The recursion hierarchy for PCF is strict

The recursion hierarchy for PCF is strict

John Longley

June 19, 2018

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 none of the Y_σ for σ of level $k+1$ are definable in PCF_k up to observational equivalence. 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 recent 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

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

As usual, we will 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

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

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

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

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 $Y_{\overline{k+1}}$. (Fortunately, 'observational equivalence' has the same meaning for all the languages in question 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 remarking that the situation is quite different for various extensions of PCF considered in the literature, in which one may restrict to recursions at level 1 types without loss of expressivity (see Subsection 2.4).

We can phrase our result more denotationally in terms of the type structure SF of *(PCF-)sequential functionals* or its effective substructure $\mathsf{SF}^{\mathrm{eff}}$. As will be reviewed in Section 2, the latter may be conveniently characterized up to isomorphism as the closed term model for PCF modulo observational equivalence. Our result can therefore be understood as saying that more elements of $\mathsf{SF}^{\mathrm{eff}}$ (and hence of SF) are denotable in PCF_{k+1} than in PCF_k. From this we may easily infer 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).

The models SF and $\mathsf{SF}^{\mathrm{eff}}$ are *extensional*: elements of type $\sigma \to \tau$ can be considered as mathematical functions mapping elements of type σ to elements of type τ. Whilst our theorem is naturally stated in terms of these extensional models, its proof will make deep use of a more intensional model which yields SF as its extensional quotient. This intensional model of PCF has been considered many times before in the literature, for instance as the PCF *Böhm tree* model of Amadio and Curien [3], and it is known to be isomorphic to the game models of PCF given by Abramsky, Jagadeesan and Malacaria [1] and by Hyland and Ong [8]. In this paper, we choose to work with the *nested sequential procedure* (NSP) presentation of this model, as studied in detail in the recent book of Longley and Normann [17]. (We will touch briefly on the possible use of other presentations for this purpose in Section 7.) We shall denote the NSP model by SP^0; its construction will be reviewed in Section 2, but in the meantime, let us offer a high-level overview of our proof method without assuming detailed knowledge of this model.

As a motivating example, fix $k \in \mathbb{N}$, and consider the PCF term

$$\Phi_{k+1} \ : \ (\mathtt{N} \to \overline{k+1} \to \overline{k+1}) \to \mathtt{N} \to \overline{k+1}$$

given informally by

$$\Phi_{k+1}\; g\; n \;=\; g\; n\; (\Phi_{k+1}\; g\; (n+1)) \;=\; g\; n\; (g\; (n+1)\; (g\; (n+2)\; (g\; \cdots)))\ ,$$

or more formally by

$$\Phi_{k+1} \;=\; \lambda g n.\, Y_{\mathtt{N}\to\overline{k+1}}\, (\lambda f m.\, g\, m\, (f(m+1)))\, n\ ,$$

where f has type $\mathtt{N} \to \overline{k+1}$. Clearly, Φ_{k+1} is a term of PCF_{k+1}; however, we will be able to show that the element of SF that Φ_{k+1} defines is not denotable in PCF_k.

What is the essential feature of this function that puts it beyond the reach of PCF_k? To get a hint of this, we may observe from the informal definition above that Φ_{k+1} seems implicitly to involve an infinite nested sequence of calls to its argument g, and indeed the NSP model makes this idea precise. Furthermore, each call to g involves an argument of type level $k+1$ resulting from another such call. Broadly speaking, we shall refer to such a sequence of nested calls (subject to certain other conditions) as a $k{+}1$-*spine* in the NSP associated with Φ_{k+1}. As a second example, we can see from the natural recursive definition of $Y_{\overline{k+1}}$ itself that this too involves a spine of this kind:

$$Y_{\overline{k+1}}\, h \;=\; h(Y_{\overline{k+1}}\, h) \;=\; h(h(h(\cdots)))\ .$$

where h has type $\overline{k+1} \to \overline{k+1}$. In fact, we may view $Y_{\overline{k+1}}$ as a 'special case' of Φ_{k+1}, since $Y_{\overline{k+1}}\, h$ is observationally equivalent to $\Phi_{k+1}\,(\lambda n.h)\,\widehat{0}$.

A suitable general definition of $k+1$-spine turns out to be quite delicate to formulate; but having done this, it will be possible to prove that no PCF$_k$-denotable NSP contains a $k+1$-spine (Theorem 28). This will be proved by induction on the generation of such NSPs: in essence, we have to show that none of the generating operations for the interpretations of PCF$_k$ terms are capable of manufacturing spinal NSPs out of non-spinal ones. This already suffices to show that within the intensional model SP^0, the evident procedure for $Y_{\overline{k+1}}$ is denotable in PCF$_{k+1}$ but not in PCF$_k$.

However, this does not yet establish our main theorem, which concerns not SP^0 but its extensional quotient SF. For this purpose, we undertake a closer analysis of the function Φ_{k+1} defined above: we show that not only the NSP arising from the above definition, but any *extensionally equivalent* NSP, must necessarily involve a $k+1$-spine. This shows that, within SF (or $\mathsf{SF}^{\mathrm{eff}}$), the functional given by Φ_{k+1} is not denotable in PCF$_k$. This establishes Berger's conjecture that the languages PCF$_k$ form a strict hierarchy.

Since Φ_{k+1} is definable from $Y_{\mathtt{N}\to\overline{k+1}}$, the above shows that the element $Y_{\mathtt{N}\to\overline{k+1}} \in \mathsf{SF}$ is not PCF$_k$-denotable. To complete the picture, however, we would also like to know that the simpler element $Y_{\overline{k+1}} \in \mathsf{SF}$ is not PCF$_k$-denotable. We show this via a more refined version of the above analysis which is of some interest in its own right. Just as PCF is 'stratified' into sublanguages PCF$_k$, we show that each PCF$_k$ may be further stratified into sublanguages PCF$_{k,1}$, PCF$_{k,2}, \ldots$ on the basis of the 'width' of the types σ for which Y_σ is permitted (see Definition 33). An easy adaptation of our earlier proofs then shows that, for any l, there are operators Y_σ in PCF$_{k,l+2}$ that are not denotable in PCF$_{k,l}$ (the appearance of $l+2$ is admittedly a curiosity here). Since all these Y_σ are themselves readily definable from $Y_{\overline{k+1}}$, it follows that $Y_{\overline{k+1}} \in \mathsf{SF}$ itself is not denotable in PCF$_{k,l}$ for any l, and hence not in PCF$_k$. This finer analysis illustrates the remarkable richness of structure that SF has to offer.

