The recursion hierarchy for PCF is strict

The recursion hierarchy for PCF is strict

John Longley

July 17, 2015

Abstract

Let PCF_k denote the sublanguage of Plotkin's PCF in which fixed point operators Y_σ are admitted only for types σ of level $\leq k$. We show that the languages PCF_k form a strict hierarchy, in the sense that for each k, there are closed programs of PCF_{k+1} that are not observationally equivalent to any programs of PCF_k. This answers a question posed by Berger in 1999. Our proof makes substantial use of the theory of *nested sequential procedures* (also called PCF *Böhm trees*) as expounded in the forthcoming book of Longley and Normann.

1 Introduction

In this paper we study sublanguages of Plotkin's functional programming language PCF, which we here take to be the simply typed λ-calculus over a single base type $\mathtt{N}$, with constants

$$\begin{array}{rlrl} \widehat{n} & : \mathtt{N} \text{ for each } n \in \mathbb{N}\,, & \mathit{suc}, \mathit{pre} & : \mathtt{N} \to \mathtt{N}\,, \\ \mathit{ifzero} & : \mathtt{N} \to \mathtt{N} \to \mathtt{N} \to \mathtt{N}\,, & Y_\sigma & : (\sigma \to \sigma) \to \sigma \text{ for each type } \sigma\,. \end{array}$$

As usual, we consider this language to be endowed with a certain (call-by-name) operational semantics, which in turn gives rise to a notion of *observational equivalence* for PCF programs.

We define the *level* $\mathrm{lv}(\sigma)$ of a type σ inductively by

$$\mathrm{lv}(\mathtt{N}) \;=\; 0\,, \qquad \mathrm{lv}(\sigma \to \tau) \;=\; \max(\mathrm{lv}(\sigma)+1, \mathrm{lv}(\tau))\,,$$

and define the *pure type* $\overline{k}$ of level $k \in \mathbb{N}$ by

$$\overline{0} \;=\; \mathtt{N}\,, \qquad \overline{k+1} \;=\; \overline{k} \to \mathtt{N}\,.$$

Modifying the definition of PCF so that the constants Y_σ are admitted only for types σ of level $\leq k$, we obtain a sublanguage PCF_k for any $k \in \mathbb{N}$. Our main result will be that for each k, the expressive power of PCF_{k+1} strictly exceeds that of PCF_k: in particular, there is no closed term of PCF_k that is observationally equivalent to $Y_{\overline{0} \to \overline{k+1}}$. (Note that 'observational equivalence' has the same meaning for all the languages of interest here, as will be explained in Section 2.) This answers a question posed explicitly by Berger in [4], but present in the folklore at least since the early 1990s. It is worth noting that the situation is quite different in several extensions of PCF considered in the literature, in which one can restrict to recursions at level 1 types without loss of expressivity (see the end of Subsection 2.4).

Our discussion will focus on two models of PCF, both studied extensively in the forthcoming book of Longley and Normann [12]: the *nested sequential procedure* model SP^0 (also known as the PCF *Böhm tree* model and by various other names), and its extensional quotient SF consisting of *sequential functionals*. These have effective submodels $\mathsf{SP}^{0,\mathrm{eff}}$ and $\mathsf{SF}^{\mathrm{eff}}$ respectively. It is well-known that SF and $\mathsf{SF}^{\mathrm{eff}}$ are fully abstract for PCF, and indeed that $\mathsf{SF}^{\mathrm{eff}}$ is isomorphic to the closed term model of PCF modulo observational equivalence. Our result can therefore be understood more denotationally as saying that more elements of $\mathsf{SF}^{\mathrm{eff}}$ are denotable in PCF_{k+1} than in PCF_k. (It follows easily that the same holds with $\mathsf{SF}^{\mathrm{eff}}$ replaced by any other adequate, compositional model of PCF, such as the Scott model.) We may also easily deduce that there is no finite 'basis' $B \subseteq \mathsf{SF}^{\mathrm{eff}}$ relative to which all elements of $\mathsf{SF}^{\mathrm{eff}}$ are λ-definable (see Corollary 3 below).

In Section 2 we recall the necessary technical background on PCF and on the models SP^0 and SF, fleshing out the ideas outlined informally above. In Section 3 we define (for arbitrary $k \geq 1$) a substructure $\mathsf{A}^0_k \subseteq \mathsf{SP}^0$, and show that every closed term of PCF_k has a denotation in A^0_k. In Section 4 we show that this substructure excludes the standard interpretation of $Y_{\overline{k+1}}$ in SP^0. This suffices to establish a weak version of our result, namely that within the model SP^0, the element $Y_{\overline{k+1}}$ is not definable in PCF_k; however, it still leaves open the possibility that there might be other NSPs, distinct from Y but extensionally equivalent to it, that are denotable in PCF_k. We will show in Section 5 that this is not the case, at least if we consider $Y_{\overline{0 \to k+1}}$ in place of $Y_{\overline{k+1}}$; we therefore have an element of SF denotable in PCF_{k+1} but not in PCF_k. We conclude in Section 6 with a discussion of related and future work.

I am grateful to Ulrich Berger, Martín Escardó, Dag Normann and Alex Simpson for valuable discussions and correspondence, and to Colin Stirling for drawing my attention to the related work of Damm and Statman (as discussed in Section 6).

2 Background

We here summarize the necessary definitions and technical background from [12], especially from Chapters 6 and 7.

2.1 The language PCF

In [19], Scott introduced the language LCF for computable functionals of simple type. This language is traditionally called PCF when equipped with a standalone operational semantics as in Plotkin [16]. We will work here with the same version of PCF as in [12], with the natural numbers as the only base type. Our types σ are thus generated by

$$\sigma \ ::= \ \mathtt{N} \ | \ \sigma \to \sigma \ ,$$

and our terms will be those of the simply typed λ-calculus constructed from the constants

$$\begin{array}{rcll} \widehat{n} & : & \mathtt{N} & \text{for each } n \in \mathbb{N} \ , \\ \mathit{suc},\ \mathit{pre} & : & \mathtt{N} \to \mathtt{N} \ , & \\ \mathit{ifzero} & : & \mathtt{N} \to \mathtt{N} \to \mathtt{N} \to \mathtt{N} \ , & \\ Y_\sigma & : & (\sigma \to \sigma) \to \sigma & \text{for each type } \sigma \ . \end{array}$$

We often abbreviate the type $\sigma_0 \to \cdots \to \sigma_{r-1} \to \mathtt{N}$ to $\sigma_0, \ldots, \sigma_{r-1} \to \mathbb{N}$ or just $\vec{\sigma} \to \mathtt{N}$. As usual, we write $\Gamma \vdash M : \sigma$ to mean that M is a well-typed term in the environment Γ (where Γ is a finite list of typed variables). For each $k \in \mathbb{N}$, the sublanguage PCF_k is obtained by admitting the constants Y_σ only for types σ of level $\leq k$.

We endow the class of closed PCF terms with the following small-step reduction rules:

$$\begin{array}{rclrcl} (\lambda x.M)N & \rightsquigarrow & M[x \mapsto N] \ , & \mathit{ifzero}\ \widehat{0} & \rightsquigarrow & \lambda xy.x \ , \\ \mathit{suc}\ \widehat{n} & \rightsquigarrow & \widehat{n+1} \ , & \mathit{ifzero}\ \widehat{n+1} & \rightsquigarrow & \lambda xy.y \ , \\ \mathit{pre}\ \widehat{n+1} & \rightsquigarrow & \widehat{n} \ , & Y_\sigma M & \rightsquigarrow & M(Y_\sigma M) \ . \\ \mathit{pre}\ \widehat{0} & \rightsquigarrow & \widehat{0} \ , & & & \end{array}$$

These reductions may be applied in any *evaluation context*—that is, a context generated by composition from the basic contexts

$$[-]N \ , \qquad \mathit{suc}\,[-] \ , \qquad \mathit{pre}\,[-] \ , \qquad \mathit{ifzero}\,[-] \ .$$

Thus, the relation $\rightsquigarrow$ is generated by the rules above along with the clause: if $M \rightsquigarrow M'$ and $E[-]$ is an evaluation context, then $E[M] \rightsquigarrow E[M']$. We write $\rightsquigarrow^*$ for the reflexive-transitive closure of $\rightsquigarrow$. If Q is any closed PCF term of type $\mathtt{N}$, it is easy to see that either $Q \rightsquigarrow^* \widehat{n}$ for some $n \in \mathbb{N}$, or the reduction sequence starting from Q is infinite.

This completes the definition of the languages PCF and PCF_k. Whilst the language PCF_0 is too weak for programming purposes (it cannot even define addition), it is not hard to show that even PCF_1 is Turing-complete: that is, any partial computable function $\mathbb{N} \rightharpoonup \mathbb{N}$ is representable by a closed PCF_1 term of type $\mathtt{N} \to \mathtt{N}$.

We will also refer to the non-effective language PCF^Ω (or *oracle* PCF) obtained by extending the definition of PCF with a constant $C_f : \mathtt{N} \to \mathtt{N}$ for every set-theoretic partial function $f : \mathbb{N} \rightharpoonup \mathbb{N}$, along with a reduction rule $C_f \widehat{n} \rightsquigarrow \widehat{m}$ for every n, m such that $f(n) = m$. (In PCF^Ω, the evaluation of a closed term $Q : \mathtt{N}$ may fail to reach a value $\widehat{n}$ either because it generates an infinite computation, or because it encounters a subterm $C_f(n)$ where $f(n)$ is undefined.) The languages PCF^Ω_k are defined analogously.

If M, M' are closed PCF^Ω terms of the same type σ, and $\mathcal{L}$ is one of our languages PCF_k, PCF_k^Ω, PCF, PCF^Ω, we say that M, M' are *observationally equivalent* in $\mathcal{L}$, and write $M \simeq_{\mathcal{L}} M'$, if for all closed program contexts $C[-^\sigma] : \mathtt{N}$ of $\mathcal{L}$ and all $n \in \mathbb{N}$, we have

$$C[M] \rightsquigarrow^* n \ \text{ iff } \ C[M'] \rightsquigarrow^* n \ .$$

Fortunately, it is easy to see that all of the above languages give rise to exactly the same relation $\simeq_{\mathcal{L}}$. First, it is immediate from the definition that if $\mathcal{L}, \mathcal{L}'$ are two of our languages and $\mathcal{L} \supseteq \mathcal{L}'$, then $\simeq_{\mathcal{L}} \subseteq \simeq_{\mathcal{L}'}$. It therefore only remains to show that $M \simeq_{\mathrm{PCF}_0} M'$ implies $M \simeq_{\mathrm{PCF}^\Omega} M'$. For this, we use the fact that any of the constants Y_σ or C_f in PCF^Ω can be 'approximated' to any desired accuracy by terms of PCF_0. Indeed, for any $j \in \mathbb{N}$, we may define PCF_0 terms

$$\begin{array}{rcl} Y_\sigma^{(j)} & = & \lambda f^{\sigma\to\sigma}.\, f^j(\lambda\vec{x}.\bot) \ , \\ C_f^{(j)} & = & \lambda n.\, \mathit{case}\ n\ \mathit{of}\ (0 \Rightarrow \widehat{f(0)} \mid \cdots \mid j-1 \Rightarrow \widehat{f(j-1)}) \ , \end{array}$$

writing $\bot$ for $Y_0(\lambda x^0.x)$, and using some evident syntactic sugar in the definition of $C_f^{(j)}$. For any PCF^Ω term M, let $M^{(j)}$ denote the 'approximation' obtained from M by replacing all occurrences of constants Y_σ, C_f by $Y_\sigma^{(j)}, C_f^{(j)}$ respectively. It is then not hard to show that for closed $Q : \mathtt{N}$, we have

$$Q \rightsquigarrow^* \widehat{n} \quad \text{iff} \quad \exists j.\, Q^{(j)} \rightsquigarrow^* \widehat{n} \ .$$

From this it follows easily that if $C[-]$ is an observing context of PCF^Ω that distinguishes M, M', then some approximation $C^{(j)}[-]$ (a context of PCF_0) also suffices to distinguish them. This establishes that $\simeq_{\mathrm{PCF}_0} \subseteq \simeq_{\mathrm{PCF}^\Omega}$. We may therefore write $\simeq$ for observational equivalence without ambiguity.

In fact, an even more restricted class of observing contexts suffices for ascertaining observational equivalence of PCF^Ω terms. The well-known *context lemma*, due to Milner [13], states that $M \simeq M' : \sigma_0, \ldots, \sigma_{r-1} \to \mathtt{N}$ iff M, M' have the same behaviour in all *applicative contexts* of PCF—that is, if for all closed PCF terms $N_0 : \sigma_0, \ldots, N_{r-1} : \sigma_{r-1}$, we have

$$MN_0 \ldots N_{r-1} \rightsquigarrow^* n \ \text{ iff } \ M'N_0 \ldots N_{r-1} \rightsquigarrow^* n \ .$$

Furthermore, using the above ideas of approximation, it is easy to see that we obtain exactly the same equivalence relation if we allow the N_i here to range only over closed PCF_0 terms—this gives us the notion of PCF_0 *applicative equivalence*, which we shall denote by $\sim_0$.

We have concentrated so far on giving a purely operational description of PCF. We are now able to express the operational content of our main theorem as follows. As in Section 1, we define the type $\overline{k}$ by $\overline{0} = \mathtt{N}$, $\overline{k+1} = \overline{k} \to \mathtt{N}$; we shall write $\overline{k}$ simply as k where there is no risk of confusion.

Theorem 1 *For any $k \geq 1$, there are functionals definable in PCF_{k+1} but not in PCF_k^Ω. More precisely, there is no closed term M of PCF_k^Ω such that $M \simeq Y_{0\to(k+1)}$ (or equivalently $M \sim_0 Y_{0\to(k+1)}$).*

For the status of the simpler operator Y_{k+1}, see Theorem 5 below.

Theorem 1 can be construed as saying that in a suitably pure fragment of a functional language such as Haskell, the computational strength of recursive function definitions increases strictly as the admissible type level for such recursions is increased. The formulation in terms of $\sim_0$ presents our result in a manifestly strong form: there is no $M \in \mathrm{PCF}_k$ that induces the same partial function as $Y_{0\to(k+1)}$ even on closed PCF_0 terms.

A slightly more denotational formulation of our theorem can be given in terms of the model SF of *sequential functionals*, which we may here define as the type structure of closed PCF^Ω terms modulo observational equivalence. Specifically, for each type σ, let $\mathsf{SF}(\sigma)$ denote the set of closed PCF^Ω terms $M : \sigma$ modulo $\simeq$. Clearly, application of PCF^Ω terms induces a well-defined function $\cdot : \mathsf{SF}(\sigma \to \tau) \times \mathsf{SF}(\sigma) \to \mathsf{SF}(\tau)$ for any σ, τ; the structure SF then consists of the sets $\mathsf{SF}(\sigma)$ along with these application operations. Using the context lemma, it is easy to see that $\mathsf{SF}(\mathtt{N}) \cong \mathbb{N}_\perp$ $(= \mathbb{N} \sqcup \{\perp\})$, and that $\mathsf{SF}(\sigma \to \tau)$ is isomorphic to a set of *functions* $\mathsf{SF}(\sigma) \to \mathsf{SF}(\tau)$: that is, if $f, f' \in \mathsf{SF}(\sigma \to \tau)$ satisfy $f \cdot x = f' \cdot x$ for all $x \in \mathsf{SF}(\sigma)$, then $f = f'$.

Any closed PCF^Ω term $M : \sigma$ naturally has a denotation $[\![M]\!]$ in $\mathsf{SF}(\sigma)$, namely its own equivalence class. We may therefore restate Theorem 1 as:

Theorem 2 *For any $k \geq 1$, the element $[\![Y_{0\to(k+1)}]\!] \in \mathsf{SF}$ is not denotable in PCF^Ω_k.*

It follows immediately that in any other adequate, compositional model of PCF^Ω (such as Scott's continuous model or Berry's stable model), the element $[\![Y_{0\to(k+1)}]\!]$ is not PCF^Ω_k-definable, since the equality on PCF^Ω terms induced by such a model must be contained within $\simeq$.

By taking closed terms of PCF rather than PCF^Ω modulo observational equivalence, we obtain the type structure $\mathsf{SF}^{\mathrm{eff}}$ of *effective sequential functionals*, which can clearly be seen as a substructure of SF. Although the above constructions of SF and $\mathsf{SF}^{\mathrm{eff}}$ are syntactic, there are other more mathematical constructions (for instance, involving game models [1, 7]) that also give rise to these structures, and experience shows them to be mathematically natural classes of higher-order functionals. We now see that Theorem 2 implies an interesting absolute property of $\mathsf{SF}^{\mathrm{eff}}$, not dependent on any choice of presentation for this structure or any selection of language primitives:

Corollary 3 (No finite basis) *There is no finite set B of elements of $\mathsf{SF}^{\mathrm{eff}}$ such that all elements of $\mathsf{SF}^{\mathrm{eff}}$ are λ-definable relative to B.*

Proof Suppose $B = \{b_0, \ldots, b_{n-1}\}$ were such a set. For each i, take a closed PCF term M_i denoting b_i. Then the terms $M_0, \ldots, M_{n-1}$ between them contain only finitely many occurrences of constants Y_σ, so these constants are all present in PCF_k for large enough k. But this means that $b_0, \ldots, b_{n-1}$, and hence all elements of $\mathsf{SF}^{\mathrm{eff}}$, are PCF_k-definable, contradicting Theorem 2. $\square$

2.2 The model SP^0

We turn next to an overview of the *nested sequential procedure* (or NSP) model, denoted by SP^0. A fuller treatment is given in [12, Chapter 6].

The ideas behind this model have a complex history. The general idea of sequential computation via nested oracle calls was the driving force behind Kleene's later papers (e.g. [8]), although the concept did not receive a particularly transparent or definitive formulation there. Many of the essential ideas of NSPs can be found in early work of Sazonov [17], in which a notion of *Turing machine with oracles* was used to characterize the 'sequentially computable' elements of the Scott model. NSPs as we study them here were first explicitly introduced in work on game semantics for PCF—both by Abramsky, Jagadeesan and Malacaria [1] (under the name of *evaluation trees*) and by Hyland and Ong [7] (under the name of *canonical forms*). In these papers, NSPs played only an ancillary role; however, it was shown by Amadio and Curien [3] how (under the name of PCF *Böhm trees*) they could be made into a model of PCF in their own right. Similar ideas were employed again by Sazonov [18] to give a standalone characterization of the class of sequentially computable functionals. More recently, Normann and Sazonov [15] gave an explicit construction of the NSP model in a somewhat more semantic spirit than [3], using the name *sequential procedures*. As in [12], we here add the epithet 'nested' to emphasize the contrast with other flavours of sequential computation.

As in [12], our NSPs are generated by means of the following infinitary grammar, interpreted coinductively:

$$\begin{array}{rrcl}
\textit{Procedures:} & p, q & ::= & \lambda x_0 \cdots x_{r-1}.\, e \\
\textit{Expressions:} & d, e & ::= & \bot \;\mid\; n \;\mid\; \mathtt{case}\; a\; \mathtt{of}\; (i \Rightarrow e_i \mid i \in \mathbb{N}) \\
\textit{Applications:} & a & ::= & x\, q_0 \cdots q_{r-1}
\end{array}$$

We will use vector notation to denote finite (possibly empty) lists of variables or procedures: $\vec{x}$, $\vec{q}$. We may use t to range over *NSP terms* of any of the above three kinds. We shall always work with terms up to (infinitary) α-equivalence. A procedure $\lambda\vec{x}.\bot$ will often be abbreviated to $\bot$.

If each variable is assigned a simple type over $\mathtt{N}$, then we may restrict our attention to *well-typed* terms. Informally, a term will be well-typed unless a typing violation occurs at some specific point within its syntax tree. The typing rules will mostly play only a background role in the present paper, but for the sake of completeness we note them here. Specifically, a term t is well-typed if for every application $x\vec{q}$ appearing within t, the type of x has the form $\vec{\sigma} \to \mathtt{N}$ where $\vec{\sigma}$ and $\vec{q}$ are of the same length, and for each i, the procedure q_i has the form $\lambda\vec{x_i}.e_i$, where the variables $\vec{x_i}$ have types $\vec{\tau_i}$ and $\sigma_i = \vec{\tau_i} \to \mathtt{N}$. If Γ is any environment (i.e. a finite list of variables), we write $\Gamma \vdash e$ and $\Gamma \vdash a$ to mean that e, a respectively are well-typed with free variables in Γ; we also write $\Gamma \vdash p : \tau$ when p is well-typed in Γ and of the form $\lambda\vec{x}.e$, where the variables $\vec{x}$ have types $\vec{\sigma}$ and $\tau = \vec{\sigma} \to \mathtt{N}$.

With these ideas in place, we may take $\mathsf{SP}(\sigma)$ to be the set of well-typed procedures of type σ, and $\mathsf{SP}^0(\sigma)$ the set of closed such procedures.

As in [12], we shall need to work not only with NSPs themselves, but with a more general calculus of NSP *meta-terms* designed to accommodate the intermediate forms that arise in the course of computations:

$$\begin{array}{rrcl} \textit{Meta-procedures:} & P, Q, R & ::= & \lambda\vec{x}.\, E \\ \textit{Meta-expressions:} & D, E & ::= & \bot \;\mid\; n \;\mid\; \mathtt{case}\; G \;\mathtt{of}\; (i \Rightarrow E_i \mid i \in \mathbb{N}) \\ \textit{Ground meta-terms:} & G & ::= & E \;\mid\; x\,\vec{Q} \;\mid\; P\vec{Q} \end{array}$$

Here again, $\vec{x}$ and $\vec{Q}$ denote finite lists. We shall use T to range over meta-terms of any of the above three kinds. (Unless otherwise stated, we use uppercase letters for general meta-terms and lowercase ones for terms.) Again, there are some evident typing rules for meta-terms, leading to typing judgements $\Gamma \vdash P : \sigma$, $\Gamma \vdash E$, $\Gamma \vdash G$ for meta-procedures, meta-expressions and ground meta-terms respectively. These typing rules are the obvious adaptation of those for terms (cf. [12, Section 6.1.1]); again, they will play only a background role in this paper. We will also often write e.g. $\Gamma \vdash P$ to mean that P is a well-typed meta-procedure in environment Γ, where the type itself is of no particular concern to us.

We have an evident notion of (simultaneous, capture-avoiding) substitution $T[\vec{x} \mapsto \vec{Q}]$ for well-typed terms. We will say a substitution $[\vec{x} \mapsto \vec{Q}]$ *covers* a set V of variables if V consists of precisely the variables $\vec{x}$.

