(\sigma) \to \mathsf{SF}^{\mathrm{eff}}(\tau)\;.$$

Here $\mathsf{SF}^{\mathrm{eff}}(\mathtt{N}) \cong \mathbb{N}_\perp$, and $\mathsf{SF}^{\mathrm{eff}}(\sigma \to \tau)$ is isomorphic to a set of *functions* $\mathsf{SF}^{\mathrm{eff}}(\sigma) \to \mathsf{SF}^{\mathrm{eff}}(\tau)$ (by Milner's context lemma).
So we have an extensional type structure $\mathsf{SF}^{\mathrm{eff}}$ of 'sequentially computable' functionals.

The following fundamental property of $\mathsf{SF}^{\mathrm{eff}}$ (making no reference to recursion) is an easy consequence of our Theorem.
Corollary: There is no finite 'basis' $B \subset \mathsf{SF}^{\mathrm{eff}}$ such that every element of $\mathsf{SF}^{\mathrm{eff}}$ is λ-definable relative to B.

Aside: Some contrasting results

The situation is quite different for several extensions of PCF studied in the literature. E.g. in each of the following languages:

- PCF + parallel-or + exists (Plotkin)
- PCF + catch (Cartwright/Felleisen)
- PCF + H (Longley)

even just $Y_{\mathtt{N}\to\mathtt{N}}$ (with finitely many other constants) suffices to express all programs up to observational equivalence.
Such results, along with the surprising things that can be done even in PCF_1 (e.g. exhaustive search over certain infinite sets), lead one not to expect a 'cheap' proof of our main theorem.

Outline of proof

We use the model of PCF consisting of sequential procedures (a.k.a. PCF Böhm trees—the history is complicated). These are generated by the following grammar, interpreted coinductively:

$$
\begin{array}{rcl}
p, q & ::= & \lambda x_0 \cdots x_{r-1}.\, e \\
d, e & ::= & \perp \;\mid\; n \;\mid\; \mathtt{case}\ x\, q_0 \cdots q_{r-1}\ \mathtt{of}\ (0 \Rightarrow e_0 \mid 1 \Rightarrow e_1 \mid \cdots)
\end{array}
$$

Can define application $p \cdot q$, so we get a type structure SP^0 of closed sequential procedures.

1. Define a substructure $\mathsf{A}_k \subseteq \mathsf{SP}^0$ of k-acceptable procedures. Show that all PCF_k terms have denotations in A_k.
2. Show that no k-acceptable procedure is spinal (i.e. has a certain kind of 'infinite spine' characteristic of Y_{k+1}).
3. For a certain PCF_{k+1}-definable functional Φ, show that every procedure representing Φ is spinal.

Sequential procedures: examples

$$* : \mathbb{N}^2 \to \mathbb{N}: \qquad \lambda x^{\mathtt{N}} y^{\mathtt{N}}.\ \mathtt{case}\ x\ \mathtt{of}\ \big(\begin{array}{l} \ 0 \Rightarrow 0 \\ \mid 1 \Rightarrow \mathtt{case}\ y\ \mathtt{of}\ (0 \Rightarrow 0 \mid 1 \Rightarrow 1 \mid 2 \Rightarrow 2 \mid \cdots) \\ \mid 2 \Rightarrow \mathtt{case}\ y\ \mathtt{of}\ (0 \Rightarrow 0 \mid 1 \Rightarrow 2 \mid 2 \Rightarrow 4 \mid \cdots) \\ \mid \cdots\ \big) \end{array}$$

$$Y_k : (\overline{k} \to \overline{k}) \to \overline{k}: \quad \begin{array}{l} \lambda g^{k \to k} x^{k-1}.\ \mathtt{case}\ g(\ ) x^{\eta}\ \mathtt{of}\ (i \Rightarrow i) \\ \qquad\qquad\qquad \downarrow \\ \qquad\qquad\qquad \lambda x'^{k-1}.\ \mathtt{case}\ g(\ ) x'^{\eta}\ \mathtt{of}\ (i \Rightarrow i) \\ \qquad\qquad\qquad\qquad\qquad \downarrow \\ \qquad\qquad\qquad\qquad\qquad \vdots \end{array}$$