The paper is organized as follows. In Section 2 we recall the necessary technical background on PCF and on the models SP^0 and SF, fleshing out many of the ideas outlined above. In Section 3 we obtain a convenient inductive characterization of the class of procedures denotable in (the 'oracle' version of) PCF$_k$, framed in terms of constructions on the procedures themselves. In Section 4 we introduce the central concept of a $k{+}1$-spinal procedure, and show using our inductive characterization that no PCF$_k$-denotable procedure can be $k{+}1$-spinal (this is the most demanding part of the proof). As noted above, this already shows that PCF_{k+1} denotes more elements of SP^0 than PCF_k does. In Section 5 we obtain the corresponding result for SF, showing that the element $\Phi_{k+1} \in \mathsf{SF}$, and hence $Y_{\mathtt{N}\to\overline{k+1}} \in \mathsf{SF}$, is not PCF_k-denotable. In Section 6 we adapt our methods to the more fine-grained hierarchy of languages $\mathrm{PCF}_{k,l}$ that takes account of the widths of types; this enables us to show also that $Y_{\overline{k+1}} \in \mathsf{SF}$ is not PCF_k-denotable. We conclude in Section 7 with a discussion of related and future work.

The present paper is a revised, corrected and expanded version of a University of Edinburgh technical report from July 2015, the most significant changes being the addition of the material on the pure type $\overline{k+1}$ in Section 6, and a simplified approach to characterizing a suitable substructure of SP^0 in Section 3.

I am grateful to Ulrich Berger, Martín Escardó, Dag Normann and Alex Simpson for valuable discussions and correspondence, and to Luke Ong and Colin Stirling for helping me to navigate the existing literature on λY and recursion schemes (as discussed in Section 7). Many of the participants in the Domains XII workshop in Cork and the Galop XI workshop in Eindhoven also offered valuable comments; thanks in particular to Neil Jones for drawing his work to my attention. Finally, I thank the anonymous referees for their careful work on the paper and their valuable suggestions; these have resulted in significant improvements in both the overall architecture and the formal details.

2 Background

We here summarize the necessary definitions and technical background from [17], especially from Chapters 6 and 7.

2.1 The language PCF

In [26], 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 [23]. We will work here with the same version of PCF as in [17], with the natural numbers as the only base type. Our types σ are thus generated by

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

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). Throughout the paper, we shall regard the type of a variable x as intrinsic to x, and will often write x^σ to indicate that x carries the type σ. 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:

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

We furthermore allow these reductions to be applied in certain term contexts. Specifically, the relation $\rightsquigarrow$ is inductively generated by the rules above along with the clause: if $M \rightsquigarrow M'$ then $E[M] \rightsquigarrow E[M']$, where $E[-]$ is one of the contexts

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

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 check that either $Q \rightsquigarrow^* \widehat{n}$ for some $n \in \mathbb{N}$ or the (unique) reduction path 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^Ω_k, 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

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

(It makes no difference to the relation $\simeq_\mathcal{L}$ whether we take $C[-]$ to range over single-hole or multi-hole contexts.)

Fortunately, it is easy to show that all of the above languages give rise to exactly the same relation $\simeq_\mathcal{L}$. Indeed, 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 verify that $M \simeq_{\mathrm{PCF}_0} M'$ implies $M \simeq_{\mathrm{PCF}^\Omega} M'$. We may show this by a 'syntactic continuity' argument, exploiting the idea that any of the constants Y_σ or C_f in PCF^Ω can be 'approximated' as closely as necessary by terms of PCF_0. Specifically, let us write $\bot$ for the non-terminating program $Y_0(\lambda x^0.x) : \mathtt{N}$ (a term of PCF_0), and for any type σ write $\bot_\sigma$ for the term of type σ of the form $\lambda\vec{x}.\bot$. For any $j \in \mathbb{N}$, we may then define PCF_0 terms

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

where we use 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 *(equational) context lemma*, due to Milner [18], 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 idea 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 theorems 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 specifically:*

(i) There is no closed term M of PCF_k^Ω such that $M \simeq Y_{\mathtt{N}\to(k+1)}$ (or equivalently $M \sim_0 Y_{\mathtt{N}\to(k+1)}$).

(ii) There is even no closed M of PCF_k^Ω such that $M \simeq Y_{k+1}$.

We shall obtain part (i) of this theorem at the end of Section 5, then in Section 6 resort to a more indirect method to obtain the stronger statement (ii). The theorem also holds when $k = 0$, but this rather uninteresting case does not require the methods of this paper; it will be dealt with in Subsection 2.5.

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 point of the formulation in terms of $\sim_0$ is to present our result in as manifestly strong a form as possible: there is no $M \in \mathrm{PCF}_k$ that induces the same partial function as Y_{k+1} even on closed PCF_0 terms.

A more denotational formulation of our theorem can be given in terms of the model SF of *sequential functionals*, which we 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 $\simeq$. It is easy to check that 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 $\mathsf{SF}(\mathtt{N}) \cong \mathbb{N}_\bot = \mathbb{N} \sqcup \{\bot\}$, and also that SF is *extensional*: if $f, f' \in \mathsf{SF}(\sigma \to \tau)$ satisfy $f \cdot x = f' \cdot x$ for all $x \in \mathsf{SF}(\sigma)$, then $f = f'$. Thus, up to isomorphism, each $\mathsf{SF}(\sigma \to \tau)$ may be considered simply as a certain set of functions from $\mathsf{SF}(\sigma)$ to $\mathsf{SF}(\tau)$.

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

Theorem 2 *Suppose $k \geq 1$.*
(i) The element $[\![Y_{\mathtt{N}\to(k+1)}]\!]^{\mathsf{SF}}$ is not denotable in PCF^Ω_k.
(ii) Even $[\![Y_{k+1}]\!]^{\mathsf{SF}}$ is not PCF^Ω_k-denotable.

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 $[\![Y_{k+1}]\!]$ is not PCF^Ω_k-denotable, since the equivalence relation 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, 8]) that also give rise to these structures, and experience suggests that these are mathematically natural classes of higher-order functionals. We now see that Theorem 2(i) 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. In other words, the cartesian category of PCF-computable functionals is not finitely generated.*

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-denotable, contradicting Theorem 2(i). $\square$

2.2 The model SP^0

We turn next to an overview of the *nested sequential procedure* (or NSP) model, denoted by SP^0. Further details and motivating examples are given in [17]. In some respects, however, our presentation here will be more formal than that of [17]: in particular, our discussion of bound variables and α-conversion issues will be somewhat more detailed, in order to provide a solid foundation for the delicate syntactic arguments that follow.

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. [11]), 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 [24], 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 [8] (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 [25] to give a standalone characterization of the class of sequentially computable functionals. More recently, Normann and Sazonov [21] gave an explicit construction of the NSP model in a somewhat more semantic spirit than [3], using the name *sequential procedures*. As in [17], we here add the epithet 'nested' to emphasize the contrast with other flavours of sequential computation.[1]

As in [17], our NSPs are generated by means of the following infinitary grammar, interpreted coinductively. Here $\bot$ is a special atomic symbol and n ranges over natural numbers.