As a mild extension of the concept of meta-term, we have an evident notion of a *meta-term context* $C[-]$: essentially a meta-term containing a 'hole' $-$, which may be of meta-procedure, meta-expression or ground meta-term type (and in the case of meta-procedures, will carry some type σ). Our convention is that a context $C[-]$ is permitted to contain only a single occurrence of the hole $-$. Multi-hole contexts $C[-_0, -_1, \ldots]$ will occasionally be used, but again, each hole $-_i$ may appear only once.

Next, there is a concept of *evaluation* whereby any meta-term $\Gamma \vdash T\,(:\sigma)$ evaluates to an ordinary term $\Gamma \vdash \ll T \gg\,(:\sigma)$. To define this, the first step is to introduce a *basic reduction* relation $\rightsquigarrow_b$ for ground meta-terms, which we do by the following rules:

(b1) $(\lambda\vec{x}.E)\vec{Q} \rightsquigarrow_b E[\vec{x} \mapsto \vec{Q}]$ $\quad$ (*β-rule*).

(b2) $\mathtt{case}\;\bot\;\mathtt{of}\;(i \Rightarrow E_i) \rightsquigarrow_b \bot$.

(b3) $\mathtt{case}\;n\;\mathtt{of}\;(i \Rightarrow E_i) \rightsquigarrow_b E_n$.

(b4) $\mathtt{case}\;(\mathtt{case}\;G\;\mathtt{of}\;(i \Rightarrow E_i))\;\mathtt{of}\;(j \Rightarrow F_j) \rightsquigarrow_b$
$\qquad\qquad \mathtt{case}\;G\;\mathtt{of}\;(i \Rightarrow \mathtt{case}\;E_i\;\mathtt{of}\;(j \Rightarrow F_j))$.

Note that the β-rule applies even when $\vec{x}$ is empty: e.g. $\lambda.2 \rightsquigarrow_b 2$.

From this, a *head reduction* relation $\rightsquigarrow_h$ on meta-terms is defined inductively:

(h1) If $G \rightsquigarrow_b G'$ then $G \rightsquigarrow_h G'$.

(h2) If $G \rightsquigarrow_h G'$ and G is not a `case` meta-term, then

$$\texttt{case}\, G\ \texttt{of}\ (i \Rightarrow E_i) \ \rightsquigarrow_h\ \texttt{case}\, G'\ \texttt{of}\ (i \Rightarrow E_i)\ .$$

(h3) If $E \rightsquigarrow_h E'$ then $\lambda\vec{x}.E \ \rightsquigarrow_h\ \lambda\vec{x}.E'$.

Clearly, for any meta-term T, there is at most one T' with $T \rightsquigarrow_h T'$. We call a meta-term a *head normal form* if it cannot be further reduced using $\rightsquigarrow_h$. The possible shapes of head normal forms are $\bot$, n, `case` $y\vec{Q}$ `of` $(i \Rightarrow E_i)$ and $y\vec{Q}$, the first three optionally prefixed by $\lambda\vec{x}$ (where $\vec{x}$ may contain y).

We now define the *general reduction* relation $\rightsquigarrow$ inductively as follows:

(g1) If $T \rightsquigarrow_h T'$ then $T \rightsquigarrow T'$.

(g2) If $E \rightsquigarrow E'$ then $\lambda\vec{x}.E \ \rightsquigarrow\ \lambda\vec{x}.E'$.

(g3) If $Q_j = Q'_j$ except at $j = k$ where $Q_k \rightsquigarrow Q'_k$, then

$$\begin{aligned} x\vec{Q} &\ \rightsquigarrow\ x\vec{Q'}\ , \\ \texttt{case}\, x\,\vec{Q}\ \texttt{of}\ (i \Rightarrow E_i) &\ \rightsquigarrow\ \texttt{case}\, x\,\vec{Q'}\ \texttt{of}\ (i \Rightarrow E_i)\ . \end{aligned}$$

(g4) If $E_i = E'_i$ except at $i = k$ where $E_k \rightsquigarrow E'_k$, then

$$\texttt{case}\, x\,\vec{Q}\ \texttt{of}\ (i \Rightarrow E_i)\ \rightsquigarrow\ \texttt{case}\, x\,\vec{Q}\ \texttt{of}\ (i \Rightarrow E'_i)\ .$$

It is easy to check that this reduction system is sound with respect to the typing rules.

An important point to note is that the terms t are precisely the meta-terms in *normal form*, i.e. those that cannot be reduced using $\rightsquigarrow$. We write $\rightsquigarrow^*$ for the reflexive-transitive closure of $\rightsquigarrow$.

The above reduction system captures the finitary aspects of evaluation. In general, however, since terms and meta-terms may be infinitely deep, evaluation must be seen as an infinite process. To account for this infinitary aspect, we use some familiar domain-theoretic ideas.

We write $\sqsubseteq$ for the evident syntactic orderings on meta-procedures and on ground meta-terms: thus, $T \sqsubseteq U$ iff T may be obtained from U by replacing zero or more subterms (possibly infinitely many) by $\bot$. It is easy to see that each set $\mathsf{SP}(\sigma)$ or $\mathsf{SP}^0(\sigma)$ forms a directed-complete partial order under $\sqsubseteq$, with least element $\lambda\vec{x}.\bot$.

By a *finite* term t we shall mean one generated by the following grammar, this time construed inductively:

$$\begin{array}{rrcl} \textit{Procedures:} & p, q & ::= & \lambda x_0 \ldots x_{r-1}.\, e \\ \textit{Expressions:} & d, e & ::= & \bot \ \mid\ n \ \mid\ \texttt{case}\ a\ \texttt{of}\ (0 \Rightarrow e_0 \mid \cdots \mid r-1 \Rightarrow e_{r-1}) \\ \textit{Applications:} & a & ::= & x\, q_0 \ldots q_{r-1} \end{array}$$

We regard finite terms as a subset of general terms by identifying the conditional branching $(0 \Rightarrow e_0 \mid \cdots \mid r-1 \Rightarrow e_{r-1})$ with

$$(0 \Rightarrow e_0 \mid \cdots \mid r-1 \Rightarrow e_{r-1} \mid r \Rightarrow \bot \mid r+1 \Rightarrow \bot \mid \ldots)\ .$$

We may now explain how a general meta-term T *evaluates* to a term $\ll T \gg$. This will in general be an infinite process, but we can capture the value of T as the limit of the finite portions that become visible at finite stages in the reduction. To this end, for any meta-term T we define

$$\Downarrow_{\mathrm{fin}} T \;=\; \{t \text{ finite } \mid \exists T'.\, T \rightsquigarrow^* T' \;\wedge\; t \sqsubseteq T'\}\ .$$

It is fairly easy to check that for any meta-term T, the set $\Downarrow_{\mathrm{fin}} T$ is directed with respect to $\sqsubseteq$, and also that each $\mathsf{SP}(\sigma)$ is directed-complete. We may therefore define $\ll T \gg$, the *value* of T, to be the ordinary term

$$\ll T \gg \;=\; \bigsqcup(\Downarrow_{\mathrm{fin}} T)\ .$$

Note in passing that the value $\ll G \gg$ of a ground meta-term G may be either an expression or an application. In either case, it is certainly a ground meta-term. It is also easy to see that $\ll \lambda\vec{x}.E \gg = \lambda\vec{x}. \ll E \gg$, and that if $T \rightsquigarrow^* T'$ then $\ll T \gg = \ll T' \gg$.

In the present paper, an important role will be played by the tracking of variable occurrences (and sometimes other subterms) through the course of evaluation. By inspection of the above rules for $\rightsquigarrow$, it is easy to see that if $T \rightsquigarrow T'$, then for any occurrence of a (free or bound) variable x within T', we can identify a unique occurrence of x within T from which it originates (we suppress the formal definition). The same therefore applies whenever $T \rightsquigarrow^* T'$. In this situation, we may say that the occurrence within T' is a *residual* of the one within T, or that the latter is the *origin* of the former. Note, however, that these relations are relative to a particular reduction path $T \rightsquigarrow^* T'$: there may be other paths for which the origin-residual relation is different.

Likewise, for any occurrence of x within $\ll T \gg$, we may pick some finite $t \sqsubseteq \ll T \gg$ containing this occurrence, and some $T' \sqsupseteq t$ with $T \rightsquigarrow^* T'$; this allows us to identify a unique occurrence of x within T that originates the given occurrence in $\ll T \gg$. It is routine to check that this occurrence in T will be independent of the choice of t and T' and of the chosen reduction path $T \rightsquigarrow^* T'$; we therefore have a robust origin-residual relationship between variable occurrences in T and those in $\ll T \gg$.

A fundamental result for NSPs is the *evaluation theorem*, which says broadly that the result of evaluating a meta-term is independent of the order of evaluating sub-expressions:

Theorem 4 (Evaluation theorem) *If $C[-_0, -_1, \ldots]$ is any meta-term context with countably many holes and $C[T_0, T_1, \ldots]$ is well-formed, then*

$$\ll C[T_0, T_1, \ldots] \gg \;=\; \ll C[\ll T_0 \gg, \ll T_1 \gg, \ldots] \gg\ .$$

The proof of this is non-trivial; see [12, Section 6.1.2].

One further piece of machinery will be useful: the notion of *hereditary η-expansion*, which enables us to convert a variable x into a procedure term (written x^η). The definition is by recursion on the type of x: if $x : \sigma_0, \ldots, \sigma_{r-1} \to \mathtt{N}$, then

$$x^\eta \;=\; \lambda z_0^{\sigma_0} \ldots z_{r-1}^{\sigma_{r-1}}.\, \mathtt{case}\; x z_0^\eta \ldots z_{r-1}^\eta \;\mathtt{of}\; (i \Rightarrow i)\;.$$

In particular, if $x : \mathtt{N}$ then $x^\eta = \lambda.\,\mathtt{case}\; x \;\mathtt{of}\; (i \Rightarrow i)$. The following useful properties of η-expansion are proved in [12, Section 6.1.3] (we assume the terms in question here are well typed):

$$\begin{array}{rcl} \ll x^\eta \vec{q} \gg & = & \mathtt{case}\; x\vec{q} \;\mathtt{of}\; (i \Rightarrow i)\;, \\ \ll \lambda \vec{y}.\, p\vec{y}^\eta \gg & = & p\;. \end{array}$$

The sets $\mathsf{SP}(\sigma)$ may now be made into a total applicative structure SP by defining

$$(\lambda x_0 \cdots x_r.e) \cdot q \;=\; \lambda x_1 \cdots x_r. \ll e[x_0 \mapsto q] \gg \;.$$

Clearly the sets $\mathsf{SP}^0(\sigma)$ are closed under this application operation, so we also obtain an applicative substructure SP^0. It is easy to check that application in SP is monotone and continuous with respect to $\sqsubseteq$. It is also shown in [12, Section 6.1.3] that both SP and SP^0 are typed *λ-algebras*: that is, they admit a compositional interpretation of typed λ-terms that validates β-equality. (The relevant interpretation of pure λ-terms is in fact given by three of the clauses from the interpretation of PCF$^\Omega$ as defined below.)

2.3 Interpretation of PCF in SP^0

A central role will be played by certain procedures $Y_\sigma \in \mathsf{SP}^0((\sigma \to \sigma) \to \sigma)$ which we use to interpret the PCF constants Y_σ (the overloading of notation will do no harm in practice). If $\sigma = \sigma_0, \ldots, \sigma_{r-1} \to \mathtt{N}$, we define $Y_\sigma = \lambda g^{\sigma \to \sigma}.F_\sigma[g]$, where $F_\sigma[g]$ is defined corecursively by:

$$F_\sigma[g] \;=\; \lambda x_0^{\sigma_0} \ldots x_{r-1}^{\sigma_{r-1}}.\, \mathtt{case}\; g\; (F_\sigma[g])\; x_0^\eta \cdots x_{r-1}^\eta \;\mathtt{of}\; (i \Rightarrow i)\;.$$

We may now give the standard interpretation of PCF$^\Omega$ in SP^0. To each PCF$^\Omega$ term $\Gamma \vdash M : \sigma$ we associate a procedure-in-environment $\Gamma \vdash [\![M]\!]_\Gamma : \sigma$ inductively as follows:

$$\begin{array}{rcl} [\![x^\sigma]\!]_\Gamma & = & x^{\sigma\eta} \\ [\![\widehat{n}]\!]_\Gamma & = & \lambda.n \\ [\![\mathit{suc}]\!]_\Gamma & = & \lambda x.\,\mathtt{case}\; x \;\mathtt{of}\; (i \Rightarrow i+1) \\ [\![\mathit{pre}]\!]_\Gamma & = & \lambda x.\,\mathtt{case}\; x \;\mathtt{of}\; (0 \Rightarrow 0 \mid i+1 \Rightarrow i) \\ [\![\mathit{ifzero}]\!]_\Gamma & = & \lambda xyz.\,\mathtt{case}\; x \;\mathtt{of}\; (0 \Rightarrow \mathtt{case}\; y \;\mathtt{of}\; (j \Rightarrow j) \\ & & \qquad\qquad\qquad\quad \mid\; i+1 \Rightarrow \mathtt{case}\; z \;\mathtt{of}\; (j \Rightarrow j)) \\ [\![Y_\sigma]\!]_\Gamma & = & Y_\sigma \end{array}$$

$$\begin{array}{rcl} [\![C_f]\!]_\Gamma & = & \lambda x.\,\mathtt{case}\ x\ \mathtt{of}\ (i \Rightarrow f(i)) \\ [\![\lambda x^\sigma.M]\!]_\Gamma & = & \lambda x^\sigma.[\![M]\!]_{\Gamma,x^\sigma} \\ [\![MN]\!]_\Gamma & = & [\![M]\!]_\Gamma \cdot [\![N]\!]_\Gamma \end{array}$$

(In the clause for C_f, we interpret $f(i)$ as $\bot$ whenever $f(i)$ is undefined.)

As is shown in [12], this interpretation is *adequate*, in the sense that $M \rightsquigarrow^* \widehat{n}$ iff $[\![M]\!] = \lambda.n$, and *universal*, in the sense that every element of $\mathsf{SP}^0(\sigma)$ is the denotation of some closed $M : \sigma$ in PCF$^\Omega$. It follows from these facts that the structure SF is a quotient of SP^0, and indeed is its *extensional collapse*; we shall write $\approx$ for the equivalence relation on SP^0 induced by the quotient map. It is also routine to check that the canonical interpretation of PCF$^\Omega$ in SF factors through the above interpretation in SP^0 via this map.

Our proof of Theorem 2 will proceed via a detailed analysis of the model SP^0. Specifically, in Sections 3 and 4 we will show the following:

Theorem 5 *For any $k \geq 1$, the elements $[\![Y_{k+1}]\!]$ and $[\![Y_{0\rightarrow(k+1)}]\!]$ in SP^0 are not* PCF$_k^\Omega$*-definable.*

For Y_{k+1}, this is the best that we have currently achieved, but in Section 5 we will go on to show that no $Z \approx [\![Y_{0\rightarrow(k+1)}]\!]$ can be PCF$_k^\Omega$-definable, which will establish Theorem 2.

Finally, we note that each set $\mathsf{SF}(\sigma)$ naturally carries an *observational ordering*, namely the partial order $\preceq$ given by

$$x \preceq x' \quad \text{iff} \quad \forall f \in \mathsf{SF}(\sigma \rightarrow \mathtt{N}),\, n \in \mathbb{N}.\ (f \cdot x = n \ \Rightarrow\ f \cdot x' = n)\ .$$

Clearly, the application operations $\cdot$ are monotone with respect to $\preceq$; moreover, it follows from an inequational version of the context lemma that the observational ordering on any $\mathsf{SF}(\sigma_0, \ldots, \sigma_{r-1} \rightarrow \mathbb{N})$ coincides with the pointwise ordering induced by the usual partial order on $\mathbb{N}_\bot$. We shall also use the symbol $\preceq$ for the preorder on each $\mathsf{SP}^0(\sigma)$ induced by $\preceq$ on SF; this coincides with the *observational preorder* on $\mathsf{SP}^0(\sigma)$ defined by

$$q \preceq q' \quad \text{iff} \quad \forall p \in \mathsf{SP}^0(\sigma \rightarrow \mathtt{N}),\, n \in \mathbb{N}.\ (p \cdot q = \lambda.n \ \Rightarrow\ p \cdot q' = \lambda.n)\ .$$

Note too that $q \approx q'$ iff $q \preceq q' \preceq q$. We shall also allow the use of the notations $\approx, \preceq$ for open terms (in the same environment) and indeed for meta-terms: e.g. $\vec{x} \vdash P \approx P'$ iff $\ll \lambda\vec{x}.P \gg \approx \ll \lambda\vec{x}.P' \gg$.

2.4 The embeddability hierarchy

The following result will play a central role in this paper:

Theorem 6 (Strictness of embeddability hierarchy) *In SF, no type $\overline{k+1}$ can be a* pseudo-retract *of any finite product $\Pi_i\sigma_i$ where each σ_i is of level $\leq k$. More formally, if z is a variable of type $\overline{k+1}$ and each x_i a variable of type σ_i, there cannot exist procedures*

$$z \vdash t_i : \sigma_i\ , \qquad \vec{x} \vdash r : \overline{k+1}$$

such that $z \vdash \ll r[\vec{x} \mapsto \vec{t}\,] \gg \succeq z^\eta$.

If in the above setting we had $z \vdash \ll r[\vec{x} \mapsto \vec{t}\,] \gg \approx z^\eta$, we would call $\overline{k+1}$ a *retract* of $\Pi_i \sigma_i$. In Appendix A we will show that the notions of retract and pseudo-retract coincide, since $z \vdash p \succeq z^\eta$ actually implies $z \vdash p \approx z^\eta$. However, this information will not be needed for our main proofs.

In our statement of Theorem 6, we have referred informally to a product $\Pi_i \sigma_i$ which we have not precisely defined (although our formal statement gives everything that is officially necessary). One may readily make precise sense of this product notation within the *Karoubi envelope* $\mathbf{K}(\mathsf{SF})$ as studied in [12, Chapter 4]: for instance, it is not hard to show that any finite product of level $\leq k$ types can be constructed as a retract of the pure type $\overline{k+1}$.

The proof of Theorem 6 appears in [12, Section 7.7], but for self-containedness we repeat it here.

Proof By induction on k. For the case $k = 0$, we note that $\mathtt{N} \to \mathtt{N}$ cannot be a pseudo-retract of any $\mathtt{N}^r$, since (for example) the set of maximal elements in $\mathsf{SF}(\mathtt{N} \to \mathtt{N})$ is of larger cardinality than the set of all elements of $\mathsf{SF}(\mathtt{N})^r$. (Alternatively, one can note that $\mathtt{N} \to \mathtt{N}$ is not a *retract* of $\mathtt{N}^r$, since the former contains infinite ascending chains while the latter does not, then use the method of Appendix A in the easy case $k = 1$ to show that any pseudo-retraction of the relevant type would be a retraction.)

Now assume the result for $k-1$, and suppose for contradiction that t_i, r exhibit $\overline{k+1}$ as a pseudo-retract of $\Pi_i \sigma_i$ where each σ_i is of level $\leq k$. Let $v = \ll r[\vec{x} \mapsto \vec{t}\,] \gg$, so that $\ll v[z \mapsto w] \gg \succeq w$ for any $w \in \mathsf{SP}^0(k+1)$. We first check that any v with this latter property must have the syntactic form $\lambda f^k.\mathtt{case}\ zp\ \mathtt{of}\ (\cdots)$ for some p of type $\overline{k}$. Indeed, it is clear that v does not have the form $\lambda f.n$ or $\lambda f.\bot$, and the only other alternative form is $\lambda f.\mathtt{case}\ fp'\ \mathtt{of}\ (\cdots)$. In that case, however, we would have

$$\ll v[z \mapsto \lambda x^k.0] \gg \cdot\, (\lambda y^{k-1}.\bot) \;=\; \bot \;,$$

contradicting $\ll v[z \mapsto \lambda x^k.0] \gg \cdot\, (\lambda y^{k-1}.\bot) \succeq (\lambda x.0)(\lambda y.\bot) = 0$.

We now focus on the subterm p in $v = \lambda f.\mathtt{case}\ zp\ \mathtt{of}\ (\cdots)$. The general direction of our argument will be to show that $\lambda f^k.p$ represents a function of type $\overline{k} \to \overline{k}$ that dominates the identity, and that moreover our two-stage construction of v via u can be used to split this into morphisms $\overline{k} \to \Pi_j \rho_j$ and $\Pi_j \rho_j \to \overline{k}$ where the ρ_j are of level $\leq k-1$, contradicting the induction hypothesis. An apparent obstacle to this plan is that z as well as f may appear free in p; however, it turns out that we still obtain all the properties we need if we specialize z (somewhat arbitrarily) to $\lambda x.0$.

Specifically, we claim that $\ll p[f \mapsto q, z \mapsto \lambda x.0] \gg \succeq q$ for any $q \in \mathsf{SP}^0(k)$. For suppose that $q \cdot s = n \in \mathbb{N}$ whereas $\ll p[f \mapsto q, z \mapsto \lambda x.0] \gg \cdot\, s \neq n$ for some $s \in \mathsf{SP}^0(k-1)$. Take $w = \lambda g.\mathtt{case}\ gs\ \mathtt{of}\ (n \Rightarrow 0)$, so that $w \cdot q' = \bot$ whenever $q' \cdot s \neq n$. Then $w \preceq \lambda x.0$ pointwise, so we have $\ll p[f \mapsto q, z \mapsto w] \gg \cdot\, s \neq n$ by the monotonicity of SF (see the end of Section 2.3). By the definition of w, it follows that $\ll (zp)[f \mapsto q, z \mapsto w] \gg = \bot$, whence $\ll v[z \mapsto w] \gg \cdot\, q = \bot$, whereas $w \cdot q = 0$, contradicting $\ll v[z \mapsto w] \gg \succeq w$. We have thus shown that $\lambda f. \ll p[z \mapsto \lambda x.0] \gg \succeq id_k$.