Here $\overline{k}$ is the pure type of level k: $\overline{0} = \mathtt{N}$, $\overline{k+1} = \overline{k} \to \mathtt{N}$.
We define x^{η} by induction on the type level of x:

$$x^{\eta} = \lambda z.\ \mathtt{case}\ x z^{\eta}\ \mathtt{of}\ (i \Rightarrow i)$$

$$
\begin{array}{rcl}
p, q & ::= & \lambda x_0 \cdots x_{r-1}.\, e \\
d, e & ::= & \perp \;|\; n \;|\; \texttt{case}\; x\, q_0 \cdots q_{r-1} \;\texttt{of}\; (0 \Rightarrow e_0 \mid 1 \Rightarrow e_1 \mid \cdots)
\end{array}
$$

Sequential procedures as defined above are in fact the normal forms in a wider calculus of meta-terms, for which a notion of reduction is defined. E.g.

$$\texttt{case}\; (\lambda x.5)(\lambda.\perp)\; \texttt{of}\; (i \Rightarrow i+1) \rightsquigarrow \texttt{case}\; 5\; \texttt{of}\; (i \Rightarrow i+1) \rightsquigarrow 6$$

Reduction is in general an infinite process: for any meta-term T, its value $\ll T \gg$ is defined as a limit of finite approximations.
Application: $p \cdot q =_{def} \ll \lambda \vec{z}.pq\vec{z}^{\eta} \gg$.

SP^0 gives a good model of PCF. All computable procedures are PCF-denotable, and the type structure SF (sequential functionals) is the extensional quotient of SP^0.
SP^0 seems very rich in interesting substructures (e.g. take the well-founded or left-well-founded or left-bounded procedures).
These have already yielded new non-definability results, e.g.:

- In a certain language Iter_1 with 'first order iteration', the 'lazy' primitive recursor rec_0 is not definable.
- Bar recursion (at the simplest type) is not definable in System T + 'iteration at all types', but is definable in PCF_1.

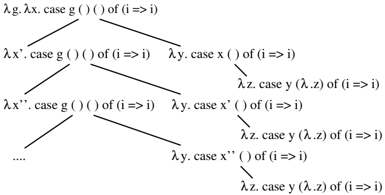
Definition of the submodel A_k is more subtle (and inductive). The key formation rule is k-plugging: combining of existing procedures via repeated substitutions for variables at level $\leq k$ (subject to certain constraints).

Why the definition must be subtle

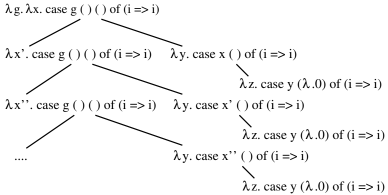
Let $Z_{k+1} : (\overline{k+1} \to \overline{k+1}) \to \overline{k+1}$ represent $Y_k : (\overline{k} \to \overline{k}) \to \overline{k}$ w.r.t. some standard embedding $\overline{k} \lhd \overline{k+1}$.

The submodel A_k must admit the NSP for Z_{k+1}, but not that for Y_{k+1}.

Y_3 :



Z_3 :



Definition of A_k

Suppose Z is a set of variables of type level $\leq k$. If $\Gamma, Z \vdash e$ is an expression and $\Gamma, Z \vdash \zeta(z)$ is a suitably typed procedure for each $z \in Z$, we obtain the plugging $\Gamma \vdash \Pi(e, \zeta)$ (a meta-term) by starting from e and repeatedly expanding each z to $\zeta(z)$.

The formation rules for A_k are:

- The obvious ones for well-founded procedure terms.
- Plugging rule: if Z, e, ζ are as above and $\Gamma, Z \vdash e$ and each $\Gamma, Z \vdash \zeta(z)$ are in A_k, then $\Gamma \vdash \ll \Pi(e, \zeta) \gg$ is in A_k.