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

Here we write $(i \Rightarrow e_i \mid i \in \mathbb{N})$ to indicate an infinite sequence of 'branches': $(0 \Rightarrow e_0 \mid 1 \Rightarrow e_1 \mid 2 \Rightarrow e_2 \mid \cdots)$.

We will use vector notation to denote finite (possibly empty) lists of variables or procedures: $\vec{x}$, $\vec{q}$. Our convention will be that a list $\vec{x}$ must be non-repetitive, though a list $\vec{q}$ need not be. We may use t to range over *NSP terms* of any of the above three kinds; note that a 'term' is formally a (possibly infinite) syntax tree as generated by the above grammar. A procedure $\lambda\vec{x}.\bot$ will often be abbreviated to $\bot$.

[1]A major theme of [17] is that NSPs serve equally well to capture the essence of PCF computation and that of Kleene's S1–S9 computability; this is one reason for preferring a name that is not biased towards PCF.

For the most part, we will be working with terms modulo (infinitary) α-equivalence, and most of the concepts we introduce will be stable under renamings of bound variables. Thus, a statement $t = t'$, appearing without qualification, will mean that t, t' are α-equivalent (although we will sometimes write $=$ as $=_\alpha$ if we wish to emphasize this). When we wish to work with terms on the nose rather than up to $=_\alpha$, we shall refer to them as *concrete terms*.

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 particular point within its syntax tree. Specifically, within any term t, occurrences of procedures $\lambda\vec{x}.e$ (of any type), applications $x\vec{q}$ (of the ground type $\mathtt{N}$) and expressions e (of ground type) have types that are related to the types of their constituents and of variables as usual in type λ-calculus extended by `case` expressions of type $\mathtt{N}$. We omit the formal definition here since everything works in the expected way; for a more precise formulation see [17, Section 6.1.1].

If Γ is any environment (i.e. a finite non-repetitive 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 type τ.

We shall often refer to variable environments that arise from combining several lists of variables, which may be represented by different notations, e.g. $\Gamma, V, \vec{x}$. Since such environments are required to be non-repetitive, we take it to be part of the content of a typing judgement such as $\Gamma, V, \vec{x} \vdash t\,(:\tau)$ that the entire list $\Gamma, V, \vec{x}$ is non-repetitive. However, the *order* of variables within an environment will typically be of much less concern to us (clearly our typing judgements $\Gamma \vdash T\,(:\tau)$ are robust under permutations of Γ), and we will sometimes abuse notation by identifying a finite *set* Z of variables with the list obtained from some arbitrary ordering of it.

It will also be convenient to place another condition on concrete well-typed terms (not imposed in [17]) in order to exclude *variable hiding*. Specifically, we shall insist that if $\Gamma \vdash t\,(:\tau)$ then no variable of Γ appears as a bound variable within t, nor are there any *nested* bindings within t of the same variable x. (Clearly any concrete term not satisfying this restriction is α-equivalent to one that does.) This will help to avoid confusion in the course of some delicate arguments in which questions of the identity of variables are crucial.

With these ideas in place, we may take $\mathsf{SP}(\sigma)$ to be the set of well-typed procedures of type σ modulo $=_\alpha$, and $\mathsf{SP}^0(\sigma) \subseteq \mathsf{SP}(\sigma)$ the subset constituted by the *closed* procedures (i.e. those that are well-typed in the empty environment). By inspection of the grammar for procedures, it is easy to see that $\mathsf{SP}^0(\mathtt{N}) \cong \mathbb{N}_\bot$.

As in [17], 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:

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

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; once again, a meta-term is formally a syntax tree as generated by the above grammar. (Unless otherwise stated, we use uppercase letters for general meta-terms and lowercase ones for terms.) Once again, we will normally work with meta-terms up to (infinitary) α-equivalence, but may also work with concrete meta-terms when required.

The reader is warned of an ambiguity arising from the above grammar: a surface form $\lambda.E$ may be parsed either as a meta-procedure $\lambda\vec{x}.E$ with $\vec{x}$ empty, or as a ground meta-term $(\lambda\vec{x}.E)\vec{Q}$ with both $\vec{x},\vec{Q}$ empty. Formally these are two quite distinct meta-terms, bearing in mind that meta-terms are officially syntax trees. To remedy this ambiguity, we shall therefore in practice write '()' to indicate the presence of an empty argument list $\vec{Q}$ to a meta-procedure P, so that the ground meta-term above will be written as $(\lambda.E)()$. In the absence of '()', a surface form $\lambda.E$ should always be interpreted as a meta-procedure.

Meta-terms are subject to the expected typing discipline, leading to typing judgements $\Gamma \vdash P : \sigma$, $\Gamma \vdash E$, $\Gamma \vdash G$ for meta-procedures, meta-expressions and ground meta-terms respectively. Again we omit the details: see [17, Section 6.1.1]. We shall furthermore require that well-typed concrete meta-terms are subject to the no-variable-hiding condition. We will sometimes write e.g. $\Gamma \vdash P$ to mean that P is a well-typed meta-procedure in environment Γ, if the type itself is of no particular concern to us.

There is an evident notion of simultaneous capture-avoiding *substitution* $T[\vec{x} \mapsto \vec{Q}]$ for well-typed concrete terms. Specifically, given $\Gamma, \vec{x} \vdash T\,(:\,\tau)$ and $\Gamma, \vec{y} \vdash Q_i : \sigma_i$ for each $i < r$, where $\vec{x} = x_0^{\sigma_0}, \ldots, x_{r-1}^{\sigma_{r-1}}$, we will have $\Gamma, \vec{y} \vdash T[\vec{x} \mapsto \vec{Q}]\,(:\,\tau)$. Note that this may entail renaming of bound variables both within T (in order to avoid capture of variables in $\vec{y}$) and in the Q_i (in order to maintain the no-hiding condition for variables bound within T). The details of how this renaming is performed will not matter, provided that for each $T, \vec{x}, \vec{Q}$ as above we have a determinate choice of a suitable concrete term $T[\vec{x} \mapsto \vec{Q}]$, so that multiple appearances of the same substitution will always yield the same result. We also note that substitution is clearly well-defined on α-equivalence classes. Finally, 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 here is that a meta-term 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.

By the *local variable environment* associated with a concrete 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. (The no-variable-hiding convention ensures that X and indeed Γ, X is non-repetitive.) Although in principle local variable environments pertain to particular choices of concrete contexts, most of the concepts that we define using such environments will be easily seen to be invariant under renamings of bound variables.