We next show how to split the function represented by this procedure so as to go through some $\Pi_j \rho_j$ as above. Since $\ll r[\vec{x} \mapsto \vec{t}] \gg = \lambda f.\mathtt{case}\ zp\ \mathtt{of}\ (\cdots)$, we have that $r[\vec{x} \mapsto \vec{t}]$ reduces in finitely many steps to a head normal form $\lambda f.\mathtt{case}\ zP\ \mathtt{of}\ (\cdots)$ where $\ll P \gg = p$. By working backward through this reduction sequence, we may locate the ancestor within $r[\vec{x} \mapsto \vec{t}]$ of this head occurrence of z. Since r is closed, this occurs within some t_i, and clearly it must appear as the head of some subterm $\mathtt{case}\ zP'\ \mathtt{of}\ (\cdots)$ where P is a substitution instance of P'.[1] Now since t_i has type σ_i of level $\leq k$, and $z : \overline{k+1}$ is its only free variable, it is easy to see that all *bound* variables within t_i have pure types of level $< k$. Let $h_0, h_1, \ldots$ denote the finitely many bound variables that are in scope at the relevant occurrence of zP', and suppose each h_j has type ρ_j of level $< k$. By considering the form of the head reduction sequence $r[\vec{x} \mapsto \vec{t}] \rightsquigarrow_h^* \lambda f.\mathtt{case}\ zP\ \mathtt{of}\ (\cdots)$, we see that $P = P'[\vec{h} \mapsto \vec{H}]$ where each $H_j : \rho_j$ contains at most f and z free.

Writing * for the substitution $[z \mapsto \lambda x.0]$, define procedures

$$f^k \vdash t'_j = H_j^* : \rho_j \ , \qquad \vec{h} \vdash r' = P'^* : \overline{k} \ .$$

Then $r' \circ t'$ coincides with the term $\ll \lambda f.\, P^* \gg = \lambda f. \ll p^* \gg$, which dominates the identity as shown above. Thus $\overline{k}$ is a pseudo-retract of $\Pi_j \rho_j$, which contradicts the induction hypothesis. So $\overline{k+1}$ is not a pseudo-retract of $\Pi_i \sigma_i$ after all, and the proof is complete. $\square$

As an aside, we remark that for several extensions of PCF studied in the literature, the situation is completely different, in that the corresponding fully abstract and universal models possess a *universal* simple type υ of which all simple types are retracts. It follows easily in these cases that one can indeed bound the type levels of recursion operators without loss of expressivity. For example:

- In the language PCF + *por* + *exists* considered by Plotkin [16], the type $\mathtt{N} \rightarrow \mathtt{N}$ is universal, and the proof of this shows that every program in this language is observationally equivalent to one in PCF_1 + *por* + *exists* (this latter fact was already noted in [16]).
- In the language SPCF, or PCF + *catch*, introduced by Cartwright and Felleisen [5], the type $\mathtt{N} \rightarrow \mathtt{N}$ is again universal, and again the sublanguage PCF_1 + *catch* has the same expressive power.
- In the language PCF+H of Longley [9], the type $(\mathtt{N} \rightarrow \mathtt{N}) \rightarrow \mathtt{N}$ is universal, but even here, all constants Y_σ with $\mathrm{lv}(\sigma) > 1$ are dispensable.

Further details of each the above scenarios may be found in [12]. These facts may help to give some sense of why a cheap proof of our present results for PCF should not be expected.

[1]The reader wishing to see a more formal justification for this step may consult the proof of Lemma 22(i) below.

2.5 The computational power of PCF_1

Our main results will in effect present a hierarchy of sublanguages of PCF with PCF_1 as the bottom rung. However, there are also other sublanguages of interest that are either weaker than or incomparable with PCF_1. We here offer a brief summary of some known results in order to situate our main theorems within a broader picture.

That PCF_1 surpasses PCF_0 in power is true but uninteresting, since the latter is an extremely weak language that does not even define addition. More representative is that PCF_1 is strictly stronger than the language $\mathrm{T}_0 + \mathit{min}$, where T_0 (a fragment of Gödel's System T) is the λ-calculus with constants

$$\widehat{0} \;: \mathtt{N}\,, \qquad \mathit{suc} \;: \mathbb{N} \to \mathbb{N}\,, \qquad \mathit{rec}_0 \;: \mathtt{N} \to (\mathtt{N} \to \mathtt{N} \to \mathtt{N}) \to (\mathtt{N} \to \mathtt{N})\,,$$

(the last of these being the standard operator for primitive recursion), and *min* is the classical minimization operator of type $(\mathtt{N} \to \mathtt{N}) \to \mathtt{N}$. On the one hand, it is an easy exercise to define both rec_0 and *min* in PCF_1; on the other hand, Berger [4] shows that the functional $F : (\mathtt{N} \to \mathtt{N} \to \mathtt{N}) \to (\mathtt{N} \to \mathtt{N})$ recursively defined by

$$F\,h\,n \;=\; h\,n\,(F\,h\,(n+1))\,,$$

whilst readily expressible in PCF_1, is not definable in $\mathrm{T}_0 + \mathit{min}$.

This situation is revisited in [12] from the perspective of substructures of SP^0: it is shown that the language $\mathrm{T} + \mathit{min}$ can be modelled within the substructure of *left-well-founded* procedures, but that the above functional F is not representable by such a procedure, so that F is not definable even in $\mathrm{T} + \mathit{min}$. At third order, there are even 'hereditarily total' functionals definable in PCF_1 but not in $\mathrm{T} + \mathit{min}$, one example being the well-known *bar recursion* operator (see [11]).

Even weaker than $\mathrm{T}_0 + \mathit{min}$ is the language of *(strict) Kleene primitive recursion plus minimization*, denoted by $\mathsf{Klex}^{\mathrm{min}}$ in [12]. This is in fact equivalent in power to a language Iter_1 which we will mention in Section 6. It is shown in [12] that the computational power of this language corresponds precisely to that of computable *left-bounded* procedures (which form a smaller class than the left-well-founded ones). It seems reasonable to regard $\mathsf{Klex}^{\mathrm{min}}$ as representing the weakest higher-order computability notion of natural interest that is still Turing complete.

3 A substructure of SP^0 for PCF_k

For the remainder of the paper, we take k to be some fixed natural number greater than 0.

We shall define a certain substructure A^0_k of SP^0, whose elements we call the *k-acceptable* procedures, in such a way that A^0_k provides a model for PCF^{Ω}_k but excludes Y_{k+1}. The substructure A^0_k will be constructed inductively by means of a system of term-forming operations that generate these procedures. Of course, this must be done in a carefully controlled way, since we wish to generate Y_k but not Y_{k+1}.

To motivate the definition of A^0_k, let us try to explain informally what is the characteristic of Y_{k+1} that A^0_k is designed to exclude. Recall that for any $l > 0$, the NSP Y_l is defined as $\lambda g^{l \to l}.\, F_l[g]$, where $F_l[g]$ is the procedure defined corecursively by

$$F_l[g] \;=\; \lambda x^{l-1}.\, \mathtt{case}\; g\,(F_l[g])\, x^\eta \;\mathtt{of}\; (i \Rightarrow i)\;.$$

The manifest difference between Y_{k+1} and Y_k, then, is that Y_{k+1} involves an infinite sequence of nested calls to a variable g of type level $k+2$, whereas Y_k does not.

One's first thought might therefore be to try and show that no procedure involving an infinite nesting of this kind can be constructed via PCF$_k$ alone. As it stands, however, this is not the case. Suppose, for example, that $up_k : k \to k+1$ and $down_k : k+1 \to k$ are PCF$_0$ terms defining a standard retraction $k \lhd k+1$. Specifically, let us inductively define

$$\begin{array}{rclrcl} up_0 & = & \lambda x^0.\lambda z^0.x\;, & down_0 & = & \lambda y^1.\, y\,\widehat{0}\;, \\ up_{k+1} & = & \lambda x^{k+1}.\lambda z^{k+1}.\, x(down_k\; z)\;, & down_{k+1} & = & \lambda y^{k+2}.\lambda w^k.\, y(up_k\; w)\;. \end{array}$$

Now consider the PCF$_k$ program

$$Z_{k+1} \;=\; \lambda g : (k+1) \to (k+1).\; up_k\,(Y_k\,(down_k \circ g \circ up_k))\;.$$

A simple calculation shows that the NSPs for Y_{k+1} and Z_{k+1} are superficially very similar in form, both involving an infinite sequence of nested calls to $g : (k+1) \to (k+1)$. (These NSPs are shown schematically in Figure 3 for the case $k = 2$.) We will therefore need to identify some more subtle property of NSPs that differentiates between Y_{k+1} and Z_{k+1}.

The intuitive idea will be that in the NSP for Z_{k+1}, the full potency of g as a variable of type $k+1 \to k+1$ is not exploited, since both the input and output to g are 'funnelled' through the simpler type k. (The force of Theorem 6 above is that the type k cannot fully represent the structure of the type $k+1$.) Broadly speaking, we shall construct a model A^0_k which admits infinite nesting for variables of high type provided that the resulting information is funnelled sufficiently often through a type of level $\leq k$, but not otherwise. This model with then contain the NSP for Z_{k+1}, but not that for Y_{k+1}.

We shall in fact define a set $\mathcal{A}_k$ of meta-procedures-in-environment $\Gamma \vdash P : \sigma$, then take A^0_k to consist of the closed normal forms in $\mathcal{A}_k$. (The larger set $\mathcal{A}_k$ will play a key role in the proof that A^0_k is closed under application.) In order to formalize the above idea of funnelling through types of level $\leq k$, our construction will make use the following notion of *k-plugging* for meta-terms. Throughout the paper, we shall use Greek capitals Γ, Δ for arbitrary environments, and Roman capitals Z, V, X for lists of variables that are constrained to be of type level $\leq k$.

Definition 7 (Plugging) *Suppose given the following data:*

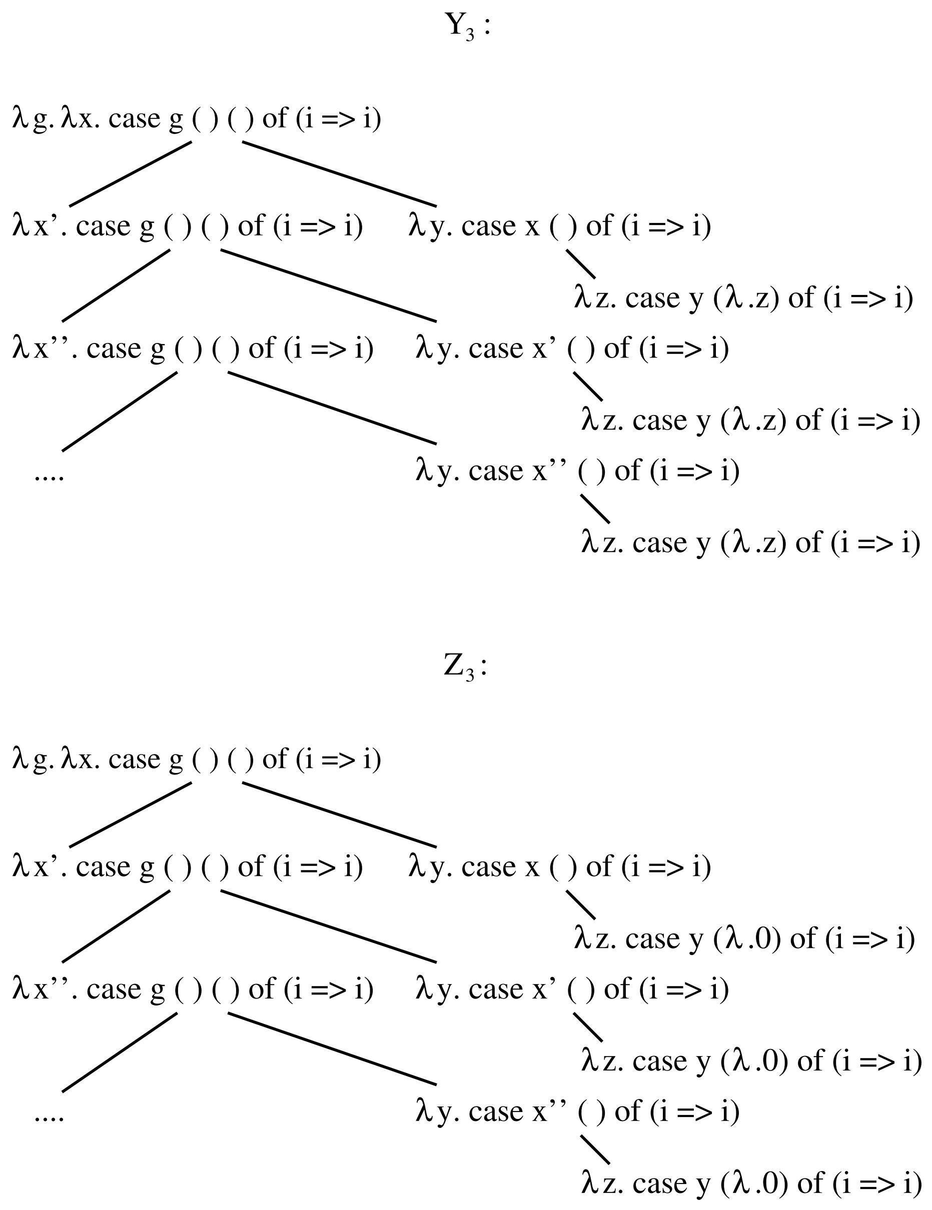
Y3 :
λg. λx. case g ( ) ( ) of (i => i)
λx'. case g ( ) ( ) of (i => i)
λy. case x ( ) of (i => i)
λz. case y (λ.z) of (i => i)
λx''. case g ( ) ( ) of (i => i)
λy. case x' ( ) of (i => i)
λz. case y (λ.z) of (i => i)
....
λy. case x'' ( ) of (i => i)
λz. case y (λ.z) of (i => i)
Z3 :
λg. λx. case g ( ) ( ) of (i => i)
λx'. case g ( ) ( ) of (i => i)
λy. case x ( ) of (i => i)
λz. case y (λ.0) of (i => i)
λx''. case g ( ) ( ) of (i => i)
λy. case x' ( ) of (i => i)
λz. case y (λ.0) of (i => i)
....
λy. case x'' ( ) of (i => i)
λz. case y (λ.0) of (i => i)

Figure 1: The NSPs for Y_3 and Z_3. Here $\lambda.z$ abbreviates $\lambda.\,\texttt{case}\ z\ \texttt{of}\ (i \Rightarrow i)$.

- *a variable environment Γ (i.e. a finite list of typed variables),*
- *a finite list Z of 'plugging variables' z of level $\leq k$, disjoint from Γ,*[2]
- *a* root meta-expression $\Gamma, Z \vdash E$,
- *for each variable $z^\sigma \in Z$, a meta-term $\Gamma, Z \vdash \xi(z) : \sigma$.*

We may assume here by α-conversion that no variables in Γ, Z appear bound *by a λ-abstraction in E or in any of the $\xi(z)$.*

In this situation, we define the (k-)plugging $\Pi_{\Gamma,Z}(E,\xi)$ *(often abbreviated to $\Pi(E,\xi)$) to be the meta-term obtained from E by repeatedly expanding variables $z \in Z$ to $\xi(z)$. More formally, writing T° for the meta-term obtained from T by replacing all subterms $z\vec{Q}$ with $z \in Z$ by $\perp$, we may define*

$$\begin{array}{rcl} \Pi^0(E,\xi) & = & E \ , \\ \Pi^{m+1}(E,\xi) & = & \Pi^m(E,\xi)[z \mapsto \xi(z) \ \textit{for all}\ z \in Z] \ , \\ \Pi(E,\xi) & = & \bigsqcup_m \Pi^m(E,\xi)^\circ \ , \end{array}$$

where $\bigsqcup$ denotes supremum with respect to the syntactic order on meta-terms.

It is easy to see that $\Pi_{\Gamma,Z}(E,\xi)$ is well-typed in environment Γ.

Lemma 8 *Under the conditions of the above definition, we have*

$$\ll \Pi_{\Gamma,Z}(E,\xi) \gg \ = \ \ll \Pi_{\Gamma,Z}(\ll E \gg, \ll \xi \gg) \gg \ ,$$

where $\ll \xi \gg$ denotes the mapping $z \mapsto \ll \xi(z) \gg$.

Proof Since $\ll - \gg$ is continuous, this will follow from the general fact that $\ll \Pi^m(E,\xi)^\circ \gg = \ll \Pi^m(\ll E \gg, \ll \xi \gg)^\circ \gg$ for each $m \in \mathbb{N}$ and all E, ξ. First, it follows easily from Definition 7 that

$$\begin{array}{rcl} \Pi^0(E,\xi)^\circ & = & E^\circ \ , \\ \Pi^{m+1}(E,\xi)^\circ & = & E[z \mapsto \Pi^m(\xi(z),\xi)^\circ] \text{ for all } z \in Z \ . \end{array}$$

We now argue by induction on m. When $m = 0$, the desired equation holds because $\ll E^\circ \gg = \ll \ll E \gg^\circ \gg$. For the induction step, we have

$$\begin{array}{rcl} \ll \Pi^{m+1}(E,\xi)^\circ \gg & = & \ll E[z \mapsto \Pi^m(\xi(z),\xi)^\circ] \gg \\ & = & \ll \ll E \gg \ [z \mapsto \ll \Pi^m(\xi(z),\xi)^\circ \gg] \gg \\ & & \quad \text{by the evaluation theorem} \\ & = & \ll \ll E \gg \ [z \mapsto \ll \Pi^m(\ll \xi(z) \gg, \ll \xi \gg)^\circ \gg] \gg \\ & & \quad \text{by the induction hypothesis} \\ & = & \ll \ll E \gg \ [z \mapsto \Pi^m(\ll \xi(z) \gg, \ll \xi \gg)^\circ] \gg \\ & & \quad \text{by the evaluation theorem} \\ & = & \ll \Pi^{m+1}(\ll E \gg, \ll \xi \gg)^\circ \gg \ , \end{array}$$

[2]The restriction that Z is finite is not very essential—the concept of plugging makes equally good sense for infinite Z. However, in the context of the present paper, allowing variable environments to be infinite would entail some additional complication in our type system.

which completes the proof. □

We now give our central definition specifying the class $\mathcal{A}_k$ of interest. This will be an inductive definition via formation rules somewhat similar in character to those used in [12, Section 6.3] to generate *left-well-founded* (LWF) procedures. However, there is an important difference: for LWF procedures, a direct structural characterization is also available (they are the procedures containing no infinite descending chains of application subterms), whereas for $\mathcal{A}_k$ the inductive definition will be the only handle on the class that we have. On the one hand, this style of definition enables us to capture the subtle structural difference between the procedures Y_{k+1} and Z_{k+1} shown above; on the other hand, the absence of a direct characterization means that some non-trivial effort will be needed to show that Y_{k+1} is *not* present in $\mathcal{A}_k$ (see Section 4). As in the grammar above, we let P, Q, R range over meta-procedures, D, E over meta-expressions, and G over ground meta-terms.

Definition 9 (Acceptable meta-terms) *(i) The class $\mathcal{A}_k$ of (k-)*acceptable *meta-terms-in-environment is generated inductively by means of the following clauses:*

1. $\Gamma \vdash \bot \in \mathcal{A}_k$ *and* $\Gamma \vdash n \in \mathcal{A}_k$ *for any* $n \in \mathbb{N}$.
2. *If* $\Gamma, \vec{x} \vdash E \in \mathcal{A}_k$, *then* $\Gamma \vdash (\lambda \vec{x}.E) \in \mathcal{A}_k$.
3. *If* $x : \vec{\sigma} \to \mathtt{N} \in \Gamma$ *and* $\Gamma \vdash Q_i : \sigma_i \in \mathcal{A}_k$ *for each* i, *then* $\Gamma \vdash x\vec{Q} \in \mathcal{A}_k$.
4. *If* $\Gamma \vdash G \in \mathcal{A}_k$ *and* $\Gamma \vdash E_i \in \mathcal{A}_k$ *for each* i,
   then $\Gamma \vdash \mathtt{case}\ G\ \mathtt{of}\ (i \Rightarrow E_i) \in \mathcal{A}_k$.
5. *If* $\Gamma \vdash P : \vec{\sigma} \to \mathtt{N} \in \mathcal{A}_k$ *and* $\Gamma \vdash Q_i : \sigma_i \in \mathcal{A}_k$ *for each* i, *then* $\Gamma \vdash P\vec{Q} \in \mathcal{A}_k$.
6. Plugging rule: *If* Γ, Z, E, ξ *are as in Definition 7 (with respect to* k*), where* E *is some normal form expression* $\Gamma, Z \vdash e \in \mathcal{A}_k$, *and each* $\xi(z)$ *is a normal form procedure* $\Gamma, Z \vdash \xi(z) \in \mathcal{A}_k$, *then* $\Gamma \vdash \ll \Pi_{\Gamma,Z}(e, \xi) \gg \in \mathcal{A}_k$.

(ii) For each type σ *and environment* Γ, *let* $\mathsf{A}_k(\Gamma, \sigma)$ *consist of all (normal-form) procedures* $p \in \mathsf{SP}(\sigma)$ *such that* $\Gamma \vdash p \in \mathcal{A}_k$, *and let* $\mathsf{A}^0_k(\sigma) = \mathsf{A}_k(\emptyset, \sigma)$, *so that* $\mathsf{A}^0_k(\sigma) = \mathsf{A}_k(\sigma) \cap \mathsf{SP}^0(\sigma)$.

Note that the k-acceptable meta-terms that can be constructed without recourse to the plugging rule are exactly the *well-founded* meta-terms as studied in Section 6.3 of [12]. Note too that the generation of *normal-form* k-acceptable terms is self-contained, in that if t is a normal form, any derivation of $\Gamma \vdash t \in \mathcal{A}_k$ consists entirely normal forms $\Gamma' \vdash t' \in \mathcal{A}_k$ (and hence does not involve rule 5).

To give some intuition for the above definition, the idea is that if M is a closed term of PCF^{Ω}_k, then the NSP $[\![M]\!]$ will be an element of $\mathcal{A}_k$ whose construction via the clauses of Definition 9 broadly parallels the syntactic structure of M. In particular, for every occurrence of an operator Y_σ within M, say at a position in which the variables in scope are those of Γ, the construction of $[\![M]\!]$

will involve an application of the plugging rule in environment Γ. Thus, if M involves multiple occurrences of operators Y_σ, then multiple applications of the plugging rule (perhaps with different Γ) may be needed to construct $[\![M]\!]$. On the other hand, if the inductive definition of our class $\mathcal{A}_k$ were to mirror the syntactic structure of PCF^Ω terms *too* closely, it may become too hard to show that this class excludes Y_{k+1}. Thus, as an extremal case, one might consider the substructure of SP^0 that consists by definition of the PCF^Ω_k-denotable procedures, but clearly this would afford no reduction of our original problem. Our actual definition of A_k can be seen as a compromise that avoids this extreme whilst still accurately characterizing the power of PCF^Ω_k.