Intuition: Information welling up from arbitrary depth must be funnelled through a type of level $\leq k$.

2. No acceptable procedure is spinal

Roughly, a term is spinal w.r.t. $g^{(k+1)\to(k+1)}$ if it contains an infinite nested sequence of applications that look a bit like

$$g\,(\lambda x'^{k}.\ \cdots)\,x^{k^{\eta}}\,.$$

Main lemma: If a k-plugging results in a spinal term q, then a spine must already be present in one of the fragments p_i involved.
Intuition: Suppose that q contained two consecutive spinal occurrences of g originating from different fragments p_i, p_j.
Any 'communication' between p_i and p_j must be mediated by variables of level $\leq k-1$. So somehow, the whole of the relevant x'^{k} must be funnelled through such variables.
Impossible: $\overline{k}$ is known not to be a retract of any level $k-1$ type.
(Can think informally of $\mathrm{SP}^0(\overline{k})$ as 'k-dimensional space'.)

The above already shows that within SP^0, Y_{k+1} is not definable in PCF_k. But we want a result of this kind within the extensional quotient SF.
In PCF_{k+1}, use $Y_{\mathtt{N}\to(k+1)}$ to define

$$\begin{aligned} \Phi \; &: \; ((k+1) \to (k+1)) \to \mathtt{N} \to (k+1) \\ \Phi\, g\, n \; &= \; g\, n\, (\Phi\, g\, (n+1)) \end{aligned}$$

Informally, $\Phi\, g\, n = g(n,\, g(n+1,\, g(n+2,\, \cdots)))$.
Can show that *any* procedure q representing $\Phi \in \mathsf{SF}$ is k-spinal. (If q deviates in any way from a spinal structure, can cook up arguments showing that the extension of q is different from Φ.)
Thus $\Phi \in \mathsf{SF}$ is not PCF_k-definable.

Type complexity: a closer look

- For first-order types σ, the language Iter_1 (weaker than PCF_1) suffices for defining all elements of $\mathsf{SF}^{\mathrm{eff}}(\sigma)$.
- Same is true for $\sigma = (\mathtt{N} \to \mathtt{N}) \to \mathtt{N}$, but not constructively.
- For general second-order types, Iter_1 is not enough, but PCF_1 suffices; in fact, we only need $Y_{\mathtt{N}\to\mathtt{N}}$.
- For third order types, $Y_{\mathtt{N}\to\mathtt{N}}$ no longer suffices. Don't know whether PCF_1 suffices; certainly PCF_2 does.
- For types of order $k \geq 4$, PCF_{k-3} does not suffice, but PCF_{k-2} does.

Open questions

- Small gap: we haven't shown that Y_{k+1} isn't PCF_k-definable, only that $Y_{\mathtt{N}\to(k+1)}$ isn't.
- Map out the expressivity hierarchy in finer detail. E.g. our current analysis shows that $Y_{\mathtt{N}\to\mathtt{N}} \prec Y_{\mathtt{N}^3\to\mathtt{N}}$, but not that $Y_{\mathtt{N}\to\mathtt{N}} \prec Y_{\mathtt{N}^2\to\mathtt{N}}$ or that $Y_{\mathtt{N}^2\to\mathtt{N}} \prec Y_{\mathtt{N}^3\to\mathtt{N}}$.
- Map out the corresponding picture for call-by-value types.
- Can the proof be presented as clearly (or better) in terms of game semantics? If not, why not?
- Find an interesting program that makes essential use of Y_2.

Paper available: Longley, J.R., *The recursion hierarchy for PCF is strict.* Informatics Research Report 1421, July 2015.

And finally . . .

My book with Dag Normann on Higher-Order Computability is scheduled to be published by Springer on 12 October 2015.
Now pre-orderable from the Springer website, and from Amazon.
(570 pages; hardcover price £99).