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

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

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

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

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

Note that the β-rule applies even when $\vec{x}$ is empty: thus, $(\lambda.2)() \rightsquigarrow_b 2$.

From this, a *head reduction* relation $\rightsquigarrow_h$ on concrete 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, $\texttt{case}\ y\vec{Q}\ \texttt{of}\ (i \Rightarrow E_i)$ 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

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

(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. We emphasize that the relation $\rightsquigarrow$ is defined on concrete meta-terms, although it is clear that it also gives rise to a well-defined reduction relation on their α-classes. An important point to note is that modulo the obvious inclusion of terms into meta-terms, terms are precisely meta-terms in *normal form*, i.e. those that cannot be reduced using $\rightsquigarrow$. (For example, the meta-procedure $\lambda.2$ is in normal form, though the ground meta-term $(\lambda.2)()$ is not.) 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 concrete meta-procedures and on ground meta-terms: thus, $T \sqsubseteq U$ iff T may be obtained from U by replacing zero or more ground subterms (perhaps infinitely many) by $\bot$. It is easy to see that for each σ, the set of all concrete procedure terms of type σ forms a directed-complete partial order under $\sqsubseteq$.

By a *finite* (concrete) 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 ordinary NSP terms by identifying the conditional branching $(0 \Rightarrow e_0 \mid \cdots \mid r-1 \Rightarrow e_{r-1})$ with

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

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 concrete meta-term T we define

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

It is not hard to check that for any meta-term T, the set $\Downarrow_{\text{fin}} T$ is directed with respect to $\sqsubseteq$ (see [17, Proposition 6.1.2]). We may therefore define $\ll T \gg$, the *value* of T, to be the ordinary concrete term

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

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 $\ll T \gg = \ll T' \gg$.

Whilst we have defined our evaluation operation $\ll - \gg$ for concrete meta-terms, it is clear that this induces a well-defined evaluation operation on their α-classes, and for the most part this is all that we shall need. We also note that the syntactic ordering $\sqsubseteq$ on concrete terms induces a partial order $\sqsubseteq$ on their α-classes, and that each $\mathsf{SP}(\sigma)$ and $\mathsf{SP}^0(\sigma)$ is directed-complete with respect to this ordering.

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 of x 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 from T to T' 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 that the result of evaluating a meta-term is unaffected if we choose to evaluate certain subterms 'in advance':

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 logically elementary but administratively complex: see [17, 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). Using this, the restriction that variables may appear only at the head of applications can be seen to be inessential: e.g. the 'illegal term' fx may be replaced by the legal term fx^η. The definition of x^η 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 [17, Lemma 6.1.14]:

Lemma 5 $\ll x^\eta \vec{q} \gg = \mathtt{case}\ x\vec{q}\ \mathtt{of}\ (i \Rightarrow i)$, *and* $\ll \lambda \vec{y}.\, p\vec{y}^{\,\eta} \gg = p$.

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 of SP. It is easy to check that application in SP is monotone and continuous with respect to $\sqsubseteq$. It is also shown in [17, 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^Ω as defined in Subsection 2.3 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 specified corecursively up to α-equivalence by:

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

(A concrete representative of $F_\sigma[g]$ satisfying the no-hiding condition will of course feature a different choice of bound variables $x_0, \ldots, x_{r-1}$ at each level.) We may now give the standard interpretation of PCF^Ω in SP. To each PCF^Ω term $\Gamma \vdash M : \sigma$ we associate a procedure-in-environment $\Gamma \vdash [\![M]\!]^{\mathsf{SP}}_\Gamma : \sigma$ (denoted henceforth by $[\![M]\!]_\Gamma$) inductively as follows:

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

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

The following key property of $[\![-]\!]^{\mathsf{SP}}$ is shown as Theorem 7.1.16 in [17]:

Theorem 6 (Adequacy) *For any closed* PCF^Ω *term* $M : \mathtt{N}$*, we have* $M \rightsquigarrow^* \widehat{n}$ *iff* $[\![M]\!] = \lambda.n$.

We may now clarify the relationship between SP^0 and SF. First, there is a natural 'observational equivalence' relation $\approx$ on each $\mathsf{SP}^0(\sigma)$, defined by

$$q \approx q' \quad \text{iff} \quad \forall r \in \mathsf{SP}^0(\sigma \to \mathtt{N}).\ \ r \cdot q \ = \ r \cdot q' \ .$$

It is not hard to see that if $p \approx p' \in \mathsf{SP}^0(\sigma \to \tau)$ and $q \approx q' \in \mathsf{SP}^0(\sigma)$ then $p \cdot q \approx p' \cdot q \approx p' \cdot q' \in \mathsf{SP}^0(\tau)$. Explicitly, the first of these equivalences holds because for any $r \in \mathsf{SP}^0(\tau \to \mathtt{N})$ we have (using Lemma 5) that

$$r \cdot (p \cdot q) \ = \ (\lambda x.r(\lambda \vec{z}.xq\vec{z}^{\eta})) \cdot p \ = \ (\lambda x.r(\lambda \vec{z}.xq\vec{z}^{\eta})) \cdot p' \ = \ r \cdot (p' \cdot q) \ ,$$

while the second equivalence holds because for any $r \in \mathsf{SP}^0(\tau \to \mathtt{N})$ we have

$$r \cdot (p' \cdot q) \ = \ (\lambda y.r(\lambda \vec{z}.p'y^{\eta}\vec{z}^{\eta})) \cdot q \ = \ (\lambda y.r(\lambda \vec{z}.p'y^{\eta}\vec{z}^{\eta})) \cdot q' \ = \ r \cdot (p' \cdot q') \ .$$

We thus obtain a well-defined applicative structure $\mathsf{SP}^0/\!\approx$ as a quotient of SP^0; we write $\theta : \mathsf{SP}^0 \to \mathsf{SP}^0/\!\approx$ for the quotient map.

It turns out that up to isomorphism, this structure $\mathsf{SP}^0/\approx$ is none other than SF. Indeed, in [17] this was taken as the definition of SF, and the characterization as the closed term model of PCF$^{\Omega}$ modulo observational equivalence proved as a consequence. In order to fill out the picture a little more, we will here exhibit the equivalence of these two definitions as a consequence of the following non-trivial fact, given as Corollary 7.1.34 in [17]:[2]

Theorem 7 *For every $p \in \mathsf{SP}^0(\sigma)$, there is a closed* PCF$^{\Omega}$ *term $M : \sigma$ such that $[\![M]\!] \approx p$.*

Proposition 8 $(\mathsf{SP}^0/\approx) \cong \mathsf{SF}$, *via an isomorphism that identifies $\theta([\![M]\!]^{\mathsf{SP}})$ with $[\![M]\!]^{\mathsf{SF}}$ for any closed* PCF$^{\Omega}$ *term M.*

Proof For any element $x \in \mathsf{SP}^0/\!\approx$, we may take $p \in \mathsf{SP}^0$ with $\theta(p) = x$, then by Theorem 7 take M closed with $[\![M]\!] \approx p$; we then have that $\theta([\![M]\!]) = x$. In this way, each $\mathsf{SP}^0(\sigma)/\!\approx$ corresponds bijectively to the set of closed PCF$^{\Omega}$ terms $M : \sigma$ modulo some equivalence relation $\sim$.