Next, we proceed to show that the sets $\mathsf{A}_k(\sigma)$ constitute a well-behaved substructure of SP that suffices for the interpretation of PCF^Ω_k. We begin with a simple analysis of the leaf structure of meta-terms that will play a role in our treatment of `case` expressions.

Definition 10 *(i) The set of* rightward *(occurrences of)* numeral leaves *within a meta-term T is defined inductively by means of the following clauses:*

1. *A meta-term n is a rightward numeral leaf within itself.*
2. *Every rightward numeral leaf within E is also one within $\lambda\vec{x}.E$.*
3. *Every rightward numeral leaf in each E_i is also one in* `case` *G* `of` *$(i \Rightarrow E_i)$.*
4. *Every rightward numeral leaf within P is also one within $P\vec{Q}$.*

(Note that there are no rightward numeral leaves within a meta-term $x\vec{Q}$. Note too that clause 4 is not required for computing the rightward leaves of a meta-procedure P or meta-expression E.)

(ii) If T is a meta-term and E_i is a meta-expression for each i, we write $T[i \mapsto E_i]$ for the result replacing each rightward leaf occurrence i in T by E_i.

Proposition 11 $\ll$ `case` d `of` $(i \Rightarrow e_i) \gg \; = d[i \mapsto e_i]$ *for any expressions d, e_i.*

Proof Define a 'truncation' operation $-^{(c)}$ on normal-form expressions for each $c \in \mathbb{N}$ as follows:

$$\begin{aligned} n^{(c)} \;=\; n\,, \quad \bot^{(c)} &= \bot\,, \\ \texttt{case}\; g \;\texttt{of}\; (i \Rightarrow e_i)^{(0)} &= \bot\,, \\ \texttt{case}\; g \;\texttt{of}\; (i \Rightarrow e_i)^{(c+1)} &= \texttt{case}\; g \;\texttt{of}\; (i \Rightarrow e_i^{(c)})\,. \end{aligned}$$

Then clearly $d = \bigsqcup_c d^{(c)}$ and $d[i \mapsto e_i] = \bigsqcup_c d^{(c)}[i \mapsto e_i]$. Moreover, we may show by induction on c that

$$\ll \texttt{case}\; d^{(c)} \;\texttt{of}\; (i \Rightarrow e_i) \gg \;=\; d^{(c)}[i \mapsto e_i]\,.$$

The case $c = 0$ is trivial since $d^{(0)}$ can only have the form n or $\bot$. For the induction step, the situation for $d = n, \bot$ is trivial, so let us suppose $d =$

$\texttt{case}\ g\ \texttt{of}\ (j \Rightarrow f_j)$. Then

$$\begin{array}{ll}
& \ll \texttt{case}\ d^{(c+1)}\ \texttt{of}\ (i \Rightarrow e_i) \gg \\
= & \ll \texttt{case}\ (\texttt{case}\ g\ \texttt{of}\ (j \Rightarrow f_j^{(c)}))\ \texttt{of}\ (i \Rightarrow e_i) \gg \\
= & \texttt{case}\ g\ \texttt{of}\ (j \Rightarrow \ll \texttt{case}\ f_j^{(c)}\ \texttt{of}\ (i \Rightarrow e_i) \gg) \\
= & \texttt{case}\ g\ \texttt{of}\ (j \Rightarrow (f_j^{(c)}[i \mapsto e_i]))\ \ \text{by induction hypothesis} \\
= & (\texttt{case}\ g\ \texttt{of}\ (j \Rightarrow f_j^{(c)}))[i \mapsto e_i] \\
= & d^{(c+1)}[i \mapsto e_i]\ .
\end{array}$$

Since $\ll - \gg$ is continuous, the proposition follows by taking the supremum over c. $\square$

Lemma 12 *(i) If $T \rightsquigarrow T'$ then the rightward numeral leaves in T' are exactly the residuals in T' of rightward numeral leaves in T.*

(ii) The rightward numeral leaves in $\ll T \gg$ are precisely the residuals of rightward numeral leaves in T.

(iii) $\ll T[n \mapsto E_n] \gg = \ll T \gg [n \mapsto \ll E_n \gg]$.

Proof (i) Easy by induction on the generation of the one-step reduction relation $\rightsquigarrow$ as given in Section 2.

(ii) If n is any rightward numeral leaf in $\ll T \gg$, then by the definition of $\ll T \gg$, we may take a finite normal form $t \sqsubseteq \ll T \gg$ containing n and a finite reduction sequence $T \rightsquigarrow^* T'$ such that $t \sqsubseteq T'$. By considering the chain of subterms witnessing that n is a rightward leaf in $\ll T \gg$, we see that the corresponding occurrence of n is also a rightward leaf in t and in T'. So by (i), this occurrence is a residual in T' of a rightward leaf n in T; and since this occurrence falls within t, this is to say that it is a residual in $\ll T \gg$ of this leaf in T. Moreover, this argument is easily reversible, so that any residual in $\ll T \gg$ of a rightward leaf n in T is itself a rightward leaf in $\ll T \gg$.

(iii) Let $e_n = \ll E_n \gg$ for each n; then by the evaluation theorem we have $\ll T[n \mapsto E_n] \gg = \ll T[n \mapsto e_n] \gg$. Let us write ${}^\bullet$ for the operation $[n \mapsto e_n]$. As in (i), we see that any finite reduction $T \rightsquigarrow T_1 \rightsquigarrow \cdots \rightsquigarrow T_s$ instantiates to a reduction $T^\bullet \rightsquigarrow T_1^\bullet \rightsquigarrow \cdots \rightsquigarrow T_s^\bullet$; in particular, the operation ${}^\bullet$ can never 'block' one of these reduction steps, since no residual of a rightward leaf n can occur as the branching condition in a subterm $\texttt{case}\ n\ \texttt{of}\ (\cdots)$. Conversely, any redex in any meta-term $U^\bullet$ is an instantiation of a redex in U, so clearly any finite reduction sequence for $T^\bullet$ is the ${}^\bullet$-instantiation of a reduction sequence for T.

We now see that if $T \rightsquigarrow^* T'$ and $t \sqsubseteq T'$ is a normal form, then $T^\bullet \rightsquigarrow^* T'^\bullet$ and $t^\bullet \sqsubseteq T'^\bullet$ is a normal form; and since ${}^\bullet$ commutes with $\bigsqcup$, this shows that $\ll T \gg^\bullet \sqsubseteq \ll T^\bullet \gg$. Conversely, if $T^\bullet \rightsquigarrow^* U$ and $u \sqsubseteq U$ is a normal form, then $U = T'^\bullet$ for some $T \rightsquigarrow^* T'$, and there is some $t \sqsubseteq T'$ with $u \sqsubseteq t^\bullet$; this shows that $\ll T^\bullet \gg \sqsubseteq \ll T \gg^\bullet$. Thus $\ll T[n \mapsto E_n] \gg = \ll T^\bullet \gg = \ll T \gg^\bullet$ as required. $\square$

Lemma 13 *If* $\Gamma \vdash T \in \mathcal{A}_k$ *and* $\Gamma \vdash e_n \in \mathcal{A}_k$ *for each* n*, where the* e_n *are normal-form expressions, then* $\Gamma \vdash T[n \mapsto e_n] \in \mathcal{A}_k$*.*

Proof By induction on the generation of $T \in \mathcal{A}_k$ as in Definition 9. The cases for clauses 1–5 are easy. For clause 6, suppose $\ll D \gg = \ll \Pi_{\Gamma,Z}(e,\xi) \gg$ for some Γ, Z, e, ξ satisfying the conditions of the plugging rule, so that $\Gamma \vdash D \in \mathcal{A}_k$; we wish to show that $\Gamma \vdash D[n \mapsto e_n] \in \mathsf{A}_k$. But by Lemma 12(iii) we have

$$\begin{array}{rcl} \ll D[n \mapsto e_n] \gg & = & \ll D \gg [n \mapsto e_n] \\ & = & \ll \Pi_{\Gamma,Z}(e,\xi) \gg [n \mapsto e_n] \\ & = & \ll \Pi_{\Gamma,Z}(e,\xi)\,[n \mapsto e_n] \gg \;, \end{array}$$

and it is easy to see that $\Pi_{\Gamma,Z}(e,\xi)[n \mapsto e_n] = \Pi_{\Gamma,Z}(e[n \mapsto e_n],\xi)$, since all rightward leaves in $\Pi_{\Gamma,Z}(e,\xi)$ must originate from e. (Specifically, every occurrence of a plugging variable z within e must be at the head of some subexpression $\texttt{case}\ z\vec{Q}\ \texttt{of}\ (\cdots)$, so that the expansion of z does not contribute to the set of rightward leaf nodes.) But $e[n \mapsto e_n]$ is a normal form and $\Gamma, Z \vdash e[n \mapsto e_n] \in \mathcal{A}_k$ by the induction hypothesis; hence the plugging $\Pi_{\Gamma,Z}(e[n \mapsto e_n],\xi)$ itself satisfies the conditions of rule 6, and this plugging now witnesses that $\Gamma \vdash D[n \mapsto e_n] \in \mathcal{A}_k$. $\Box$

The core of our analysis is the proof of the following lemma. As in [12], we say a meta-term T is of *order* $\delta \in \mathbb{N}$ if for every β-redex $P\vec{Q}$ within T, P has type level $\leq \delta$ (so that the Q_i each have type level $< \delta$). Thus, a meta-term of order 0 may contain only nullary redexes $(\lambda.E)$; contracting each of these to E immediately yields a normal form.

Lemma 14 *If* T *is of finite order and* $\Gamma \vdash T \in \mathcal{A}_k$*, then* $\Gamma \vdash \ll T \gg \in \mathcal{A}_k$*.*

Proof We will establish the following two claims by a simultaneous induction on $\delta \in \mathbb{N}$:

1. For all T of order δ, if $\Gamma \vdash T \in \mathcal{A}_k$ then $\Gamma \vdash \ll T \gg \in \mathcal{A}_k$.
2. For all normal-form t and all $\vec{q}$ of type level $\leq \delta$, if $\Gamma, \vec{x} \vdash t \in \mathcal{A}_k$ and $\Gamma \vdash \vec{q} \in \mathcal{A}_k$ then $\Gamma \vdash \ll t[\vec{x} \mapsto \vec{q}] \gg \in \mathcal{A}_k$.

We will show that both claims hold for δ assuming they hold for all $\delta' < \delta$, so that there is no need for a separate base case. For each δ, we first establish claim 1, then use this to help establish claim 2.

For claim 1, we argue by an inner induction on the generation of $\Gamma \vdash T \in \mathcal{A}_k$ via the clauses of Definition 9. The cases for meta-terms n and $\perp$ are trivial, and those for meta-terms $\lambda\vec{x}.E$ and $x\vec{Q}$ are straightforward, bearing in mind that $\ll \lambda\vec{x}.E \gg = \lambda\vec{x}. \ll E \gg$ and $\ll xQ_0Q_1\cdots \gg = x \ll Q_0 \gg \ll Q_1 \gg \cdots$. For meta-terms $\Gamma \vdash \texttt{case}\ G\ \texttt{of}\ (n \Rightarrow E_n)$, let $g = \ll G \gg$ and $e_n = \ll E_n \gg$ for each n, so that $\Gamma \vdash g, e_n \in \mathcal{A}_k$ by the inner induction hypothesis. If g is an application $x\vec{q}$ then $\ll \texttt{case}\ G\ \texttt{of}\ (n \Rightarrow E_n) \gg = \texttt{case}\ g\ \texttt{of}\ (n \Rightarrow e_n) \in \mathcal{A}_k$ by clause 4 of Definition 9. Otherwise, g is a normal-form expression, and

$$\ll \texttt{case}\ G\ \texttt{of}\ (n \Rightarrow E_n) \gg \;=\; \ll \texttt{case}\ g\ \texttt{of}\ (n \Rightarrow e_n) \gg \;=\; g[n \mapsto e_n]$$

by Proposition 11. But $\Gamma \vdash g[n \mapsto e_n] \in \mathcal{A}_k$ by Lemma 13.

The case for applications $P\vec{Q}$ follows a line of argument familiar from [12, Section 6.3]. Suppose $\Gamma \vdash P\vec{Q} \in \mathcal{A}_k$ is of order δ, and let $p = \ll P \gg$, $q_i = \ll Q_i \gg \in \mathcal{A}_k$ so that $\Gamma \vdash p, q_i \in \mathcal{A}_k$ by the inner induction hypothesis. Write p as $\lambda\vec{x}.e$; then $\Gamma, \vec{x} \vdash e \in \mathcal{A}_k$, since clause 2 of Definition 9(i) provides the only possible way to generate $\Gamma \vdash p \in \mathcal{A}_k$. If $\vec{Q}$ and $\vec{x}$ are empty, this is all we require. Otherwise, we must have that $\delta > 0$, and that each q_i has type level $< \delta$, since $P\vec{Q}$ is of order δ. Hence $\Gamma \vdash \ll e[\vec{x} \mapsto \vec{q}] \gg \in \mathcal{A}_k$ by claim 2 for $\delta - 1$. But $\ll e[\vec{x} \mapsto \vec{q}] \gg = \ll (\lambda\vec{x}.e)\vec{q} \gg = \ll P\vec{Q} \gg$ using the evaluation theorem, so $\Gamma \vdash \ll P\vec{Q} \gg \in \mathcal{A}_k$ as required.

The case for the plugging rule itself is trivial, since this rule only generates normal forms in $\mathcal{A}_k$. This completes the proof of claim 1 for δ.

To show claim 2 for δ, we again argue by an inner induction on the generation of $\Gamma, \vec{x} \vdash t \in \mathcal{A}_k$, writing * for the substitution $[\vec{x} \mapsto \vec{q}]$. The cases for clauses 1 and 2 are trivial, and clause 5 does not arise here as t is a normal form.

For clause 3, suppose $t = yp_0 \ldots p_{r-1}$. If $y \notin \vec{x}$, then $\ll t^* \gg = y \ll p_0^* \gg \cdots \ll p_{r-1}^* \gg$ which is clearly in $\mathcal{A}_k$ by the induction hypothesis. If $y = x_i$, then $\ll t^* \gg = \ll q_i p_0^* \ldots p_{r-1}^* \gg = \ll G \gg$ where $G = q_i \ll p_0^* \gg \cdots \ll p_{r-1}^* \gg$, using the evaluation theorem. But $\Gamma \vdash G \in \mathcal{A}_k$ by the induction hypothesis and clause 5, and clearly G is of order δ, so $\Gamma \vdash \ll G \gg \in \mathcal{A}_k$ by claim 1 for δ.

For clause 4, suppose $t = \texttt{case}\ a\ \texttt{of}\ (i \Rightarrow e_i)$. If $\ll a^* \gg$ is an application b, then $\ll t^* \gg = \texttt{case}\ b\ \texttt{of}\ (i \Rightarrow \ll e_i^* \gg)$, which is in $\mathcal{A}_k$ by the induction hypothesis and clause 4. If $\ll a^* \gg$ is an expression d, then clearly $\ll t^* \gg = d[i \mapsto \ll e_i^* \gg]$ by Proposition 11; but this too belongs to $\mathcal{A}_k$ by the induction hypothesis and Lemma 13.

For clause 6, suppose $t = \ll \Pi_{(\Gamma,\vec{x}),Z}(e,\xi) \gg$ where $(\Gamma,\vec{x}), Z, e, \xi$ satisfy the conditions of the plugging rule. Then

$$\begin{array}{rcccl} \ll t^* \gg & = & \ll\ll \Pi(e,\xi) \gg^* \gg & = & \ll (\lambda\vec{x}. \ll \Pi(e,\xi) \gg)\, \vec{q} \gg \\ & = & \ll (\lambda\vec{x}. \Pi(e,\xi))\, \vec{q} \gg & = & \ll \Pi(e,\xi)^* \gg\ , \end{array}$$

where the third equality holds by the evaluation theorem. But it is easy to see from Definition 7 that the relevant plugging commutes with the substitution *, so that

$$\ll \Pi_{(\Gamma,\vec{x}),Z}(e,\xi)^* \gg \;=\; \ll \Pi_{\Gamma,Z}(e^*,\xi^*) \gg$$

where $\xi^*(z) = \xi(z)^*$. Furthermore, as an instance of Lemma 8, we have that

$$\ll \Pi_{\Gamma,Z}(e^*,\xi^*) \gg \;=\; \ll \Pi_{\Gamma,Z}(e^\dagger,\xi^\dagger) \gg$$

where $e^\dagger = \ll e^* \gg$ and $\xi^\dagger(z) = \ll \xi(z)^* \gg$. We have thus shown that

$$\ll t^* \gg \;=\; \ll \Pi_{\Gamma,Z}(e^\dagger,\xi^\dagger) \gg\ .$$

It remains to check that this plugging conforms to the requirements of the plugging rule, and so witnesses $\Gamma \vdash \ll t^* \gg \in \mathcal{A}$ as required. By the inner induction hypothesis we have $\Gamma \vdash e^* \in \mathcal{A}_k$; moreover, e^* is of order δ since e is a normal form and the q_i are of level $\leq \delta$, so by claim 1 for δ (as established above) we have that $\Gamma \vdash e^\dagger = \ll e^* \gg \in \mathcal{A}_k$. The same reasoning also shows that $\Gamma \vdash \xi^\dagger(z) \in \mathcal{A}_k$ for each $z \in Z$. Since also $e^\dagger$ and each $\xi^\dagger(z)$ are normal forms, the plugging $\Pi_{\Gamma,Z}(e^\dagger, \xi^\dagger)$ satisfies the conditions of the plugging rule. This concludes the proof of claim 2 for δ, and the whole of the outer induction is now complete. $\Box$

Corollary 15 *The sets $\mathsf{A}_k(\sigma)$ form a substructure $\mathsf{A}_k \subseteq \mathsf{SP}$ closed under application. Likewise, the sets $\mathsf{A}^0_k(\sigma)$ form an applicatively closed substructure $\mathsf{A}^0_k \subseteq \mathsf{SP}^0$.*

Proof Suppose $\Gamma \vdash p \in \mathsf{A}_k(\sigma \to \tau)$ and $\Gamma \vdash q \in \mathsf{A}_k(\sigma)$. Then for a suitable $\vec{x}$ we have $\Gamma \vdash \lambda\vec{x}.pq\vec{x}^{\,\eta} \in \mathcal{A}_k(\tau)$ using Definition 9(i); moreover, this term contains just a single redex so is of finite order. Thus $\Gamma \vdash p \cdot q = \ll \lambda\vec{x}.pq\vec{x}^{\,\eta} \gg \in \mathcal{A}_k$ by Lemma 14, so $p \cdot q \in \mathsf{A}_k(\Gamma, \tau)$. Moreover, if p and q are closed then so is $p \cdot q$. $\Box$

Proposition 16 *If* $\mathrm{lv}(\sigma) \leq k$*, then* $Y_\sigma \in \mathsf{A}^0_k$.

Proof Suppose $\sigma = \sigma_0, \ldots, \sigma_{r-1} \to \mathtt{N}$ has level $\leq k$. Let g be a variable of type $\sigma \to \sigma$, and for each i, let x_i, x'_i be variables of type σ_i. We also let z be a plugging variable of type σ. Now let

$$\begin{aligned} p &= \lambda x'_0 \cdots x'_{r-1}.\,\mathtt{case}\ g z^\eta {x'_0}^\eta \cdots {x'_{r-1}}^\eta\ \mathtt{of}\ (i \Rightarrow i)\ , \\ e &= \mathtt{case}\ g z^\eta {x_0}^\eta \cdots {x_{r-1}}^\eta\ \mathtt{of}\ (i \Rightarrow i)\ , \end{aligned}$$

and let $\Gamma = \{g, x_0, \ldots, x_{r-1}\}$, $Z = \{z\}$, and $\xi(z) = p$. Then the plugging $\Pi_{\Gamma,Z}(e, \xi)$ conforms to the requirements of the plugging rule since e and p are well-founded and in normal form, and clearly $Y_\sigma = \lambda g\vec{x}. \ll \Pi_{\Gamma,Z}(e,\xi) \gg$. Thus $Y_\sigma \in \mathsf{A}^0_k$ via clauses 6 and 2 of Definition 9. (Note that we have used two sets of variables $\vec{x}$ and $\vec{x}\,'$ here simply to conform to the stipulation on bound variables in Definition 7.) $\Box$

We are now ready to consider the interpretation of PCF^Ω_k in SP^0 as defined by the clauses in Section 2:

Theorem 17 *If* $\Gamma \vdash M : \sigma$ *is any term of* PCF^Ω_k*, then* $[\![M]\!]_\Gamma \in \mathsf{A}_k(\Gamma, \sigma)$.

Proof By induction on M, using the definition of $[\![-]\!]$ from Subsection 2.3. For the terms x^σ, $\widehat{n}$, *suc*, *pre*, *ifzero* and C_f, it suffices to note that the corresponding NSPs are well-founded, and the case of Y_σ where $\mathrm{lv}(\sigma) \leq k$ is handled by Proposition 16. The induction case for λ-abstraction is trivial using clause 2 of Definition 9, and the case for application is given by Corollary 15. $\Box$

4 Acceptable procedures are not spinal

In this section, we will prove that the element $Y_{k+1} \in \mathsf{SP}^0$ is not PCF^Ω_k-definable by showing that $Y_{k+1} \notin \mathsf{A}_k$. Recall that the procedure Y_{k+1} may be defined as $\lambda g^{(k+1)\to(k+1)} x^k.\, C[g,x]$, where $C[g,x]$ is given co-recursively (up to α-equivalence) by:

$$C[g,x] \;=\; \texttt{case}\; g\,(\lambda x'^{k}.C[g,x'])\,x^{\eta}\;\texttt{of}\;(i \Rightarrow i)\;.$$

Since the only way to generate λ-abstractions in A_k is via rule 2 of Definition 9, to show that $Y_{k+1} \notin \mathsf{A}_k$ it will suffice to show that $g, x \vdash C[g,x] \notin \mathsf{A}_k$. We will henceforth treat $g : (k+1) \to (k+1)$ as a global variable whose identity will remain fixed throughout the subsequent discussion.

We will actually prove a more general claim than this, to the effect that A_k contains no *spinal terms*. Roughly speaking, these are terms with an 'infinite spine' of nested applications of g broadly similar to the one in $C[g,x]$. The notion of spinal term will be defined formally in Definition 21, once we have introduced some necessary technical prerequisites. We will also explain how the same theory, with one slight modification, provides what we will need in order to show in Section 5 that $Y_{0\to(k+1)}$ is not $\mathrm{PCF}^{\Omega}_{k}$-definable in SF.