Recall now that $\simeq$ denotes observational equivalence in PCF$^{\Omega}$. To see that $\sim \subseteq \simeq$, suppose $M \sim M'$, and let $C[-]$ be any suitable observing context of PCF$^{\Omega}$. By the compositionality of $[\![-]\!]^{\mathsf{SP}}$, we obtain some $[\![C]\!] \in \mathsf{SP}^0$ such that $[\![C[M]]\!] = [\![C]\!] \cdot [\![M]\!]$ and similarly for M'. But $M \sim M'$ means that $[\![M]\!] \approx [\![M']\!]$, whence by the definition of $\approx$ we conclude that $[\![C[M]]\!] = [\![C[M']]\!]$ at type $\mathtt{N}$. So by Theorem 6 we have $C[M] \rightsquigarrow^* \widehat{n}$ iff $C[M'] \rightsquigarrow^* \widehat{n}$. Since $C[-]$ was arbitrary, we have shown that $M \simeq M'$.

To see that $\simeq \subseteq \sim$, suppose $M \simeq M' : \sigma$. It will suffice to show that $[\![M]\!] \approx [\![M']\!]$. So suppose $r \in \mathsf{SP}^0(\sigma \to \mathtt{N})$, and using Theorem 7, take a PCF$^{\Omega}$ term R such that $[\![R]\!] \approx r$. Then $r \cdot [\![M]\!] = [\![R]\!] \cdot M = [\![RM]\!]$ at type $\mathtt{N}$, and similarly for M'. But since $M \simeq M'$, we have $RM \rightsquigarrow^* n$ iff $RM' \rightsquigarrow^* n$, whence

[2]What is actually shown in [17] is that every element of SP^0 is denotable on the nose in a language PCF + *byval*, with a certain choice of denotation for the constant *byval*. Since the latter satisfies $[\![\mathit{byval}]\!] \approx [\![\lambda f^{\mathtt{N}\to\mathtt{N}} x^{\mathtt{N}}.\ \mathit{ifzero}\ x\,(fx)(fx)]\!]$, the present Theorem 7 follows easily.

$[\![RM]\!] = [\![RM']\!]$ by Theorem 6. Thus $r \cdot [\![M]\!] = r \cdot [\![M']\!]$, and we have shown $[\![M]\!] \approx [\![M']\!]$.

Since each $\mathsf{SF}(\sigma)$ consists of closed PCF^Ω terms $M : \sigma$ modulo $\simeq$, we have established a bijection $(\mathsf{SP}^0(\sigma)/\approx) \cong \mathsf{SF}(\sigma)$ for each σ. Moreover, both $[\![-]\!]^{\mathsf{SP}}$ and θ respect application, so it follows that $(\mathsf{SP}^0/\approx) \cong \mathsf{SF}$, and it is immediate by construction that $\theta([\![M]\!]^{\mathsf{SP}})$ is identified with $[\![M]\!]^{\mathsf{SF}}$. $\square$

As we have already seen in Subsection 2.1, Milner's context lemma for PCF^Ω implies that SF is extensional. From this and Proposition 8, we may now read off the following useful characterization of the equivalence $\approx$:

Lemma 9 (NSP context lemma) *Suppose $p, p' \in \mathsf{SP}^0(\sigma_0, \ldots, \sigma_{r-1} \to \mathtt{N})$. Then $p \approx p'$ iff*

$$\forall q_0 \in \mathsf{SP}^0(\sigma_0), \ldots, q_{r-1} \in \mathsf{SP}^0(\sigma_{r-1}).\;\; p \cdot q_0 \cdot \ldots \cdot q_{r-1} \;=\; p' \cdot q_0 \cdot \ldots \cdot q_{r-1}\;.$$

We shall also make use of the *observational ordering* on SF and the associated preorder on SP. Let $\sqsubseteq$ be the usual information ordering on $\mathsf{SF}(\mathtt{N}) \cong \mathbb{N}_\bot$, and let us endow each $\mathsf{SF}(\sigma)$ with the partial order $\preceq$ defined by

$$x \preceq x' \quad \text{iff} \quad \forall h \in \mathsf{SF}(\sigma \to \mathtt{N}),\, n \in \mathbb{N}.\;\; h \cdot x \;\sqsubseteq\; h \cdot x'\;.$$

It is easy to see that the application operations $\cdot$ are monotone with respect to $\preceq$. Moreover, Milner's context lemma also exists in an *inequational* form which says, in effect, that if $f, f' \in \mathsf{SF}(\sigma_0, \ldots, \sigma_{r-1} \to \mathtt{N})$ then

$$f \preceq f' \quad \text{iff} \quad \forall y_0 \in \mathsf{SF}(\sigma_0), \ldots, y_{r-1} \in \mathsf{SF}(\sigma_{r-1}).\;\; f \cdot y_0 \cdot \ldots \cdot y_{r-1} \;\sqsubseteq\; f' \cdot y_0 \cdot \ldots \cdot y_{r-1}\;.$$

Thus, if elements of $\mathsf{SF}(\sigma \to \tau)$ are considered as functions $\mathsf{SF}(\sigma) \to \mathsf{SF}(\tau)$, the partial order $\preceq$ coincides with the pointwise partial order on functions.

We write $\preceq$ also for the preorder on each $\mathsf{SP}^0(\sigma)$ induced by $\preceq$ on SF: that is, $p \preceq p'$ iff $\theta(p) \preceq \theta(p')$. Furthermore, we extend the use of the notations $\approx$, $\preceq$ in a natural way to open terms (in the same environment), and indeed to meta-terms: e.g. we may write $\vec{x} \vdash P \preceq P'$ to mean $\ll \lambda\vec{x}.P \gg \preceq \ll \lambda\vec{x}.P' \gg$.

Clearly $p \approx p'$ iff $p \preceq p' \preceq p$. The following is also now immediate:

Lemma 10 *Suppose $p, p' \in \mathsf{SP}^0(\sigma)$ where $\sigma = \sigma_0, \ldots, \sigma_{r-1} \to \mathtt{N}$. Then the following are equivalent:*

(i) $p \preceq p'$.

(ii) $\forall r \in \mathsf{SP}^0(\sigma \to \mathtt{N}).\ r \cdot p \sqsubseteq r \cdot p'$.

(iii) $\forall q_0 \in \mathsf{SP}^0(\sigma_0), \ldots, q_{r-1} \in \mathsf{SP}^0(\sigma_{r-1}).\;\; p \cdot q_0 \cdot \ldots \cdot q_{r-1} \sqsubseteq p' \cdot q_0 \cdot \ldots \cdot q_{r-1}$.

We conclude this subsection by reformulating some of the major milestones in our proof using the notation now available. Specifically, in Sections 3 and 4 we will show the following:

Theorem 11 *For any $k \geq 1$, the element $[\![Y_{k+1}]\!] \in \mathsf{SP}^0$ is not PCF^Ω_k-denotable (whence neither is $[\![Y_{0\to(k+1)}]\!]$).*

In Section 5 we will go on to show that no $Z \approx [\![Y_{0\to(k+1)}]\!]$ can be $\mathrm{PCF}^{\Omega}_{k}$-denotable, establishing Theorem 2(i). In Section 6 we will resort to a more refined version of our methods to show the same for Y_{k+1}; this will establish Theorem 2(ii).

2.4 The embeddability hierarchy

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

Theorem 12 (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* $\le k$*. 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 $z \vdash r[\vec{x} \mapsto \vec{t}\,] \succeq z^{\eta}$*.*

If in the above setting we had $z \vdash r[\vec{x} \mapsto \vec{t}\,] \approx z^{\eta}$, 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 actually coincide, since $z \vdash p \succeq z^{\eta}$ implies $z \vdash p \approx z^{\eta}$. However, this fact will not be needed for the main results of this paper.

In our statement of Theorem 12, we have referred informally to a product $\Pi_i \sigma_i$ which we have not precisely defined (although the formal statement of the theorem 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 [17, Chapter 4]: for instance, it is not hard to show that any finite product of level $\le k$ types can be constructed as a retract of the pure type $\overline{k+1}$. In the present paper, however, references to product types may be taken to be purely informal and motivational.

The proof of Theorem 12 appears in [17, Section 7.7], but because of its crucial role in the paper we reprise it here with some minor stylistic improvements.

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 may note that $\mathtt{N} \to \mathtt{N}$ is not a *retract* of $\mathtt{N}^r$, since the former contains strictly ascending chains of length $r+2$ 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 $z \vdash t_i$ and $\vec{x} \vdash r$ exhibit $\overline{k+1}$ as a pseudo-retract of $\Pi_i \sigma_i$ where each σ_i is of level $\le k$. Let $v = \ll r[\vec{x} \mapsto \vec{t}\,] \gg$, so that $z \vdash v \succeq z^{\eta}$, whence $\ll v[z \mapsto u] \gg \succeq u$ for any $u \in \mathsf{SP}^0(k+1)$. 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

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

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

We now focus on the subterm p in $v = \lambda f^k.\,\texttt{case}\ zp\ \texttt{of}\ (\cdots)$. 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 construction of v as $\ll r[\vec{x} \mapsto \vec{t}] \gg$ can be used to split this into functions $\overline{k} \to \Pi_j \rho_j$ 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 here (somewhat arbitrarily) to $\lambda w.0$.

Specifically, we claim that $\lambda f. \ll p[z \mapsto \lambda w.0] \gg \succeq id_k$. By Lemma 10, it will suffice to show that $\ll p[f \mapsto q, z \mapsto \lambda w.0] \gg \succeq q$ for any $q \in \mathsf{SP}^0(k)$. The idea is that if it is not, then (ignoring the presence of z in p for now) we may specialize z to some u that will detect the difference between $p[f \mapsto q]$ and q, so that the subterm 'zp' within v will yield $\bot$, contradicting that $z \vdash v \succeq z^\eta$. We can even allow for the presence of z in p by a suitably careful choice of u.

Again by Lemma 10, it suffices to show that $\ll p[f \mapsto q, z \mapsto \lambda w.0] \gg \cdot\, s \succeq q \cdot s$ for any s. So suppose $q \cdot s = \lambda.n$ whereas $\ll p[f \mapsto q, z \mapsto \lambda w.0] \gg \cdot\, s \neq \lambda.n$ for some $s \in \mathsf{SP}^0(k-1)$ and $n \in \mathbb{N}$. Take $u = \lambda g.\,\texttt{case}\ gs\ \texttt{of}\ (n \Rightarrow 0)$, so that $u \cdot q' = \bot$ whenever $q' \cdot s \neq \lambda.n$. Then $u \preceq \lambda w.0$ by Lemma 10, so we have $\ll p[f \mapsto q, z \mapsto u] \gg \cdot\, s \neq \lambda.n$ since $\lambda.n$ is maximal in $\mathsf{SP}^0(\mathtt{N})$. By the definition of u, it follows that $\ll (zp)[f \mapsto q, z \mapsto u] \gg = \bot$, whence $\ll v[z \mapsto u] \gg \cdot\, q = \bot$, whereas $u \cdot q = 0$, contradicting $\ll v[z \mapsto u] \gg \succeq u$. This completes the proof that $\lambda f. \ll p[z \mapsto \lambda w.0] \gg \succeq id_k$.

Next, we show how to split the function represented by this procedure so as to go through some $\Pi_j \rho_j$ as above. Since $\ll r[\vec{x} \mapsto \vec{t}] \gg = \lambda f.\,\texttt{case}\ zp\ \texttt{of}\ (\cdots)$, we have that $r[\vec{x} \mapsto \vec{t}]$ reduces in finitely many steps to a head normal form $\lambda f.\,\texttt{case}\ zP\ \texttt{of}\ (\cdots)$ 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 z does not appear free in r, this occurs within some t_i, and clearly it must appear as the head of some subterm $\texttt{case}\ zp'\ \texttt{of}\ (\cdots)$. 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 $x'_0, x'_1, \ldots$ denote the finitely many bound variables that are in scope at the relevant occurrence of zp', and suppose each x'_j has type ρ_j of level $< k$. By considering the form of the head reduction sequence $r[\vec{x} \mapsto \vec{t}] \rightsquigarrow_h^* \lambda f.\,\texttt{case}\ zP\ \texttt{of}\ (\cdots)$, we now see that P has the form $p'[\vec{x}\,' \mapsto \vec{T}]$ where each $T_j : \rho_j$ contains at most f and z free.[3]

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

$$f^k \vdash t'_j = \ll T_j^* \gg : \rho_j\ , \qquad \vec{x}\,' \vdash r' = \ll p'^* \gg : \overline{k}\ .$$

Then $\ll r'[\vec{x}\,' \mapsto \vec{t}\,'] \gg$ 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

[3]The reader wishing to see a more formal justification for this step may consult the proof of Lemma 26(i) below.