We begin by identifying some properties that will be shared by all the terms-in-environment that we shall ever need to consider. The following ad hoc notions will be useful:

Definition 18 *(i) An environment* Γ *is* regular *if* Γ *contains* g *but no other variables of type level* $> k$.

(ii) A term-in-environment $\Gamma \vdash t$ *is* regular *if* Γ *is regular and* t *is not a procedure of type level* $> k+1$.

Proposition 19 *If* $\Gamma \vdash t$ *is regular, then all variables bound by a* λ*-abstraction within* t *are of level* $\leq k$.

Proof Suppose not, and suppose $\lambda\vec{x}.e$ is some outermost subterm of t with $\mathrm{lv}(\vec{x}) > k$. Then $\lambda\vec{x}.e$ cannot be the whole of t, since t would then be a procedure of level $> k+1$. Since t is a normal form, the subterm $\lambda\vec{x}.e$ (of level $> k+1$) must therefore occur as an argument to some variable w of level $> k+2$. But this is impossible, since Γ contains no such variables, nor can such a w be bound within t, since the relevant subterm $\lambda\vec{w}.d$ would then properly contain $\lambda\vec{x}.e$, contradicting the choice of the latter. $\square$

The above proposition suggests a generalization of Definition 18(ii) to meta-terms, which will sometimes be useful in the sequel:

Definition 20 *A meta-term-in-environment* $\Gamma \vdash T$ *is* regular *if* Γ *is regular all abstractions* $\lambda\vec{x}$ *within* T *are of level* $\leq k$ *(whence* T *is not a meta-procedure of level* $> k+1$*).*

Let us now consider the generation of k*-acceptable* (normal-form) terms-in-environment $\Gamma \vdash t \in \mathsf{A}_k$, where $\Gamma \vdash t$ is regular. As noted after Definition 9, any such term either arises from rule 1 of this definition, or is constructed from previously generated terms-in-environment by rule 2, 3, 4 or 6. Moreover, an inspection of these rules readily confirms that these previously generated terms

$\Gamma' \vdash t' \in \mathsf{A}_k$ must themselves be regular and in normal form. Thus, by induction, any derivation of $\Gamma \vdash t \in \mathsf{A}_k$ will consist entirely of regular normal-form terms-in-environment, and hence all λ-abstractions within these terms will be of level $\leq k$. In particular, since $g, x \vdash C[g,x]$ is regular, the above applies to any putative derivation of $g, x \vdash C[g,x] \in \mathsf{A}_k$.

We shall adopt the convention that any environment denoted by Γ will contain g as its only variable of level $> k$; recall that Roman letters such as V, X, Z always denote lists of variables of level $\leq k$ (which may also contribute to the environments we consider).

We are now ready to introduce the concept of a *spinal term*, generalizing the structure exhibited by the terms $C[g,x]$. In particular, we wish to allow the crucial applications of g to occur at positions other than the head of the enclosing abstractions $(\lambda x. \cdots)$, and also to allow the second argument in such applications to be something other than a pure η-expanded variable (the precise details are rather delicate). By the *local variable environment* associated with a meta-term context $\Gamma \vdash C[-]$, we shall mean the set X of variables x bound within $C[-]$ whose scope includes the hole, so that the environment in force at the hole is Γ, X.

Definition 21 (Spinal terms) *(i) If x is a variable of type k and V a set of variables, a substitution $^\circ = [\vec{w} \mapsto \vec{r}]$ is called x,V-*closed *if the r_i contain no free variables, except that if $w_i \in V$ and $\mathrm{lv}(w_i) < k$ then r_i may contain x free.*

*(ii) We coinductively declare a regular expression $\Gamma \vdash e$ to be (k)-*head-spinal *with respect to x, V iff e has the form*

$$\texttt{case}\ g(\lambda x'.E[e'])o\ \texttt{of}\ (\cdots)$$

where $E[-]$ is any expression context, and

1. *for some x,V-closed specialization $^\circ$ covering the free variables of o other than x, we have $o^\circ \succeq x^\eta$,*

2. *e' is head-spinal with respect to x', V', where V' is the local variable environment for $E[-]$.*

(That is, we take 'e is head-spinal w.r.t. x, V' to be the largest relation that satisfies the above equivalence.) In the above setting, we may also refer to the application $g(\lambda x'.E[e'])o$ as a head-spinal term.

*(iii) We say a regular term $\Gamma \vdash t$ is $(k$-)*spinal *if it contains a head-spinal subexpression w.r.t. some x, V.*

Certain aspects of condition 1 in this definition deserve comment. For the purpose of showing the non-definability of Y_{k+1} as an element of SP^0, one could make do with the following simpler condition: for some *closed* specialization $^\circ$ of the free variables of o other than x, we have $\ll o^\circ \gg = x^\eta$. The looser condition adopted above is designed with the proof of non-definability in SF in mind: we will see in Section 5 that every (simple) procedure representing a certain functional $\Phi \in \mathsf{SF}$ is spinal in the above sense. With this in mind, one might naturally expect to see the condition $o^\circ \approx x^\eta$ here, but it turns that the argument goes through most smoothly with $\succeq$ in place of $\approx$. (In Appendix A we will see that $o^\circ \succeq x^\eta$ is actually equivalent to $o^\circ \approx x^\eta$, although this is not needed for our main proof.) The reason for requiring $^\circ$ to be x,V-closed is somewhat technical, but it will emerge from the proof of Lemma 23 that this is indeed the appropriate condition to work with.[3]

We emphasize that the notion of a head-spinal term is ad hoc, and the class of such terms is not a particularly natural one. It is easy to devise procedures q (lying outside A_k) that do not possess the specific form of a spinal term, but which nonetheless contain what it takes to generate such a term, so that for example $p \cdot q$ is spinal for some innocuous (well-founded) procedure p. Thus, the non-spinal terms are by no means closed under application; in particular, the class of such terms could not itself play the role of A_k in our proof.

At this point, we digress briefly to explain the small modification of this machinery that we will need for the results of Section 5. For reasons to be explained there, these results call for a setup in which the global variable g has the slightly different type $0 \rightarrow (k+1) \rightarrow (k+1)$. We may now vary the above definition by coinductively declaring e to be head-spinal w.r.t. x, V iff e has the form

$$\texttt{case}\ gb(\lambda x'.E[e'])o\ \texttt{of}\ (\cdots)$$

where b is a procedure term of type 0 and conditions 1 and 2 above are also satisfied. Subject to this adjustment, all the results and proofs of the present section go through in this modified setting, with the extra argument b playing no active role. For the remainder of this section, we shall work with a global variable g of the simpler type $(k+1) \rightarrow (k+1)$, on the understanding that the extra arguments b can be inserted where needed to make formal sense of the material in the modified setting. We do not expect that any confusion will arise from this.

Clearly $g, x \vdash C[g, x]$ is spinal. As a generalization of our claim that $g, x \vdash C[g, x] \notin \mathsf{A}_k$, the main result in this section will be that no regular term $\Gamma \vdash t \in \mathsf{A}_k$ can be spinal (Theorem 24).

Our proof strategy will be broadly as follows. Suppose for contradiction that $\Gamma \vdash t$ is some spinal term of minimal ordinal rank in the generation of A_k. We argue by cases on the final rule from Definition 9 applied in the generation of $\Gamma \vdash t \in \mathsf{A}_k$, the only challenging case being the one for rule 6. Here we require some technical machinery to show that a spinal structure must already be present in one of the individual term fragments from which t is constructed by plugging: there is no way to 'assemble' a spinal structure from material in two or more non-spinal fragments. Since these term fragments have a lower ordinal rank than t itself, we obtain a contradiction.

The bulk of the proof will consist of some lemmas developing the machinery

[3]It can be shown using Theorem 6 that if $o^\circ \succeq x^\eta$ where $^\circ$ is x,V-closed, then at least one x in $\ll o^\circ \gg$ must originate from o rather than from $^\circ$. We have not actually settled the question of whether there are procedures o such that $o^\circ \succeq x^\eta$ for some x,V-closed $^\circ$ but not for any closed $^\circ$; fortunately this is not necessary for the purpose of our proof.

necessary for tackling rule 6. We start with some technical but essentially straightforward facts concerning evaluation and the tracking of subterms and variable substitutions. This will serve as a useful tool several times in what follows, and is formulated with these intended applications in mind.

Lemma 22 *Suppose that*

$$\Gamma \vdash \ll K[d] \gg \;=\; K'[c]$$

where $\Gamma, \vec{v} \vdash d = \texttt{case}\ gpq\ \texttt{of}\ (\cdots)$ *and* $\Gamma, \vec{v}\,' \vdash c = \texttt{case}\ gp'q'\ \texttt{of}\ (\cdots)$*; here* $\vec{v}, \vec{v}\,'$ *are the local environments for* $K[-], K'[-]$ *respectively. Suppose also that:*

1. $\Gamma \vdash K[d]$ *is regular.*
2. *In the evaluation above, the head* g *of* c *originates from that of* d*.*

Then:

(i) There is a substitution $^\dagger = [\vec{v} \mapsto \vec{s}]$ *of level* $\leq k$*, with* $\Gamma, \vec{v}\,' \vdash \vec{s}$ *regular, such that* $\Gamma, \vec{v}\,' \vdash gp'q' = \ll (gpq)^\dagger \gg$*, whence* $\ll d^\dagger \gg$ *has the form* $\texttt{case}\ gp'q'\ \texttt{of}\ (\cdots)$*.*

(ii) If furthermore we have

3. c *is head-spinal w.r.t.* x, V*,*

then also $\ll d^\dagger \gg$ *is head-spinal w.r.t.* x, V*.*

(iii) If $K[-]$ *contains no redexes* $P\vec{Q}$ *where* P *is of type level* $k+1$*, then* † *is* trivial for level k variables*: that is, if* $v_i \in \vec{v}$ *is of level* k*, then also* $v_i \in \vec{v}\,'$ *and* $v_i^\dagger = s_i = v_i^\eta$*. More generally,* † *can be non-trivial for a level* k *variable* v_i *only if* v_i *occurs as an eigenvariable of some* β*-redex* $P\vec{Q}$ *in* $K[-]$*, where the hole* $-$ *lies within* P*.*

Part (iii) of the above statement should of course be understood relative to suitably chosen representatives for the α-equivalence classes of $K[d]$ and $K'[c]$. To adhere to our convention of working with meta-terms only up to α-equivalence, one should strictly speaking frame the statement in terms of a partial injection from variables of $\vec{v}$ to those of $\vec{v}\,'$.

Proof (i) We formulate a suitable property of terms that is preserved under all individual reduction steps. Let $K[-], p, q$ and $\vec{v}$ be fixed as above, and suppose that

$$K^0[\texttt{case}\ gP^0Q^0\ \texttt{of}\ (\cdots)] \rightsquigarrow K^1[\texttt{case}\ gP^1Q^1\ \texttt{of}\ (\cdots)]$$

via a single reduction step, where the g on the right originates from the one on the left, and moreover K^0, P^0, Q^0 enjoy the following properties (we write $\vec{v}^{\,0}$ for the local environment for $K^0[-]$):

1. $\Gamma \vdash K^0[\texttt{case}\ gP^0Q^0\ \texttt{of}\ (\cdots)]$ is regular.
2. There exists a substitution $^{\dagger 0} = [\vec{v} \mapsto \vec{s}^{\,0}]$ (with $\Gamma, \vec{v}^{\,0} \vdash \vec{s}^{\,0}$ regular) such that $\ll gP^0Q^0 \gg = \ll (gpq)^{\dagger 0} \gg$.

We claim that K^1, P^1, Q^1 enjoy these same properties with respect to the list $\vec{v}^{\,1}$ of local variables in scope at $K^1[-]$. For property 1, it suffices to note that $K^1[\texttt{case}\ gP^1Q^1\ \texttt{of}\ (\cdots)]$ cannot contain any bound variables of level $> k$ because $K^0[\texttt{case}\ gP^0Q^0\ \texttt{of}\ (\cdots)]$ does not (see Proposition 19). For property 2, we define the required substitution ${}^{\dagger 1} = [\vec{v} \mapsto \vec{s}^{\,1}]$ by cases on the nature of the reduction step in question:

- If the subexpression $\texttt{case}\ gP^0Q^0\ \texttt{of}\ (\cdots)$ is unaffected by the reduction (so that $P^0 = P^1$ and $Q^0 = Q^1$), or if the reduction is internal to P^0, Q^0 or to the rightward portion $(\cdots)$, or if the reduction has the form
$$\begin{array}{c}\texttt{case}\ (\texttt{case}\ gP^0Q^0\ \texttt{of}\ (i \Rightarrow E_i^0))\ \texttt{of}\ (j \Rightarrow F_j) \quad \rightsquigarrow \\ \texttt{case}\ gP^0Q^0\ \texttt{of}\ (i \Rightarrow \texttt{case}\ E_i^0\ \texttt{of}\ (j \Rightarrow F_j))\end{array}$$
then the conclusion is immediate, noting that $\vec{v}^{\,1} = \vec{v}^{\,0}$ and taking ${}^{\dagger 1} = {}^{\dagger 0}$.

- If the reduction is for a β-redex $(\lambda \vec{x}.E)\vec{R}$ where the indicated subexpression $\texttt{case}\ gP^0Q^0\ \texttt{of}\ (\cdots)$ lies within some R_i, we may again take ${}^{\dagger 1} = {}^{\dagger 0}$. In this case $\vec{v}^{\,1}$ may extend $\vec{v}^{\,0}$, but we will still have $\Gamma, \vec{v}^{\,1} \vdash \vec{s}^{\,1}$.

- If the reduction is for a β-redex $(\lambda \vec{x}.E)\vec{R}$ where $\texttt{case}\ gP^0Q^0\ \texttt{of}\ (\cdots)$ lies within E, then $P^1 = P^0[\vec{x} \mapsto \vec{R}]$ and similarly for Q^1. In this case, the local environment $\vec{v}^{\,1}$ for $K^1[-]$ will be $\vec{v}^{\,0} - \vec{x}$, so that the conclusion follows if we take ${}^{\dagger 1}$ to be the composite substitution ${}^{\dagger 0}; [\vec{x} \mapsto \vec{R}]$, recast as a single substitution $[\vec{v} \mapsto \vec{s}^{\,1}]$. (Any variables in $\vec{x} - \vec{v}$ may be safely omitted from the domain of ${}^{\dagger 1}$ as they do not occur free in gpq.) Here $\Gamma, \vec{v}^{\,1} \vdash \vec{s}^{\,1}$ will be regular since the $\vec{R}$ are regular.

Now in the situation of the lemma we will have some finite reduction sequence
$$K[\texttt{case}\ gpq\ \texttt{of}\ (\cdots)] \ \rightsquigarrow^* \ K''[\texttt{case}\ gP'Q'\ \texttt{of}\ (\cdots)]\ ,$$
where $\ll P' \gg = p'$, $\ll Q' \gg = q'$, $\ll K''[-] \gg = K'[-]$, and there is a finite normal-form context $t[-] \sqsubseteq K''[-]$ containing the hole in $K''[-]$ such that $t[-] \sqsubseteq K'[-]$. Since K, p, q themselves trivially satisfy the above invariants (taking ${}^{\dagger} = [\vec{y} \mapsto \vec{y}^{\,\eta}]$), we may infer that K'', P', Q' also satisfy these invariants with respect to some ${}^{\dagger} = [\vec{v} \mapsto \vec{s}]$ with $\Gamma, \vec{v}^{\,\prime} \mapsto \vec{s}$ regular. (The environment $\Gamma, \vec{v}^{\,\prime}$ is correct here, as $K'[-], K''[-]$ have the same local environment.) We now have $gp'q' = \ll gP'Q' \gg = \ll (gpq)^{\dagger} \gg$; note too that the $\vec{v}$ are of level $\leq k$ because $K[d]$ is regular. It also follows immediately that $\ll d^{\dagger} \gg$ has the stated form.

(ii) If c is head-spinal w.r.t. x, V, then we see from Definition 21 that $gp'q'$ and hence $d = \texttt{case}\ gp'q'\ \texttt{of}\ (\cdots)$ are head-spinal w.r.t. x, V.

(iii) From the proof of (i), we see that in the reduction of $K[\texttt{case}\ gpq\ \texttt{of}\ (\cdots)]$ to $K''[\texttt{case}\ gP'Q'\ \texttt{of}\ (\cdots)]$, any $v_i \in \vec{v}$ can be tracked through the local environments for the intermediate contexts $K^0[-], K^1[-], \ldots$ until (if ever) it is an eigenvariable for a β-reduction. For those v_i that never serve as an eigenvariable, it is clear from the construction that $v_i \in \vec{v}^{\,\prime}$ and $v_i^{\dagger} = s_i = v_i^{\eta}$. We wish to show that all $v_i \in \vec{v}$ of level k are in this category.

As in the proof of Proposition 19, we see that the meta-procedure P whose leading λ binds v_i cannot occur as an argument to another λ-abstraction, as this would require a bound variable of level $\geq k+1$. It must therefore occur as a level $k+1$ argument to g, so that we have a subterm $g(\lambda v_i.E[-])\cdots$. But this form of subterms is stable under reductions, since g is a global variable; it follows easily that this subterm has a residue $g(\lambda v_i.E'[-])\cdots$ in each of the intermediate reducts, and thus that v_i is never the eigenvariable of a β-reduction. $\square$

Thus, in the setting of the above lemma, the subterm d can be specialized and evaluated to yield a head-spinal term via the substitution $[\vec{v} \mapsto \vec{s}]$. However, we wish to show more, namely that in this setting, d itself is already a spinal term, so that the $\vec{s}$ make no essential contribution to the spinal structure. This is shown by the next lemma, whose proof forms the most complex and demanding part of our entire argument. The main challenge will be to show that all the head-spinal occurrences of g in $\ll d[\vec{v} \mapsto \vec{s}] \gg$ originate from d rather than from $\vec{s}$. The reader is advised that great care is needed regarding which variables can appear free where, and for this reason we make a habit of explicitly recording the variable environment for almost every term or meta-term we mention.

Lemma 23 *Suppose we have regular terms*

$$\Gamma, \vec{v} \vdash d = \texttt{case}\; gpq\; \texttt{of}\; (\cdots)\,, \qquad \Gamma, \vec{v}\,' \vdash \vec{s}\,, \qquad \mathrm{lv}(\vec{v}), \mathrm{lv}(\vec{v}\,') \leq k\,,$$

where $\Gamma, \vec{v}\,' \vdash t = \ll d[\vec{v} \mapsto \vec{s}] \gg$ *is head-spinal with respect to some* x, V. *Then* d *itself is spinal.*

Proof By renaming variables if necessary, we may assume for clarity that $\Gamma, \vec{v}$ and $\vec{v}\,'$ are all disjoint, all bound variables within d or $\vec{s}$ are distinct from these and from each other, and that the same variable is never bound in two places within d or $\vec{s}$.

We begin with some informal intuition. The head-spinal term t will be of the form

$$\Gamma, \vec{v}\,' \vdash t = \texttt{case}\; g\,(\lambda x'.\, E[\texttt{case}\; gF'o'\; \texttt{of}\; (\cdots)])\, o\; \texttt{of}\; (\cdots)\,,$$

where $o'^{\circ} \succeq x'^{\eta}$ for some $^{\circ}$. Here the $\lambda x'$ clearly originates from d rather than from $\vec{s}$. Suppose, however, that the second spinal occurrence of g in t originated from some s_i rather than from d. In order to form the application of this g to o', the whole extension of x'^{η} would in effect need to be 'passed in' to s_i when d and $\vec{s}$ are combined. But this is impossible, since the arguments to s_i are of level $< k$, so by Theorem 6 we cannot funnel the whole of x'^{η} through them: that is, the interface between d and $\vec{s}$ is too 'narrow' for the necessary communication to occur. (The situation is made slightly more complex by the fact that some components of $^{\circ}$ may also involve x', but the same idea applies.) It follows that the second spinal g in t originates from d. By iterating this argument, we can deduce that all the spinal occurrences of g, and indeed the entire spinal structure, comes from d.

We now proceed to the formal proof. Let $^\dagger = [\vec{v} \mapsto \vec{s}]$, and consider the subterm $p = \lambda x'.e$ within d, where $\Gamma, \vec{v}, x' \vdash e$. Since $\ll d^\dagger \gg$ is head-spinal, $\ll e^\dagger \gg$ will be some spinal term $\Gamma, x', \vec{v}\,' \vdash E[c]$, where $\Gamma, x', \vec{v}\,', \vec{y}\,' \vdash c = \texttt{case}\ gF'o'\ \texttt{of}\ (\cdots)$ is head-spinal with respect to x'. (Here $\vec{y}\,'$ denotes the local environment for $E[-]$.) We will first show that the head g of c comes from e rather than from †; we will later show that the same argument can be repeated for lower spinal occurrences of g.

Claim 1: In the evaluation $\ll e^\dagger \gg = E[c]$, the head g of c originates from e.

Proof of Claim 1: Suppose for contradiction that the head g of c originates from some substituted occurrence of an s_i within $e^\dagger$, say as indicated by $e^\dagger = D[s_i]$ and $s_i = L[d']$, where $\Gamma, x', \vec{v}\,' \vdash D[-]$, $\Gamma, \vec{v}\,' \vdash s_i$, and $\Gamma, \vec{v}\,', \vec{z} \vdash d' = \texttt{case}\ gp'q'\ \texttt{of}\ (\cdots)$. (Here $\vec{z}$ is the local variable environment for $L[-]$.) Then

$$\Gamma, x', \vec{v}\,' \ \vdash \ \ll e^\dagger \gg \ = \ \ll D[L[d']] \gg \ = \ E[c] \ ,$$

where the head g in d' is the origin of the head g in c. We will use this to show that a head-spinal term may be obtained from d' via a substitution of level $< k$; this will provide the bottleneck through which x'^{η} is unable to pass.

We first note that the above situation satisfies the conditions of Lemma 22, where we take the Γ, K, d, K', c of the lemma to be respectively $(\Gamma, x', \vec{v}\,')$, $D[L[-]], d', E, c$. Condition 1 of the lemma holds because $\Gamma, \vec{v}, x' \vdash e$ and $\Gamma, \vec{v}\,' \vdash \vec{s}$ are clearly regular, so that e and hence $e^\dagger$ involve no bound variables of level $> k$ (see Proposition 19); conditions 2 and 3 are immediate in the present setup.