$\Pi_j \rho_j$, 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. $\Box$

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 [23], 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 $\mathrm{PCF}_1 + por + exists$. (This latter fact was already noted in [23].)

- In PCF + *catch* (a slight strengthening of Cartwright and Felleisen's language SPCF [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 [14], 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 [17]. These facts may offer some indication of why a 'cheap' proof of our present results in the setting of pure PCF is not to be expected.[4]

2.5 Other sublanguages of PCF

Our main theorems establish a hierarchy of languages $\mathrm{PCF}_1 < \mathrm{PCF}_2 < \cdots$. Before proceeding further, however, we pause to clarify the relationship between PCF_0 and PCF_1, and also to survey some of the interesting territory that lies between them, in order to situate our theorems within a wider picture.

On the one hand, PCF_0 is a rather uninteresting language. As regards the elements of SF that it denotes, it is equivalent in expressivity to $\mathrm{PCF}_\perp$, a variant of PCF_0 in which we replace Y_0 by a constant $\perp$ (denoting $\perp \in \mathsf{SF}(\mathtt{N})$). This is clear since Y_0 and $\perp$ are interdefinable: we have $\perp = Y_0(\lambda x.x)$, and it is easy to see that $Y_0 = \lambda f.f\perp$ (in SF). By a syntactic analysis of the possible normal forms of type $\overline{1}$ in $\mathrm{PCF}_\perp$, one can show that these are very weak languages that do not even define addition.

However, such an analysis is unnecessary for our purposes, since there are more interesting languages that clearly subsume PCF_0 but are known to be weaker than PCF_1. For instance, Berger [4] considered the language $\mathrm{T}_0 + min$,

[4]That PCF manifests greater structural complexity than many stronger languages is also a moral of Loader's undecidability theorem for finitary PCF [13]. However, the complexity we explore here seems quite orthogonal to that exhibited by Loader: we are concerned purely with 'infinitary' aspects of definability, the entire finitary structure being already represented by our PCF_0.

where T_0 (a fragment of Gödel's System T) is the λ-calculus with first-order primitive recursion over the natural numbers:

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

and *min* is the classical *minimization* (i.e. unbounded search) operator of type $(\mathtt{N} \to \mathtt{N}) \to \mathtt{N}$. On the one hand, it is an easy exercise to define $\perp$ in $\mathrm{T}_0 + \mathit{min}$, and to define both rec_0 and *min* in PCF_1. On the other hand, Berger showed that the PCF_1-definable functional $\Phi_0 : (\mathtt{N} \to \mathtt{N} \to \mathtt{N}) \to (\mathtt{N} \to \mathtt{N})$ given by

$$\Phi_0 \; g \; n \;\; = \;\; g \; n \; (\Phi_0 \; g \; (n+1)) \ ,$$

is not expressible in $\mathrm{T}_0 + \mathit{min}$.[5] As already indicated in Section 1, this functional and its higher-type analogues will play a crucial role in the present paper.

This situation is revisited in [17, Section 6.3] from the perspective of substructures of SP^0. It is shown that $\mathrm{T}_0 + \mathit{min}$, and indeed the whole of $\mathrm{T} + \mathit{min}$, can be modelled within the substructure $\mathsf{SP}^{0,\mathrm{lwf}}$ of *left-well-founded* procedures, whereas the above functional Φ_0 is not representable by any such procedure; thus Φ_0 is not expressible in $\mathrm{T} + \mathit{min}$. (The reader may wish to study these results and proofs before proceeding further, since they provide simpler instances of the basic method that we will use in this paper.) At third order, there are even 'hereditarily total' functionals definable in PCF_1 but not by higher-type iterators, one example being the well-known *bar recursion* operator (see [16]).

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 [17]; this again subsumes PCF_0. It is shown in [17] that the computational power of $\mathsf{Klex}^{\mathrm{min}}$ coincides with that of computable *left-bounded* procedures; this is used to show, for example, that even rec_0 is not computable in $\mathsf{Klex}^{\mathrm{min}}$. We find it reasonable to regard left-bounded procedures as embodying the weakest higher-order computability notion of natural interest that is still Turing complete.

3 Sequential procedures for PCF_k terms

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

In this section we give a direct inductive characterization of the PCF_k^Ω-denotable elements of SP by making explicit how our interpretation works for terms of PCF_k^Ω. The first point to observe is that we may restrict attention to PCF_k^Ω terms in *long $\beta\eta$-normal form*: that is, terms in β-normal form in which every variable or constant z of type $\sigma_0, \ldots, \sigma_{r-1} \to \mathtt{N}$ is fully applied (i.e. appears at the head of a subterm $zN_0 \ldots N_{r-1}$ of type $\mathtt{N}$). Moreover, an inductive characterization of the class of such terms is easily given.

[5] Berger actually considered denotability in the Scott model, but his argument applies equally to SF.

Proposition 13 *(i) A procedure $\Gamma \vdash p : \sigma$ is denotable by a PCF_k^Ω term $\Gamma \vdash M : \sigma$ iff it is denotable by one in long $\beta\eta$-normal form.*

(ii) The class of long $\beta\eta$-normal forms of PCF_k^Ω is inductively generated by the following clauses:

1. *If $\Gamma \vdash N_i : \sigma_i$ is a normal form for each $i < r$ and $x^{\sigma_0,\ldots,\sigma_{r-1}\to\mathtt{N}} \in \Gamma$, then $\Gamma \vdash xN_0 \ldots N_{r-1} : \mathtt{N}$ is a normal form (note that r may be 0 here).*

2. *If $\Gamma, x^\sigma \vdash M : \tau$ is a normal form then so is $\Gamma \vdash \lambda x.M : \sigma \to \tau$.*

3. *The numeric literals $\Gamma \vdash \widehat{n} : \mathtt{N}$ are normal forms.*

4. *If $\Gamma \vdash M : \mathtt{N}$ is a normal form then so are $\Gamma \vdash suc\ M : \mathtt{N}$, $\Gamma \vdash pre\ M : \mathtt{N}$ and $\Gamma \vdash C_f\, M : \mathtt{N}$ for any $f : \mathbb{N} \rightharpoonup \mathbb{N}$.*

5. *If $\Gamma \vdash M : \mathtt{N}$, $\Gamma \vdash N : \mathtt{N}$ and $\Gamma \vdash P : \mathtt{N}$ are normal forms, then so is $\Gamma \vdash ifzero\ M\ N\ P : \mathtt{N}$.*

6. *If $\sigma = \sigma_0, \ldots, \sigma_{r-1} \to \mathtt{N}$ is of level $\leq k$ and $\Gamma \vdash M : \sigma \to \sigma$ and $\Gamma \vdash N_i : \sigma_i$ are normal forms, then $\Gamma \vdash Y_\sigma M N_0 \ldots N_{r-1} : \mathtt{N}$ is a normal form.*

Proof (i) It is a well-known property of simply typed λ-calculi that every term M is $\beta\eta$-equivalent to one in long $\beta\eta$-normal form: indeed, we may first compute the β-normal form of M and then repeatedly apply the η-rule to expand any subterms that are not already fully applied. Moreover, it is shown in [17, Theorem 6.1.18] that SP is a $\lambda\eta$-algebra, so that if $\Gamma \vdash M =_{\beta\eta} M'$ then $[\![M]\!]_\Gamma = [\![M']\!]_\Gamma$ in SP. This establishes the claim.