We conclude by Lemma 22 that there is a substitution $[\vec{y} \mapsto \vec{u}]$ (called $[\vec{v} \mapsto \vec{s}]$ in the statement of the lemma), covering exactly the local variables of $D[L[-]]$, and with $\Gamma, x', \vec{v}\,', \vec{y}\,' \vdash \vec{u}$ (recalling that $\vec{y}\,'$ are the local variables for $E[-]$), such that $\ll d'[\vec{y} \mapsto \vec{u}\,] \gg$ is head-spinal and indeed of the form $\texttt{case}\ gF'o'\ \texttt{of}\ (\cdots)$. Furthermore, the only β-redexes in $e^\dagger$ are those arising from the substitution †, with some s_j of level k as operator. There are therefore no β-redexes with an eigenvariable of level k, so by Lemma 22(iii), the substitution $[\vec{y} \mapsto \vec{u}]$ is trivial for variables of level k.

In fact, as shown by the proof of Lemma 22(iii), we may split $\vec{y}$ into the variables $\vec{y}^{\,-}$ that are specialized by a β-reduction in the course of the evaluation of $D[L[d']]$, and the variables $\vec{y}^{\,+}$ that are not. Here the former are all of level $< k$, whilst the latter may be identified with certain variables in $\vec{y}\,'$, and for these the substitution is trivial. We correspondingly split $\vec{u}$ into $\vec{u}^{\,-}$ and $\vec{u}^{\,+}$.

Summarizing, we have that

$$\Gamma, x', \vec{v}\,', \vec{y}\,' \ \vdash \ll d'[\vec{y} \mapsto \vec{u}] \gg \ = \ \texttt{case}\ gF'o'\ \texttt{of}\ (\cdots) \ ,$$

where $\Gamma, \vec{v}\,', \vec{z} \vdash d' = \texttt{case}\ gp'q'\ \texttt{of}\ (\cdots)$, $\Gamma, x', \vec{v}\,', \vec{y} \vdash \vec{u}$, $[\vec{y} \mapsto \vec{u}]$ is trivial for level k variables, and the right-hand side is head-spinal. From this we may read off that

$$\Gamma, x', \vec{v}\,', \vec{y}\,' \ \vdash \ll q'[\vec{y} \mapsto \vec{u}] \gg \ = \ o' \ .$$

Moreover, x' does not occur free in q'. This is because q' is a part of d' which is here presumed to be a subterm of s_i, whereas x' is bound within d: in particular, we have $\Gamma, \vec{v}\,', \vec{z} \vdash q'$, where $x' \notin \Gamma, \vec{v}\,', \vec{z}$.[4] Thus, each free occurrence of x' in o' above must originate from some u_j, which must have some type ρ_j of level $< k$, since if u_j has level k then $u_j = y_j^\eta$, where $y_j \neq x'$. Indeed, since the variables of $\vec{y}^{\,+}$ are shared by $\vec{y}$ and $\vec{y}\,'$, and $[\vec{y}^{\,+} \mapsto \vec{u}^{\,+}]$ is trivial, we may rewrite the above equation as

$$\Gamma, x', \vec{v}\,', \vec{y}\,' \;\vdash \ll q'[\vec{y}^{\,-} \mapsto \vec{u}^{\,-}] \gg \;\; = \;\; o' \,,$$

where $[\vec{y}^{\,-} \mapsto \vec{u}^{\,-}]$ is of level $< k$. Recall that $\Gamma, \vec{v}\,', \vec{y} \vdash q'$ and $\Gamma, \vec{v}\,', \vec{y}\,', x' \vdash \vec{u}^{\,-}$.

We may exploit this to manifest $\overline{k}$ as a pseudo-retract of $\Pi_j \rho_j$, contradicting Theorem 6. Specifically, take $^\circ = [\vec{w} \mapsto \vec{r}]$ an x',W-closed substitution (where $W = \vec{y}\,'$) such that $o'^\circ \succeq x'^\eta$ as in Definition 21; we may assume that $\vec{w}$ covers the variables of $\Gamma, \vec{v}\,', \vec{y}\,'$. Regrouping our variables, and noting that $\vec{y}^{\,+} \subseteq \vec{y}\,'$, we may now write $\vec{w}, x' \vdash \vec{u}^{\,-}$ and $\vec{w}, \vec{y}^{\,-} \vdash q'$, where neither $\vec{w}$ nor $\vec{y}^{\,-}$ contains x'. From the former of these, we obtain procedures $x' \vdash u_j^\bullet = \ll u_j^\circ \gg$ for $y_j \in \vec{y}^{\,-}$.

Next, let us split $^\circ$ into two parts: a substitution $[\vec{w}\,' \mapsto \vec{r}\,']$ covering the variables of level $\geq k$ (where $\vdash \vec{r}\,'$), and a substitution $[\vec{w}\,'' \mapsto \vec{r}\,'']$ covering those of level $< k$ (where $x' \vdash \vec{r}\,''$). Set $q'^\bullet = \ll q'[\vec{w}\,' \mapsto \vec{r}\,'] \gg$, so that $\vec{w}\,'', \vec{y}^{\,-} \vdash q'^\bullet$ and $\vec{y}^{\,-}, x' \vdash \ll q'^\bullet[\vec{w}\,'' \mapsto \vec{r}\,''] \gg = \ll q'^\circ \gg$. We now have

$$x' \;\vdash\; \vec{r}\,'' \,, \qquad x' \;\vdash\; \vec{u}^{\,\bullet -}, \qquad \vec{w}\,'', \vec{y}^{\,-} \vdash q'^\bullet$$

with $x' : k$ and $\vec{w}\,'', \vec{y}\,'$ of level $< k$. Moreover, $\vec{w}\,'$ and $\vec{y}^{\,-}$ are disjoint since the latter is of level $< k$, and $\vec{r}\,', \vec{u}^{\,\bullet -}$ contain at most x free, so we have

$$\begin{array}{rcl} x' & \vdash & \ll q'^\bullet[\vec{y}^{\,-} \mapsto \vec{u}^{\,\bullet -}][\vec{w}\,'' \mapsto \vec{r}\,''] \gg \\ & = & \ll q'[\vec{y}^{\,-} \mapsto \vec{u}^{\,\bullet -}][\vec{w}\,' \mapsto \vec{r}\,', \vec{w}\,'' \mapsto \vec{r}\,''] \gg \\ & = & \ll (q'[\vec{y}^{\,-} \mapsto \vec{u}^{\,-}])^\circ \gg \\ & = & \ll o'^\circ \gg \;\; \succeq \;\; x'^\eta \,. \end{array}$$

Since $[\vec{y}^{\,-} \mapsto \vec{u}^{\,\bullet -}][\vec{w}\,'' \mapsto \vec{r}\,'']$ is equivalent to a single substitution of level $< k$ (whether or not $\vec{y}^{\,-}$ and $\vec{w}\,''$ are disjoint), this contradicts Theorem 6, and the proof of Claim 1 is complete.

In order to continue this analysis to greater depth, let us now suppose that the originating occurrence of the head g in c is as indicated by $\Gamma, x', \vec{v} \vdash e = C[d']$, where $\Gamma, x', \vec{v}, \vec{v}\,'' \vdash d' = \texttt{case}\; g p' q'\; \texttt{of}\; (\cdots)$, with $\vec{v}\,''$ the local environment for $C[-]$. (The use of d', p', q' here is distinct from their use within the proof of Claim 1.) Then we have

$$\Gamma, x', \vec{v}\,' \vdash \;\; \ll e^\dagger \gg \;\; = \;\; \ll (\lambda \vec{v}.\, C[d'])\vec{s} \gg \;\; = \;\; E[c] \,,$$

[4]Note that $\vec{y}$ (the local environment for $D[L[-]]$) subsumes $\vec{z}$ (the local environment for $L[-]$). We will sometimes write $\Gamma, \vec{v}\,', \vec{y} \vdash q'$ because all the variables of $\vec{y}$ are in scope at the position of q' within $e^\dagger$; however, the variables of $\vec{y} - \vec{z}$ do not actually occur in q'.

where c is head-spinal w.r.t. x', W, and the head g of d' is the origin of the head g of c.

We claim that once again we are in the situation of Lemma 22, taking Γ, K, d, K', c of the lemma to be respectively $(\Gamma, x', \vec{v}\,')$, $(\lambda\vec{v}.\,C[-])\vec{s}$, d', E, c. Conditions 2 and 3 of the lemma are immediate in the present setup; for condition 1, we again note that $\Gamma, x', \vec{v}\,' \vdash C[d'] = e$ and $\Gamma, \vec{v}\,' \vdash \vec{s}$ are regular, so by Proposition 19 contain no bound variables of level $> k$; hence the same is true for $(\lambda\vec{v}.\,C[d'])\vec{s}$. Applying Lemma 22, we obtain a substitution ${}^{\dagger'} = [\vec{v}^{\,+} \mapsto \vec{s}^{\,+}]$ of level $\leq k$ (with $\vec{v}^{\,+} = \vec{v}, \vec{v}\,''$) where $\Gamma, x', \vec{v}\,', \vec{v}^{\,+} \vdash d'$ and $\Gamma, x', \vec{v}\,', \vec{y}\,' \vdash \vec{s}^{\,+}$, such that

$$\Gamma, x', \vec{v}\,', \vec{y}\,' \;\vdash \ll d'[\vec{v}^{\,+} \mapsto \vec{s}^{\,+}] \gg$$

is head-spinal w.r.t. x', W, and indeed of the form `case` $gF'o'$ `of` $(\cdots)$. (We may in fact write $\Gamma, x', \vec{v}^{\,+} \vdash d'$, as the variables of $\vec{v}\,'$ do not appear in d'.) We may also read off that $\ll (p')^{\dagger'} \gg = F'$ and $\ll (q')^{\dagger'} \gg = o'$. Note too that by Lemma 22(ii), the substitution ${}^{\dagger'}$ is trivial for any level k variables in $\vec{v}\,''$, as $C[-]$ is in normal form.

We are now back precisely where we started, in the sense that $d', \vec{v}^{\,+}, \vec{s}^{\,+}$ satisfy the hypotheses of the lemma, with (Γ, x') now playing the role of Γ and $(\vec{v}\,', \vec{y}\,')$ that of $\vec{v}\,'$. We may therefore iterate the above argument to obtain an infinite descending chain of subterms

$$\begin{array}{rrll}
\Gamma, \vec{v} \vdash & d = & \texttt{case}\ gpq\ \texttt{of}\ (\cdots)\ , & p = \lambda x'.\,C[d']\ , \\
\Gamma, \vec{v}, x', \vec{v}\,'' \vdash & d' = & \texttt{case}\ gp'q'\ \texttt{of}\ (\cdots)\ , & p' = \lambda x''.\,C'[d'']\ , \\
\Gamma, \vec{v}, x', \vec{v}\,'', x'', \vec{v}\,'''' \vdash & d'' = & \texttt{case}\ gp''q''\ \texttt{of}\ (\cdots)\ , & p'' = \lambda x'''.\,C''[d''']\ , \\
 & \cdots & & \cdots
\end{array}$$

along with associated substitutions ${}^{\dagger}, {}^{\dagger'}, {}^{\dagger''}, \ldots$ applicable to $p, p', p'', \ldots$ respectively, such that the terms $\ll p^{\dagger} \gg$, $\ll (p')^{\dagger'} \gg$, $\ll (p'')^{\dagger''} \gg, \ldots$ coincide with the successive procedure subterms $F, F', \ldots$ from the spine of the original t, and likewise $\ll q^{\dagger} \gg$, $\ll (q')^{\dagger'} \gg$, $\ll (q'')^{\dagger''} \gg, \ldots$ coincide with $o, o', \ldots$.

We cannot quite conclude that d is head-spinal, because the critical x in $q^{\dagger}$ might originate not from q but from a level k term in $\vec{s}$ (for example). However, we can show that this problem does not arise for $q', q'', \ldots$, essentially because $x', x'', \ldots$ are locally bound within p. We will in fact show that d' is head-spinal w.r.t. x', U, where U is the local environment for $C[-]$ (so that $U = \vec{v}\,''$); this will imply that d is spinal. In the light of Definition 21, it will be sufficient to show that $(q')^{\circ'} \succeq x'^{\eta}$ for some x',U-closed specialization ${}^{\circ'}$ covering the free variables of q' except x' (namely those of $\Gamma, \vec{v}^{\,+}$ where $\vec{v}^{\,+} = \vec{v}, \vec{v}\,''$); the same argument will then obviously apply also to $q'', q''', \ldots$.

Recall that $\Gamma, \vec{v}, \vec{v}\,'', x \vdash q'$ and $\Gamma, \vec{v}\,', \vec{y}\,', x' \vdash o'$. As before, let ${}^{\circ} = [\vec{w} \mapsto \vec{r}]$ be x',W-closed such that $o'^{\circ} \succeq x'^{\eta}$, where W is the local environment for $E[-]$ (so that $W = \vec{y}\,'$), and $\vec{w}$ covers $\Gamma, \vec{v}\,', \vec{y}\,'$. Now define

$${}^{\circ'} \;=\; [\vec{v} \mapsto \vec{s}^{\,\circ},\ \vec{v}\,'' \mapsto (\vec{s}\,'')^{\circ},\ \vec{w}^{\,\Gamma} \mapsto \vec{r}^{\,\Gamma}]\ ,$$

where $[\vec{v} \mapsto \vec{s}^{\,\circ},\ \vec{v}\,'' \mapsto (\vec{s}\,'')^{\circ}] = [\vec{v}^{\,+} \mapsto \vec{s}^{\,+}] = {}^{\dagger}$, and $\vec{w}^{\,\Gamma} \mapsto \vec{r}^{\,\Gamma}$ is the restriction of $^\circ$ to the variables of Γ. Then clearly $(q')^{\circ'} = (q'^{\dagger'})^\circ$ where $q'^{\dagger'} \approx o'$, so $(q')^{\circ'} \approx o'^\circ \succeq x'^\eta$. Moreover $^{\circ'}$ is x',U-closed, as we may verify by case analysis:

- The terms $\vec{s}$ exist in environment $\Gamma, \vec{v}\,'$, so do not involve x' or any of the variables of W. It follows that the terms $\vec{s}^\circ$ do not involve x'.
- For any variables $v \in \vec{v}\,''$ of level k, we have $v^{\dagger'} = v^\eta$, so $(v^{\dagger'})^\circ$ cannot involve x'.
- For any variables $v \in \vec{v}\,''$ of level $< k$, the variable x' is permitted in $v^{\circ'}$ as $\vec{v}\,'' = U$. (It is because $v^{\dagger'}$ might involve x' for these v that the machinery of x, V-closed substitutions is necessary.)
- The $\vec{r}^\Gamma$ cannot involve x', since $^\circ$ is x',W-closed and Γ is disjoint from W.

This completes the verification that d is spinal. □

From the above lemma we may immediately conclude, for example, that in the setting of Lemma 22 the term d is spinal.

We are now ready for the main result of this section:

Theorem 24 *No regular normal-form term $\Gamma \vdash t \in \mathsf{A}_k$ is spinal.*

Proof Suppose for contradiction that $\Gamma \vdash t$ were a regular normal-form term-in-environment of minimal ordinal rank α in the inductive generation of $\mathcal{A}_k$ such that t is spinal, and let c be a head-spinal expression occurring in t at position $K[-]$. Recall from Proposition 19 and the subsequent discussion that t cannot contain any bound variables of level $> k$, and that the same will also hold for all other terms occurring in the derivation of $\Gamma \vdash t \in \mathsf{A}_k$.

We consider in turn the possibilities for the last rule from Definition 9 applied in the construction of $\Gamma \vdash t \in \mathsf{A}_k$. This cannot be rule 1 because $n, \bot$ are not spinal, and cannot be rule 5 because t is normal. Moreover, rules 2, 3 and 4 can be easily eliminated as follows:

- Suppose t is constructed by rule 2, say as $\lambda\vec{x}.e$. Since c is an expression, it appears within e, so that e itself is spinal. But $\Gamma, \vec{x} \vdash e$ has rank $< \alpha$, so the minimality of α is contradicted.
- Suppose t is constructed via rule 3, say as $xp_0 \ldots p_{r-1}$, and suppose c is a head-spinal subterm within t. If c occurs within some p_i, then clearly p_i is spinal and is of rank $< \alpha$, a contradiction. The only other possibility is that $c = t$, in which case $x = g$ and $r = 2$. But then by inspection of Definition 21, we see that p_0 is spinal, again contradicting the minimality of α. (If g has the modified type $0 \to (k+1) \to (k+1)$, we of course have $r = 3$ and p_1 spinal here.)
- Suppose t is constructed via rule 4, say as `case` a `of` $(i \Rightarrow e_i)$, and suppose c is a head-spinal subterm within t. If c appears within a or some e_i, then once again, this subterm is itself spinal and of lower rank. On the other hand, if t itself is head-spinal, then by Definition 21 we see that a too is spinal. Either way, the minimality of α is contradicted.

It remains to consider rule 6 (the plugging rule), and here the machinery of Lemmas 22 and 23 comes into play. Suppose that t is constructed via rule 6, say as $t = \ll \Pi_{\Gamma,Z}(e,\xi) \gg$ where $\Gamma, Z \vdash e \in \mathsf{A}_k$ and each $\Gamma, Z \vdash \xi(z) \in \mathsf{A}_k$ have rank $< \alpha$. For later convenience, to each $z_i \in Z$ let us associate the procedure $\Gamma \vdash r_i = \ll \Pi_{\Gamma,Z}(\xi(z_i),\xi) \gg$; it is then clear from the definition of plugging and the evaluation theorem that $t = \ll e[\vec{z} \mapsto \vec{r}] \gg$ and that $r_i = \ll \xi(z_i)[\vec{z} \mapsto \vec{r}] \gg$ for each i. We also have by hypothesis that c is a head-spinal expression within t at position $K[-]$; we shall focus on the head occurrence of g in c.

Since c is a subterm of t, this occurrence of g must originate from some normal-form term fragment $\Gamma, Z \vdash t_0$ involved in the above plugging (either e or some $\xi(z)$). Suppose that this occurrence of g in t_0 is as indicated by

$$\Gamma, Z \;\vdash\; t_0 \;=\; L[d]\;, \qquad \Gamma, Z, \vec{v} \;\vdash\; d \;=\; \texttt{case}\; gpq \;\texttt{of}\; (\cdots)\;,$$

where $\vec{v}$ is the local environment for $L[-]$. Writing * for $[\vec{z} \mapsto \vec{r}]$, we have $\ll \Pi(t_0,\xi) \gg = \ll t_0^\star \gg$; and if $C[-]$ is the context encapsulating the remainder of the plugging $\Pi(e,\xi)$, then we may write

$$\Gamma \vdash\; K[c] \;=\; t \;=\; \ll C[\Pi(t_0,\xi)] \gg \;=\; \ll C[t_0^\star] \gg \;=\; \ll C[L^\star[\ll d^\star \gg]] \gg\;,$$

where

$$\ll d^\star \gg \;=\; \texttt{case}\; g \ll p^\star \gg \ll q^\star \gg \;\texttt{of}\; (\cdots)\;.$$

We claim that we are in the situation of Lemma 22, taking Γ, K, d, K', c of the lemma to be respectively $\Gamma, C[L^\star[-]], \ll d^\star \gg, K, c$. Condition 1 of the lemma holds because $C[t_0^\star]$ is constructed by substitution from normal-form terms of level $\leq k$, and conditions 2 and 3 are immediate in the present setup.

By Lemma 22, we may therefore conclude that for a suitable substitution $[\vec{v} \mapsto \vec{s}]$ with $\Gamma, \vec{v}\,' \vdash \vec{s}$ regular, $\Gamma \vdash \ll \ll d^\star \gg [\vec{v} \mapsto \vec{s}] \gg$ is head-spinal. (Note that the local variables of $C[-]$ do not appear in $d^\star$, because $\Gamma, Z \vdash t_0$ and $\Gamma \vdash \vec{r}$.) Equivalently, we may say that $\ll d[\vec{v}^{\,+} \mapsto \vec{s}^{\,+}] \gg$ is head-spinal, where

$$[\vec{v}^{\,+} \mapsto \vec{s}^{\,+}] \;=\; [\vec{z} \mapsto \vec{r},\; \vec{v} \mapsto \vec{s}]\;,$$

so that $\Gamma, \vec{v}\,' \vdash \vec{s}^{\,+}$ and $\mathrm{lv}(\vec{v}^{\,+}), \mathrm{lv}(\vec{v}\,') \leq k$. (Note that the $\vec{z}$ do not appear free in $\vec{s}$, nor the $\vec{v}$ in $\vec{r}$.)

Since $\Gamma, Z, V \vdash d$ and $\Gamma, \vec{v}\,' \vdash \vec{s}^{\,+}$ are regular, we are in the situation of Lemma 23, so may conclude that d itself is spinal, and hence that t_0 is spinal. Once again, this gives a contradiction, as $\Gamma, Z \vdash t_0 \in \mathsf{A}_k$ has rank $< \alpha$. This completes the proof. □

In particular, since the expression $C[g,x]$ mentioned at the start of the section is spinal, we may conclude that $g, x \vdash C[g,x] \notin \mathsf{A}_k$ and hence that $Y_{k+1} \notin \mathsf{A}_k$. It also follows also that $Y_{0\to(k+1)} \notin \mathsf{A}_k$, since $Y_{0\to(k+1)}$ is readily definable from Y_{k+1} even in PCF_0. Since by Theorem 17 every PCF^{Ω}_k-definable procedure lives in A_k, we have established Theorem 5.

In the following section, we will exploit the fact that Theorem 24 also holds relative to the modified notion of spinal term appropriate to a variable $g : 0 \to (k+1) \to (k+1)$.

5 Non-definability in the extensional model

To obtain corresponding non-definability results for SF rather than SP^0, one must show not only that the canonical procedures Y_τ considered above are not PCF$_k$-definable, but also that no extensionally equivalent procedures are. It is easy to see that there are indeed many other procedures Z with the same extension as Y_τ. To give a trivial example, let us revisit the term $C[g,x]$ introduced at the start of Section 4, which we here define by

$$C[g,x] \;=\; \texttt{case}\; A[g,x]\; \texttt{of}\; (i \Rightarrow i)\;, \qquad A[g,x] \;=\; g(\lambda x'.C[g,x'])x^\eta\;.$$

Then the canonical procedure for the fixed point operator is $\lambda gx.C[g,x]$, but another candidate is

$$Z_0 \;=\; \lambda gx.\,\texttt{case}\; A[g,x]\; \texttt{of}\; (i \Rightarrow C[g,x])\;.$$

Intuitively, this procedure computes the desired value twice, discarding the first result. As a slightly more subtle example, consider the procedure

$$Z_1 \;=\; \lambda gx.\,\texttt{case}\; g(\lambda x'.\texttt{case}\; A[g,x]\; \texttt{of}\; (i \Rightarrow C[g,x']))x^\eta\; \texttt{of}\; (k \Rightarrow k)\;.$$

Here, within the $\lambda x'$ subterm, we have smuggled in a repetition of the top-level computation $A[g,x]$ before proceeding to evaluate what is really required. The effect is that $\lambda x'.\texttt{case}\; A[g,x]\; \texttt{of}\; (i \Rightarrow C[g,x'])$ may be extensionally below $\lambda x'.C[g,x']$, and this may indeed affect the result when g is applied. However, this can only happen when $Y_{k+1}gx$ is undefined anyway, so it is easy to see that Z_1 as a whole will have the same extension as Y_{k+1}.

Yet another way to construct procedures extensionally equivalent to Y_{k+1} is to vary the subterms of the form x^η (where x has type k). For instance, in the case $k = 1$, we could replace x^η by an extensionally equivalent term such as

$$X_0 \;=\; \lambda y^0.\, x(\lambda.\,\texttt{case}\; y\; \texttt{of}\; (0 \Rightarrow \texttt{case}\; x(\lambda.0)\; \texttt{of}\; (j \Rightarrow 0) \mid i+1 \Rightarrow i+1))\;.$$

This is different in character from the previous examples: rather than simply repeating the computation of xy^η, we are performing the specific computation $x(\lambda.0)$ which we can see to be harmless given that this point in the tree has been reached. Clearly, such 'time-wasting' tricks as the above may be combined and elaborated to yield more complex examples of procedures equivalent to Y.

However, all of the above are rather innocuous variations and do not really yield a fundamentally different method of computing fixed points. For example, the bodies of both Z_0, Z_1 are still head-spinal terms, and it is essentially the spines that are really computing the desired fixed point by the canonical method. This suggests that we should try to show that every procedure extensionally equivalent to Y_{k+1} or $Y_{0\to(k+1)}$ is spinal; from this it would follow easily by Theorem 24 that the fixed point functional Y_{k+1} or $Y_{0\to(k+1)}$ in SF is not definable in PCF$^\Omega_k$.

Unfortunately, we are currently unable to show this in the case of Y_{k+1}: indeed, the syntactic analysis of procedures $Z \approx Y_{k+1}$ appears to present considerable technical difficulties. We shall establish the result for the case of

$Y_{0\to(k+1)}$, although even here, it is easiest to concentrate not on $Y_{0\to(k+1)}$ itself, but on a certain functional Φ that is readily definable from it.

Specifically, within PCF_{k+1}, let us define

$$\begin{array}{rcl} \Phi & : & (0 \to (k+1) \to (k+1)) \to (0 \to (k+1)) \\ \Phi\, g^{0\to(k+1)\to(k+1)} & = & Y_{0\to(k+1)}\,(\lambda f^{0\to(k+1)}.\lambda n.\, g\, n\, (f(suc\ n)))\ , \end{array}$$

so that informally

$$\Phi\, g\, n \ = \ g(n, g(n+1, g(n+2, \cdots)))\ .$$

(This generalizes the definition of the functional F mentioned at the end of Section 2.) For each $n \in \mathbb{N}$, let $g \vdash \Phi_n[g] = \Phi\, g\, \widehat{n} : k+1$, and let p_n denote the canonical NSP for $\Phi_n[g]$ (that is, the one arising from the above PCF definition via the standard interpretation in SP^0). These procedures may be defined simultaneously by:

$$g \ \vdash \ p_n \ = \ \lambda x^k.\, \texttt{case}\ g\,(\lambda.n)\, p_{n+1}\, x^\eta\ \texttt{of}\ (i \Rightarrow i) \ : \ k+1\ .$$

By a syntactic analysis of the possible forms of (simple) procedures $g \vdash q \approx p_n$, we will show that any such q is necessarily spinal. Here we have in mind the modified notion of spinal term that is applicable to terms involving a global variable $g : \rho$, where $\rho = 0 \to (k+1) \to (k+1)$ (see the explanation following Definition 21). Using Theorem 24 (understood relative to this modified setting), it will then be easy to conclude that within SF, the element $[\![\lambda g.\, \Phi_0[g]]\!]$, and hence $Y_{0\to(k+1)}$ itself, is not PCF^Ω_k-definable in SF.

To show that any $q \approx p_n$ is head-spinal, our approach will be as follows. First, we show that any such q must broadly resemble p_n in at least its top-level structure, in that q must have the form $\lambda x.\,\texttt{case}\ garo\ \texttt{of}\ (\cdots)$, where the arguments a, r, o are closely related to the corresponding arguments $(\lambda.n), p_{n+1}, x^\eta$ occurring within p_n. We do this by showing that if q were to deviate in any way from this prescribed form, we would be able to cook up procedures $G \in \mathsf{SP}^0(\rho)$ and $X \in \mathsf{SP}^0(k)$ manifesting an extensional difference between q and p_n, i.e. such that $q[g \mapsto G]X \not\approx p_n[g \mapsto G]X$. (Contrary to our usual convention, we will here use the uppercase letters G, X to range over normal-form closed procedures that may be substituted for g, x respectively.) In particular, we shall establish a sufficiently close relationship between r and p_{n+1} that the same analysis can then be iterated to arbitrary depth, showing that q has a spinal structure as required.

The main complication is that r need not superficially resemble p_{n+1}, since within r, the crucial application of g that effectively computes the value of p_{n+1} may be preceded by other 'time-wasting' applications of g (the idea is illustrated by the example Z_1 above). However, it turns out that such time-wasting subterms $ga^1r^1o^1$ must be of a certain kind if the extensional behaviour $q \approx p_n$ is not to be jeopardized: in particular, the first argument a^1 must evaluate to some $i < n$. (As in the example of Z_1, the idea is that if the subterm $ga^1r^1o^1$ merely repeats some 'outer' evaluation, it will make no overall difference to the extension if the evaluation of this subterm does not terminate.) In order to formulate the relationship between r and p_{n+1}, we therefore need a means of skipping past such time-wasting applications in order to reach the application of g that does the real work. This is achieved with the help of a *masking* operator $\mu_{n,n'}$, which (for any $n \leq n'$) overrides the behaviour of g on numerical arguments $n \leq i < n'$ with a trivial behaviour returning the dummy value 0.

We now proceed to our formal development. As a brief comment on notation, we recall from Section 2 that the relations $\approx$ and $\preceq$ of observational equivalence and inequality make sense not just for elements of SP^0 but for arbitrary meta-terms (including applications), closed or otherwise. Throughout this section, for typographical convenience, we will tend to express the required relationships mostly at the level of meta-terms, writing for instance $pq \approx n$ rather than the equivalent $\ll pq \gg = \lambda.n$ or $p \cdot q = \lambda.n$. We shall also perpetrate other mild abuses of notation, such as writing a procedure $\lambda.n$ simply as n (except for special emphasis), $\lambda\vec{x}.\bot$ as $\bot$, x^η as x, a meta-expression $\texttt{case}\ A\ \texttt{of}\ (i \Rightarrow i)$ just as A, and abbreviating a substitution $[g \mapsto G,\, x \mapsto X]$ to $[G, X]$.

We shall say that $G \in \mathsf{SP}^0(0 \to (k+1) \to (k+1))$ is *strict* if $G\bot ro \approx \bot$ for any r, o. Clearly, G is strict iff $G \approx \lambda izx.\, \texttt{case}\ i\ \texttt{of}\ (j \Rightarrow G(\lambda.j)z^\eta x^\eta)$. In connection with meta-terms with free variable g, we shall write $T \approx' T'$ to mean that $T[g \mapsto G] \approx T'[g \mapsto G]$ for all *strict* G; the notation $\preceq'$ is used similarly. We shall actually analyse the syntactic forms of procedures $g \vdash q$ based on the assumption that $q \succeq' p_n$.

We shall say a procedure $g \vdash q$ is *simple* if for every application $garo$ appearing within q, the first argument a is a numeral $\lambda.n$. The following observation allows us to simplify our analysis of terms somewhat.

Proposition 25 *If A_k contains some procedure $g \vdash q \succeq' p_n$, then A_k contains some simple q with this property.*

Proof Suppose $g \vdash q \in \mathsf{A}_k$ where $q \succeq' p_n$, and write

$$S[g] \;=\; \lambda izx.\, \texttt{case}\ i\ \texttt{of}\ (j \Rightarrow g(\lambda.j)z^\eta x^\eta)\ .$$

Then $g \vdash S[g] \in \mathsf{A}_k$ since $S[g]$ is well-founded, so also $g \vdash q' = \ll q[g \mapsto S[g]] \gg\ \in \mathsf{A}_k$ by Corollary 15. But clearly $q \approx' q'$ since $S[G] \approx G$ for all strict G, so $q' \succeq' p_n$. Finally, q' is clearly simple: every occurrence of g within $q[g \mapsto S[g]]$ has a numeral as its first argument, so the same will be true of $\ll q[g \mapsto S[g]] \gg$. $\square$

For any $n \leq n'$, let us define the *masking* $\mu_{n,n'}(g)$ of g to be the following procedure term:

$$g^\rho \;\vdash\; \mu_{n,n'}(g) \;=\; \lambda izx.\, \texttt{case}\ i\ \texttt{of}\ (n \Rightarrow 0 \mid \cdots \mid n'-1 \Rightarrow 0 \mid - \Rightarrow gizx) \;:\; \rho\ .$$

We may also write $\mu_{n,n'}(P)$ for $\mu_{n,n'}(g)[g \mapsto P]$ if P is any meta-procedure of type ρ. We write $\mu_{n,n+1}$ simply as μ_n; note also that $\mu_{n,n}(g) \approx' g$. Clearly $\mu_n(\mu_{n+1}(\cdots(\mu_{n'-1}(g))\cdots)) \approx \mu_{n,n'}(g)$.

We shall say that a meta-term $P : \rho$ is *trivial at n* if $P(\lambda.n)\bot\bot \approx 0$. Note that $\mu_{n,n'}(G)$ is trivial at each of $n, \ldots, n'-1$ for any G; indeed, G is trivial at $n, \ldots, n'-1$ iff $G \succeq \mu_{n,n'}(G)$.

The following lemma now implements our syntactic analysis of the top-level structure of simple procedures $q \succeq' p_n$.

Lemma 26 *Suppose $g \vdash q$ is simple and $q \succeq' p_n$. Then q has the form $\lambda x^k.\,\texttt{case}\ garo\ \texttt{of}\ (\cdots)$, where:*

1. *$a = \lambda.n$,*
2. *$o[g \mapsto G] \succeq x^\eta$ whenever G is trivial at n,*
3. *$r[g \mapsto \mu_n(g),\, x \mapsto X] \succeq' p_{n+1}$ for any X.*

Proof Suppose $q = \lambda x^k.e$. Clearly e is not constant since $q \succeq p_n$; and if e had head variable x, we would have $q[g \mapsto \lambda izx.\,\texttt{case}\ i\ \texttt{of}\ (- \Rightarrow 0)](\lambda w.\bot) = \bot$, whereas $p_n[g \mapsto \lambda izx.\,\texttt{case}\ i\ \texttt{of}\ (- \Rightarrow 0)](\lambda w.\bot) = 0$, contradicting $q \succeq' p_n$. So e has the form $\texttt{case}\ garo\ \texttt{of}\ (\cdots)$.

For claim 1, we have $a = \lambda.m$ for some $m \in \mathbb{N}$ because q is simple. Suppose that $m \neq n$, and consider

$$G' \;=\; \lambda izx.\,\texttt{case}\ i\ \texttt{of}\ (n \Rightarrow 0 \mid - \Rightarrow \bot)\ .$$

Then for any X, clearly $q[G']X \approx \bot$, whereas $p_n[G']X \approx G'(\lambda.n)(\cdots)X \approx 0$, contradicting $q \succeq' p_n$. Thus $a = \lambda.n$.

For claim 2, suppose that $G(\lambda.n)\bot\bot \approx 0$ but not $o[G] \succeq x^\eta$; then we may take $X \in \mathsf{SP}^0(k)$ and $u \in \mathsf{SP}^0(k-1)$ such that $Xu \approx l \in \mathbb{N}$ but $o[G,X]u \not\approx l$. Now define

$$G' \;=\; \lambda izx.\,\texttt{case}\ i\ \texttt{of}\ (j \Rightarrow \texttt{case}\ xu\ \texttt{of}\ (l \Rightarrow Gjzx \mid - \Rightarrow \bot))\ .$$

Then $G' \preceq G$, so $o^*u \not\approx l$ where $^* = [G', X]$; hence $G'a^*r^*o^* \approx \bot$ and so $q[G']X \approx \bot$. On the other hand, we have

$$\begin{array}{rcl} p_n[G']X & \approx & \texttt{case}\ G'(\lambda.n)p^*_{n+1}X\ \texttt{of}\ (i \Rightarrow i) \\ & \approx & \texttt{case}\ Xu\ \texttt{of}\ (l \Rightarrow G(\lambda.n)p^*_{n+1}X) \;\approx\; 0\ , \end{array}$$

contradicting $q \succeq' p_n$. Thus $o[G] \succeq x^\eta$.

For claim 3, suppose that $p_{n+1}[G]X' \approx l$ for some strict $G \in \mathsf{SP}^0(\rho)$ and $X' \in \mathsf{SP}^0(k)$. We wish to show that $r[\mu_n(G), X]X' \approx l$ for any X. Suppose not, and consider

$$G' \;=\; \lambda izx.\,\texttt{case}\ i\ \texttt{of}\ (n \Rightarrow \texttt{case}\ zX'\ \texttt{of}\ (l \Rightarrow 0 \mid - \Rightarrow \bot) \mid - \Rightarrow Gizx)\ .$$

Then $G' \preceq \mu_n(G)$, so $r[G', X]X' \not\approx l$. Moreover, since $a = \lambda.n$ by claim 1, we see that $G'a^*r^*o^* \approx \bot$, where $^* = [G', X]$, so that $q[G']X \approx \bot$. On the other hand, we have

$$\begin{array}{rcl} p_n[G']X & \approx & \texttt{case}\ G'(\lambda.n)p^*_{n+1}X\ \texttt{of}\ (i \Rightarrow i) \\ & \approx & \texttt{case}\ p^*_{n+1}X'\ \texttt{of}\ (l \Rightarrow 0)\ . \end{array}$$

Here, since p_{n+1} does not contain x free, we have $p^*_{n+1} = p_{n+1}[G']$. But it is easy to see that $p_{n+1}[G'] \approx p_{n+1}[G]$, since every occurrence of g within p_{n+1} is applied to $\lambda.n'$ for some $n' > n$, and for all such n' we have $G'(\lambda.n') \approx G(\lambda.n')$. (Since p_{n+1} contains infinitely many applications of g, an appeal to continuity is formally required here.) But $p_{n+1}[G]X' \approx l$ by assumption; thus $p^*_{n+1}X' \approx l$, allowing us to complete the proof that $p_n[G']X \approx 0$. Once again, this contradicts $q \succeq' p_n$, so claim 3 is established. $\Box$

In the light of Appendix A, one may strengthen claim 2 of the above lemma by writing $o[g \mapsto G] \approx x^\eta$. This gives a fuller picture of the possible forms of terms $q \approx p_n$, but is not needed for showing that such q are spinal.

We have now completed a circle, in the sense that claim 3 tells us that $\ll r[g \mapsto \mu_n(g), x \mapsto X] \gg$ itself satisfies the hypothesis for q (with $n+1$ in place of n), and so can be iteratively subjected to the same analysis. However, it still remains to see what this analysis tells us about the syntactic form of r itself, as distinct from $\ll r[g \mapsto \mu_n(g)] \gg$. (In the light of claim 3, the variable x may be safely ignored here.)

Lemma 27 *Suppose $g \vdash q$ is simple and $q[g \mapsto \mu_{n,n'}(g)] \succeq' p_{n'}$. Then q has the form $\lambda x^k.\, E[\texttt{case}\ garo\ \texttt{of}\ (\cdots)]$, where:*

1. *$E[-]$ has empty local variable environment,*
2. *$a = \lambda.n'$,*
3. *$o[g \mapsto G] \succeq x^\eta$ whenever G is trivial at $n, \cdots, n'$,*
4. *$r[g \mapsto \mu_{n,n'+1}(g),\ x \mapsto X] \succeq' p_{n'+1}$ for any X.*

Proof Let $q' = \ll q[g \mapsto \mu_{n,n'}(g)] \gg$. Under the above hypotheses, we have by Lemma 26 that q' is of the form $\lambda x.\, \texttt{case}\ ga'r'o'\ \texttt{of}\ (\cdots)$, where $a' = \lambda.n'$, $o'[g \mapsto G] \succeq x^\eta$ whenever G is trivial at n', and $r'[g \mapsto \mu_{n'}(g)] \succeq' p_{n'+1}$. Write q as $\lambda x.\, E[\texttt{case}\ garo\ \texttt{of}\ (\cdots)]$ where the displayed occurrence of g originates the head g of q' via the substitution $^\dagger = [g \mapsto \mu_{n,n'}(g)]$.

Suppose that the hole in $E[-]$ appeared within an abstraction $\lambda y.-$; then the hole in $E[g \mapsto \mu_{n,n'}(g)][-]$ would likewise appear within such an abstraction. Moreover, the evaluation of $q[g \mapsto \mu_{n,n'}(g)]$ consists simply of the contraction of certain expressions $\mu_{n,n'}(g)(\lambda.m)r''o''$ to either 0 or $g(\lambda.m)r''o''$, followed by some reductions $\texttt{case}\ 0\ \texttt{of}\ (i \Rightarrow e_i) \rightsquigarrow e_0$; thus, any residue in q' of the critical g identified above will likewise appear underneath λy. But this is impossible, because the head g of q' is a residue of this g by assumption. This establishes condition 1.

In the light of this, by Lemma 22(i) we have $a' \approx a^\dagger$, $o' \approx o^\dagger$ and $r' \approx r^\dagger$. But since q is simple, a is a numeral, so $a = \lambda.n'$, giving condition 2. For condition 3, suppose G is trivial at $n, \ldots, n'$. Then $G \succeq \mu_{n,n'+1}(G)$, so that

$$o[g \mapsto G] \ \succeq\ o[g \mapsto \mu_{n,n'}(\mu_{n'}(G))] \ \approx\ o^\dagger[g \mapsto \mu_{n'}(G)] \ \approx\ o'[g \mapsto \mu_{n'}(G)] \ \succeq\ x^\eta\ ,$$

since $\mu_{n'}(G)$ is trivial at n'. Condition 4 also holds since $r'[g \mapsto \mu_{n'}(g)] \approx r^\dagger[g \mapsto \mu_{n'}(g)] \approx r[g \mapsto \mu_{n,n'+1}(g)]$, where $r'[g \mapsto \mu_{n'}(g)] \succeq' p_{n'+1}$. $\Box$

Corollary 28 *If $g \vdash q$ is simple and $q \succeq' p_n$, then q is spinal (in the modified sense).*

Proof Since condition 3 of the above lemma matches its hypotheses, starting from the assumption that $q \approx' q[g \mapsto \mu_{n,n}(g)] \succeq' p_n$, we may apply the lemma iteratively to obtain a spinal structure as prescribed by Definition 21 (subject to the relevant adjustments for $g : 0 \to (k+1) \to (k+1)$). $\Box$

Thus, if there exists a k-acceptable term $t \approx \lambda g.p_0$, for instance, then by Proposition 25 there is also a simple such term t', and by Corollary 28, this t' will be spinal in the modified sense, which contradicts Theorem 24 (understood relative to the modified setting). We conclude that no $t \approx \lambda g.p_0$ is present in A^0_k; by Theorem 17, this means that no such t can be definable in PCF^{Ω}_k. Since the interpretation of PCF^{Ω}_k in SF factors through SP^0, this in turn means that within the model SF, the element $[\![\lambda g.\, \Phi_0[g]]\!]$ is not definable in PCF^{Ω}_k. On the other hand, this element is obviously definable relative to $Y_{0\to(k+1)} \in \mathsf{SF}$ even in PCF_1, so the proof of Theorem 2 is complete.

6 Related and future work

6.1 Other hierarchies of Y combinators

Previous results to the effect that Y combinators for level $k+1$ are in some way not definable from those for level k have been obtained by Damm [6] and Statman [20]. It is convenient to discuss the latter of these first.

Statman works in the setting of the simply typed *λY-calculus*, essentially the pure λ-calculus extended with constants $Y_\sigma : (\sigma \to \sigma) \to \sigma$ and reduction rules $Y_\sigma M \rightsquigarrow M(Y_\sigma M)$. He gives a succinct proof that Y_{k+1} is not definable from Y_k up to computational equality, based on the following idea. If Y_{k+1} were definable from Y_k, it would follow that the recursion equation $Y_{k+1}g = g(Y_{k+1}g)$ could be derived using only finitely many uses of the equation $Y_k M = M(Y_k)M$ (say m of them). It would then follow, roughly speaking, that mn recursion unfoldings for Y_k would suffice to fuel n recursion unfoldings for Y_{k+1}. On the other hand, it can be shown that the size of normal-form terms definable using n unfoldings of Y_{k+1} (as a function of the size of the starting term) grows faster than can be accounted for with mn unfoldings of Y_k.

The language λY is seemingly less powerful than PCF,[5] although this is perhaps not the most essential difference between Statman's work and ours. More fundamentally, Statman's method establishes the non-definability only up to computational equality (that is, the equivalence relation generated by the reduction rules), whereas we have been concerned with non-definability modulo observational (or extensional) equivalence. Even for non-definability in SP^0, it would seem an approach along Statman's lines would be unlikely to yield much information, since there is no reason why the number of unfoldings of Y_k

[5]One might consider translating PCF into λY by representing natural numbers as Church numerals; however, it appears that predecessor is not λY-definable for this representation.

required to generate the NSP for Y_{k+1} to depth n should not grow dramatically as a function of n. Nonetheless, it is striking that methods so markedly different from ours can be used to obtain closely related results.

A result very similar to Statman's was obtained earlier in Damm [6], but by a much more indirect route as part of a far-reaching investigation of the theory of tree languages. In Damm's setting, programs are *recursion schemes*—essentially, families of simultaneous (and possibly mutually recursive) defining equations in typed λ-calculus—but in essence these can be considered as terms of λY relative to some signature consisting of typed constants. Any such program can be expanded to an infinite tree (essentially a kind of Böhm tree), and Damm's result (Theorem 9.8 of [6]) is that up to tree equality, there are programs definable by level $k+1$ recursions but not level k ones. The notion of tree equality here is more generous than computational equality (making Damm's result somewhat stronger than Statman's), although still much more fine-grained (in the case of a signature for PCF) than equality in SP^0, let alone in SF. It therefore seems unlikely, once again, that an approach to our theorems from this direction will be forthcoming.

What is certainly of interest, however, is the connection that Damm establishes between programming-style recursion schemes on the one hand, and hierarchies of language families on the other. Specifically, Damm establishes a close connection between tree grammars and λ-calculus, showing that the languages inhabiting level k of the *OI-hierarchy*[6] are exactly those that can be generated by level k recursion schemes. He also shows that the OI-hierarchy is strict, and it is from this that the superiority of level $k+1$ schemes over level k ones is deduced. Thus, whilst the statement of Damm's theorem on the recursion scheme hierarchy has a syntactic flavour (it is formulated in terms of tree equality), the theorem is reached via an essentially 'semantic' investigation of the expressive power of such schemes within a particular domain, namely that of language definitions. Of course, this is a long way from the setting of computable functionals over $\mathbb{N}_\perp$ that we consider in the present paper,[7] and it is not clear whether there is any substantial connection to be made here, but the applications of higher-type recursions in language theory are certainly intriguing.

6.2 Relationship to game semantics

Next, we comment briefly how our work relates to the known *game models* of PCF [1, 7]. It is known that these models are in fact isomorphic to our SP^0, although the equivalence between the game-theoretic definition of application and our own is mathematically non-trivial (the situation is discussed in [12, Section 7.4]). This raises the obvious question of whether our proofs could

[6]This is a language hierarchy whose first two levels are the sets of regular and context-free languages respectively. OI stands for *outermost-innermost*.

[7]One noteworthy difference is that even at base type, more objects are definable by $(k+1)$-recursions than k-recursions in Damm's setting, whereas for us, all base type objects are of course trivially definable in PCF_0.

be conducted equally well, or better, in a game-semantical setting. Whilst a direct translation is presumably possible, our impression (at present) is that sequential procedures allow one to work at a slightly higher level of abstraction, and that the calculus of meta-terms plays an important role here. (At the very least, our intuition has come from thinking about meta-term reduction rather than dialogues between players.) However, a closer look at game-semantical approaches would be needed in order to judge whether either approach really offers some genuine mathematical advantage over the other.

6.3 Sublanguages of PCF: further questions

We now turn our attention to some potential extensions and generalizations of our work.

So far, we have considered only a coarse stratification of types in terms of their levels, and there is certainly scope for a more fine-grained analysis of types within each level. Naturally, this is closely related to the task of mapping out the embeddability relation between types (as in Subsection 2.4) in finer detail.

Even at level 1, there is some interest here. The following information can be extracted from our present analysis with some modest adaptations; we write $\mathtt{N}^r \to \mathtt{N}$ for the type $\mathtt{N} \to \mathtt{N} \to \cdots \to \mathtt{N}$ with r argument positions.

Theorem 29 *Suppose $r \geq 1$.*
(i) In SP^0, the element $Y_{\mathtt{N}^{r+1}\to\mathtt{N}}$ is not PCF_0^Ω definable from $Y_{\mathtt{N}^r\to\mathtt{N}}$.
(ii) In SF, the element $Y_{\mathtt{N}^{r+2}\to\mathtt{N}}$ is not PCF_0^Ω definable from $Y_{\mathtt{N}^r\to\mathtt{N}}$

However, this leaves us with a small gap for SF: e.g. we have not shown either that $Y_{\mathtt{N}\to\mathtt{N}}$ is strictly weaker than $Y_{\mathtt{N}^2\to\mathtt{N}}$ or that $Y_{\mathtt{N}^2\to\mathtt{N}}$ is strictly weaker than $Y_{\mathtt{N}^3\to\mathtt{N}}$, although (classically) at least one of these must be the case. We expect that a more delicate analysis will allow us to fill this gap, and also to close an earlier gap by showing that Y_{k+1} is not PCF_k^Ω-definable in SF. One can also envisage even more fine-grained hierarchy obtained by admitting other base types such as the booleans or unit type.

A closely related task is to obtain analogous results for the *call-by-value* interpretation of the simple types (as embodied in Standard ML, for example). As is shown in [12, Section 4.3], a call-by-value (partial) type structure SF_v can be constructed by fairly general means from SF: here, for example, $\mathsf{SF}_v(\overline{1})$ consists of all partial functions $\mathbb{N} \rightharpoonup \mathbb{N}$ rather than (monotone) total functions $\mathbb{N}_\perp \to \mathbb{N}_\perp$, and $\mathsf{SF}_v(\overline{2})$ consists of partial functions $\mathbb{N}_\perp^\mathbb{N} \rightharpoonup \mathbb{N}$. From known results on the interencodability of call-by-name and call-by-value models (see [12, Section 7.2]), it is easy to read off the analogue of Corollary 3 for $\mathsf{SF}_v^{\mathrm{eff}}$; however, more specific results on the relative strengths of various Y combinators within SF_v would require more detailed reworking of our arguments.

Of course, one can also pose relative definability questions for other elements besides Y combinators (see also Subsection 2.5). For instance, it is natural to consider the *higher-order primitive recursors rec_σ* of System T, as well as the closely related *iterators* $iter_{\sigma\tau}$:

$$\begin{array}{rcl} rec_\sigma & : & \sigma \to (\sigma \to \mathtt{N} \to \sigma) \to \mathtt{N} \to \sigma \ , \\ iter_{\sigma\tau} & : & (\sigma \to (\sigma + \tau)) \to \sigma \to \tau \ . \end{array}$$

Here $iter_{\sigma\tau} f\, x$ will start from x and repeatedly apply f until (if ever) a value in τ rather than σ is reached.[8] It is not hard to see that up to extensional equality, any $iter_{\sigma\tau}$ is definable from $iter_{\sigma\mathtt{N}}$, and this is in turn definable from rec_σ and min; in particular, any $iter_{\mathtt{N}\tau}$ is definable in the language $\mathrm{T}_0 + min$ mentioned in Subsection 2.5. It is also clear that rec_σ is definable from $Y_{\mathtt{N}\to\sigma}$.

Since the NSPs for the rec_σ are all well-founded, they are k-acceptable in our sense, so our proofs in fact show that $Y_{0\to(k+1)} \in \mathsf{SF}$ is not definable even in PCF_k^Ω extended with System T recursors (and iterators) for all types. The dual question, roughly speaking, is whether any or all of the recursors rec_σ or iterators $iter_{\sigma\mathtt{N}}$ for types σ of level $k+1$ are definable in PCF_k. We conjecture that they are not, and that this can be shown by suitably adjusting the definition of our substructure A_k so as to exclude such $iter_{\sigma\mathtt{N}}$. (This would answer Question 2 of [4, Section 5].) One could also look for substructures that more precisely determine the strength of various *bar recursion* operators or the *fan functional*. All in all, our experience leads us to expect that many further substructures of SP^0 should be forthcoming, leading to a harvest of non-definability results exhibiting a rich 'degree structure' among PCF-computable functionals.

Another very natural kind of question is the following: given a particular sublanguage $\mathcal{L}$ of PCF, what is the *simplest* possible type for an element of $\mathsf{SF}^{\mathrm{eff}}$ that is not definable in $\mathcal{L}$? Or to look at it another way: given a type σ, what is the smallest natural sublanguage of PCF that suffices for defining all elements of $\mathsf{SF}^{\mathrm{eff}}(\sigma)$? Here the analysis of [12, Section 7.1] yields several positive definability results, whilst the analysis of the present paper provides ammunition on the negative side. The current state of our knowledge is broadly as follows:

- For first-order types σ, even the language Iter_1 (consisting of the basic functions plus the iterators $iter_{\mathtt{N}^t\,\mathtt{N}}$ for $t \in \mathbb{N}$) suffices for defining all elements of $\mathsf{SF}^{\mathrm{eff}}(\sigma)$; likewise, the oracle version Iter_1^Ω suffices for $\mathsf{SF}(\sigma)$.
- For second-order types σ of the special form $(\mathtt{N} \to \mathtt{N})^r \to \mathtt{N}$, Iter_1^Ω still suffices for $\mathsf{SF}(\sigma)$; however, this result is non-constructive, and Iter_1 does not suffice for $\mathsf{SF}^{\mathrm{eff}}(\sigma)$. (This follows from recent unpublished work of Dag Normann.)
- For general second-order types, Iter_1^Ω no longer suffices, but the languages PCF_1, PCF_1^Ω suffice for $\mathsf{SF}^{\mathrm{eff}}$, SF respectively— indeed, even the single recursion operator $Y_{\mathtt{N}\to\mathtt{N}}$ is enough here.
- For third-order types, we do not know whether PCF_1 suffices (for $\mathsf{SF}^{\mathrm{eff}}$). We do know that PCF_2 suffices, and that $Y_{\mathtt{N}\to\mathtt{N}}$ alone is not enough.

[8]The sum type $\sigma + \tau$ is not technically present in our language, but is easily encodable as a retract of an existing type—see [12, Section 4.2].

- For types of level $k \geq 4$, PCF_{k-3} does not suffice, but PCF_{k-2} does.

Again, there is scope for a more fine-grained view of the hierarchy of types.

It is worth pausing to ask what wider repercussions all these results are likely to have. On the one hand, it may be that our analysis allows us identify some non-trivial extensional properties of functionals definable in a restricted language that are not shared by all PCF-definable functionals. (A result in this genre was obtained in Berger [4].) Given that restricted languages like Iter_1 or at most $\mathrm{PCF}_0 + Y_{\mathtt{N}\rightarrow\mathtt{N}}$ appear to suffice for all ordinary programming purposes, such extensional properties could in principle turn out to be useful for the analysis of realistic programs. On the other hand, our results indicate that even PCF_1 only scratches the surface of what is possible in principle within PCF, and it may be that our analysis can provide hints of where to look for exciting examples of PCF functionals that make essential use of Y_2 (for instance). We suspect, however, that the main long-term benefit of our work will be more conceptual, and that its main value will consist simply in the separating out of different aspects of what is implicit in the powerful concept of 'general recursion'.

6.4 Other models and languages

We have so far concentrated almost entirely on PCF-style sequential computation. To conclude, we briefly consider which other notions of higher-order computation are likely to present us with an analogous situation.

As already noted at the end of Subsection 2.4, several extensions of PCF studied in the literature present a strikingly different picture: in these settings, universal types exist quite low down, and as a consequence, only Y combinators of low type (along with the other constants of the language) are required for full definability. There is, however, one important language which appears to be more analogous to pure PCF in these respects, namely an extension with *local state* (essentially Reynolds's *Idealized Algol*). This language was studied in [2], where a fully abstract game model was provided (consisting of well-bracketed but possibly non-innocent strategies). Unpublished work by Jim Laird has shown that there is no universal type in this setting. We would conjecture also that the recursion hierarchy for this language is strict; this would constitute a significant strengthening of our present results.

Related questions also arise in connection with *total* functionals. Consider, for example, the type structure Ct of total continuous functionals, standardly constructed as the extensional collapse (relative to $\mathbb{N}$) of the Scott model PC of partial continuous functionals. It is shown by Normann [14] that every effective element of Ct is represented by a PCF-*definable* element of PC, and the proof actually shows that PCF_1 suffices here. (The further generalization of these ideas by Longley [10] actually makes mild use of second-order recursions as in PCF_2; we do not know whether these can be eliminated.) Thus, in this setting, only recursions of low type are needed to obtain all computable functionals. Similar remarks apply to the total type structure HEO, obtained as the extensional collapse of $\mathsf{PC}^{\mathrm{eff}}$.

On the other hand, one may consider the *Kleene computable* functionals over Ct, or over the full set-theoretic type structure S, as defined by the schemes S1–S9. As explained in [12, Chapter 6], sequential procedures can be seen as abstracting out the algorithmic content common to both PCF-style and Kleene-style computation (note that Kleene's S9 in some sense does duty for the Y combinators of PCF). This naturally suggests that our strict hierarchy for PCF may have an analogue for the Kleene computable functionals (say over S or Ct), where at level k we consider the evident restriction of S9 to types of level $\leq k$. We conjecture that this is indeed the case, although the required counterexamples may be more difficult to find given that we are limited to working with total functionals.

Appendix A: Super-identity procedures

In the course of our proof, we have frequently encountered assertions of the form $p \succeq x^\eta$ for various procedures $x^k \vdash p$. Although not necessary for our main argument, it is natural to ask whether there are any such procedures other than those for which $p \approx x^\eta$. The following theorem shows that the answer is no: in other words, no procedure $\lambda x.p : \overline{k} \to \overline{k}$ can extensionally 'improve on' the identity function. We present this as a result of some interest in its own right, whose proof is perhaps less trivial than one might expect.

Recall that $\preceq$ denotes the extensional order on SF, as well as the associated preorder on SP^0. Within SF, we will write $f \prec f'$ to mean $f \preceq f'$ but $f \neq f'$; we also write $f \sharp f'$ to mean that f, f' have no upper bound with respect to $\preceq$.

We call an element of SP^0 *finite* if it is a finite tree once branches of the form $i \Rightarrow \bot$ have been deleted. We say an element of SF is *finite* if it is represented by some finite element of SP^0. We write $\mathsf{SP}^{0,\mathrm{fin}}(\sigma), \mathsf{SF}^{\mathrm{fin}}(\sigma)$ for the set of finite elements in $\mathsf{SP}^0(\sigma), \mathsf{SF}(\sigma)$ respectively.

Theorem 30 *(i) If $f \in \mathsf{SF}^{\mathrm{fin}}(k)$ and $f \prec f'$, then there exists $f'' \sharp f'$ with $f \prec f''$.*

(ii) If $\Phi \in \mathsf{SF}(k \to k)$ and $\Phi \succeq id$, there can be no $f \in \mathsf{SF}(k)$ with $\Phi(f) \succ f$; hence $\Phi = id$.

Proof Both statements are easy for $k = 0, 1$, so we will assume $k \geq 2$.

(i) Suppose $f \prec f'$ where f is finite. Then for some $g \in \mathsf{SF}(k-1)$ we have $f(g) = \bot$ but $f'(g) = n \in \mathbb{N}$, say, and by continuity in SP^0 we may take g here to be finite. Take $p, q \in \mathsf{SP}^{0,\mathrm{fin}}$ representing f, g respectively; we may assume that p, q are 'pruned' so that every subtree containing no leaves $m \in \mathbb{N}$ must itself be a leaf $\bot$.

Case 1: $g(\bot^{k-2}) = a \in \mathbb{N}$. In this case, we may suppose that $q = \lambda x.a$. Consider the computation of $p \cdot q$. Since all calls to q trivially evaluate to a, this computation follows the rightward path through p consisting of branches $a \Rightarrow \cdots$. But since p is finite and $p \cdot q = \bot$ (because $f(g) = \bot$), this path must end in a leaf occurrence of $\bot$ within p. Now extend p to a procedure p' by replacing this leaf occurrence by some $n' \neq n$; then clearly $p' \cdot q = n'$.

Taking f'' to be the function represented by p', we then have $f \preceq f''$ and $f''(g) = n' \sharp n = f'(g)$, so $f'' \sharp f'$ (whence also $f'' \neq f$).

Case 2: $g(\bot^{k-2}) = \bot$. Take N larger than n and all numbers appearing in p, q. Define $p' \sqsupseteq p$ as follows: if $p = \lambda x.\bot$, take $p' = \lambda x.N$, otherwise obtain p' from p by replacing each case branch $j \Rightarrow \bot$ anywhere within p by $j \Rightarrow N$ whenever $j \leq N$. Extend q to q' in the same way. Note in particular that every `case` subterm within p', q' will now be equipped with a branch $N \Rightarrow N$.

Now consider the computation of $p \cdot q$. Since p, q are finite and $f(g) = \bot$, this evaluates to an occurrence of $\bot$ which originates from p or q. Since no numbers $> N$ ever arise in the computation, this occurrence of $\bot$ cannot be part of a branch $j \Rightarrow \bot$ for $j > N$, so will have been replaced by N in p' or q'. Now suppose that we head-reduce pq until $\bot$ first appears in head position, and let U be the resulting meta-term. Then it is easy to see that $p'q'$ correspondingly reduces to a meta-term U' with N in head position. (Formally, we reason here by induction on the length of head-reduction sequences not involving the rule for `case` $\bot$ `of` $(\cdots)$.)

We now claim that $p' \cdot q' = N$. Informally, this is because the head occurrence of N in U' will be propagated to the top level by the case branches $N \Rightarrow N$ within both p' and q'. More formally, let us define the set of meta-expressions *led by* N inductively as follows:

- N is led by N.
- If E is led by N, then so is `case` E `of` $(i \Rightarrow D_i)$.

We say that an NSP meta-term T is *saturated at* N if every `case` subterm within T has a branch $N \Rightarrow E$ where E is led by N. Clearly $p'q'$ is saturated at N, and it is easy to check that the terms saturated at N are closed under head reductions; thus U' is saturated at N. But we have also seen that U' has N in head position, so is led by N. Finally, an easy induction on term size shows that every finite meta-term that is led by N and saturated at N evaluates to N itself. This shows that $p' \cdot q' = N$.

To conclude, let f'', g' be the functions represented by p', q' respectively, so that $f \preceq f''$ and $g \preceq g'$. Then $f'(g') = n$, but $p'' \cdot q = N$ so $f''(g') = N \neq n$, whence $f'' \sharp f'$ (and also $f'' \neq f$).

(ii) Suppose $\Phi \succeq id$ and $\Phi(f) \succ f$ for some f. Again by continuity, we may take f to be finite. Then by (i), we may take $f'' \succ f$ such that $f'' \sharp \Phi(f)$. But this is impossible because $\Phi(f'') \succeq f''$ and $\Phi(f'') \succeq \Phi(f)$. Thus $\Phi = id$. $\Box$

It is easy to see that the above theorem holds with any finite type over $\mathtt{N}$ in place of $\overline{k}$. However, it may trivially fail if the unit type $\mathtt{U}$ is admitted as an additional base type: e.g. the function $(\lambda x.\top) \in \mathsf{SF}(\mathtt{U} \to \mathtt{U})$ strictly dominates the identity. An interesting question is whether the theorem holds for all finite types over the type $\mathtt{B}$ of booleans: note that the above proof fails here since it requires the base type to be infinite. For comparison, we mention that in other models of computation, improvements on the identity are sometimes possible for such types. For example, if $\sigma = \mathtt{B} \to \mathtt{B}$, then a functional of type $\sigma \to \sigma$ strictly dominating the identity exists in the Scott model. Indeed, such a function J can be defined in PCF augmented with the parallel conditional pif, e.g. as

$$J \;=\; \lambda f^{\sigma}.\lambda x^{\mathsf{B}}.\, \mathit{vote}(f(x), f(\mathit{tt}), f(\mathit{ff}))\;.$$

Here vote is Sazonov's voting function, definable by

$$\mathit{vote}(x,y,z) \;=\; \mathit{pif}(x, \mathit{pif}(y,\mathit{tt},z), \mathit{pif}(y,z,\mathit{ff}))\;.$$

The point is that J will 'improve' the function $\lambda x.\, \mathit{if}(x,\mathit{tt},\mathit{tt})$ to $\lambda x.\mathit{tt}$. We do not know whether phenomena of this kind can arise within the model SF.

References

[1] Abramsky, S., Jagadeesan, R., Malacaria, P.: Full abstraction for PCF. Information and Computation **163(2)**, 409–470 (2000)

[2] Abramsky, S., McCusker, G.: Game semantics. In: Schwichtenberg, H., Berger, U. (eds.) Computational Logic: Proceedings of the 1997 Marktoberdorf Summer School, pp. 1-56. Springer, Heidelberg (1999)

[3] Amadio, R., Curien, P.-L.: Domains and Lambda Calculi. Cambridge Tracts in Theoretical Computer Science 46, Cambridge University Press (1998)

[4] Berger, U.: Minimisation vs. recursion on the partial continuous functionals. In: Gärdenfors, P., Woleński, J., Kijania-Placek, K. (eds.) In the Scope of Logic, Methodology and Philosophy of Science: Volume One of the 11th International Congress of Logic, Methodology and Philosophy of Science, Cracow, August 1999, pp. 57-64. Kluwer, Dordrecht (2002)

[5] Cartwright, R., Felleisen, M.: Observable sequentiality and full abstraction. In: Proceedings of the 19th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 328-342. ACM Press, New York (1992)

[6] Damm, W.: The IO- and OI-hierarchies. Theoretical Computer Science **20**, 95-207 (1982)

[7] Hyland, J.M.E., Ong, C.-H.L.: On full abstraction for PCF: I, II, and III. Information and computation **163**, 285–408 (2000)

[8] Kleene, S.C.: Unimonotone functions of finite types (Recursive functionals and quantifiers of finite types revisited IV). In: A. Nerode and R.A. Shore (eds.), Proceedings of the AMS-ASL Summer Institute on Recursion Theory, pp. 119-138. American Mathematical Society, Providence (1985)

[9] Longley, J.R.: The sequentially realizable functionals. Annals of Pure and Applied Logic **117(1)**, 1–93 (2002)

[10] Longley, J.R.: On the ubiquity of certain total type structures. Mathematical Structures in Computer Science **17(5)**, 841–953 (2007)

[11] Longley, J.: Bar recursion is not T+min definable. Informatics Research Report EDI-INF-RR-1420, University of Edinburgh (2015)

[12] Longley, J. and Normann, D.: Higher-Order Computability. To appear in 'Computability in Europe' series, Springer (2015-16)

[13] Milner, R.: Fully abstract models of typed λ-calculi. Theoretical Computer Science **4(1)**, 1–22 (1977)

[14] Normann, D.: Computability over the partial continuous functionals. Journal of Symbolic Logic **65(3)**, 1133–1142 (2000)

[15] Normann, D., Sazonov, V.Yu., The extensional ordering of the sequential functionals. Annals of Pure and Applied Logic **163(5)** 575–603 (2012)

[16] Plotkin, G.D.: LCF considered as a programming language. Theoretical Computer Science **5(3)**, 223–255 (1977)

[17] Sazonov, V.Yu.: Expressibility in D. Scott's LCF language. Algebra and Logic **15(3)**, 192–206 (1976)

[18] Sazonov, V.Yu.: An inductive definition and domain theoretic properties of fully abstract models of PCF and PCF+. Logical Methods in Computer Science **3(3)**, 50 pages (2007)

[19] Scott, D.S.: A type-theoretical alternative to ISWIM, CUCH, OWHY. In: A collection of contributions in honor of Corrado Böhm on the occasion of his 70th birthday. (Edited version of an unpublished note first circulated in 1969.) Theoretical Computer Science **121(1-2)**, 411–440 (1993).

[20] Statman, R.: On the λY calculus. Annals of Pure and Applied Logic **130**, 325–337 (2004).