(ii) This is clear from the fact that no application may be headed by a λ-abstraction and that all occurrences of variables and constants must be fully applied. $\Box$

It follows that the class of PCF_k^Ω-denotable procedures may be generated inductively by a set of clauses that mirror the above formation rules for long $\beta\eta$-normal PCF_k^Ω terms. We now consider each of these formation rules in turn in order to spell out the corresponding operation at the level of NSPs. In Section 4 we will show that these operations cannot give rise to $k+1$-*spinal* procedures, from which it will follow that no PCF_k^Ω-denotable procedure can be $k+1$-spinal.

For the first three formation rules, the effect on NSPs is easily described:

Proposition 14 *(i) If $\Gamma \vdash xN_0 \ldots N_{r-1} : \mathtt{N}$ in PCF^Ω, then $[\![xN_0 \ldots N_{r-1}]\!]_\Gamma = \mathtt{case}\ x[\![N_0]\!]_\Gamma \cdots [\![N_{r-1}]\!]_\Gamma\ \mathtt{of}\ (j \Rightarrow j)$.*

(ii) If $\Gamma \vdash \lambda x.M : \sigma \to \tau$ in PCF^Ω, then $[\![\lambda x.M]\!]_\Gamma = \lambda x.\, [\![M]\!]_{\Gamma,x}$.

(iii) $[\![\widehat{n}]\!]_\Gamma = \lambda.n$.

Proof (i) Easy using the definition of $[\![-]\!]$ and Lemma 5.

(ii), (iii) are part of the definition of $[\![-]\!]$. $\Box$

As regards the formation rules for *suc*, *pre*, C_f and *ifzero*, the situation is again fairly straightforward, although a little more machinery is needed:

Definition 15 *(i) The set of* rightward *(occurrences of)* numeral leaves *within a term t is defined inductively by means of the following clauses:*

1. *A 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* $\texttt{case}\ a\ \texttt{of}\ (i \Rightarrow e_i)$.

(ii) If t is a term and e_i an expression for each i, let $t[i \mapsto e_i]$ denote the result of replacing each rightward leaf occurrence i in t by the corresponding e_i.

Lemma 16 $\ll \texttt{case}\ d\ \texttt{of}\ (i \Rightarrow e_i) \gg\ = d[i \mapsto e_i]$ *for any expressions d, e_i.*

Proof For each $c \in \mathbb{N}$, define a 'truncation' operation $-^{(c)}$ on expressions as follows:

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

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 $d = \texttt{case}\ a\ \texttt{of}\ (j \Rightarrow f_j)$. Then

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

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

From this lemma we may now read off the operations on NSPs that correspond to clauses 4 and 5 of Proposition 13(ii):

Proposition 17 *(i) If $\Gamma \vdash M : \mathtt{N}$ in* PCF$^\Omega$*, then $[\![C_f\ M]\!]_\Gamma = [\![M]\!]_\Gamma[i \mapsto f(i)]$ (understanding $f(i)$ to be $\bot$ when $i \notin \mathrm{dom}\ f$); similarly for suc and pre.*

(ii) If $\Gamma \vdash M : \mathtt{N}$, $\Gamma \vdash N : \mathtt{N}$ and $\Gamma \vdash P : \mathtt{N}$, then $[\![ifzero\ M\ N\ P]\!]_\Gamma = [\![M]\!]_\Gamma[0 \mapsto d,\ i+1 \mapsto e]$ where $[\![N]\!]_\Gamma = \lambda.d$ and $[\![P]\!]_\Gamma = \lambda.e$.

Proof (i) The definition of $[\![-]\!]$ yields

$$[\![C_f\, M]\!]_\Gamma \;=\; \ll \lambda.\, \texttt{case}\ [\![M]\!]_\Gamma\ \texttt{of}\ (i \Rightarrow f(i)) \gg ,$$

and by Lemma 16 this evaluates to $[\![M]\!]_\Gamma[i \mapsto f(i)]$. Likewise for *suc* and *pre*.

(ii) The definition of $[\![-]\!]$ yields

$$[\![\mathit{ifzero}\ M\ N\ P]\!]_\Gamma \;=\; \ll \lambda.\, \texttt{case}\ [\![M]\!]_\Gamma\ \texttt{of}\ (0 \Rightarrow d \mid i+1 \Rightarrow e) \gg ,$$

and by Lemma 16 this evaluates to $[\![M]\!]_\Gamma[0 \mapsto d,\ i+1 \mapsto e]$. $\Box$

It remains to consider the formation rule involving Y_σ. It will be convenient to regard the NSP for $YMN_0 \ldots N_{r-1}$ as a result of plugging some simpler NSPs together, in the sense indicated by the following definition. Here and later, we shall follow the convention that Greek capitals Γ, Δ denote arbitrary environments, while Roman capitals Z, X, V denote lists of variables of type level $\leq k$. (Of course, the idea of plugging can be formulated without any restrictions on types, but we wish to emphasize at the outset that only pluggings at level $\leq k$ will feature in our proof.)

Definition 18 (Plugging) *Suppose given the following data:*

- *a variable environment* Γ,
- *a finite list* Z *of 'plugging variables'* z *of level* $\leq k$, *disjoint from* Γ,
- *a* root expression $\Gamma, Z \vdash e$,
- *a substitution* ξ *assigning to each* $z^\sigma \in Z$ *a procedure* $\Gamma, Z \vdash \xi(z) : \sigma$.

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* $\xi(z)$*. To formalize this, let* T° *denote the meta-term obtained from* T *by replacing each ground type subterm* $z\vec{Q}$ *(where* $z \in Z$*) by* $\bot$*. We may now define, up to* α*-equivalence,*

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

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. Note that some renaming of bound variables will typically be necessary in order to realize $\Pi_{\Gamma,Z}(e,\xi)$ as a concrete term conforming to the no-variable-hiding condition; we will not need to fix on any one particular way of doing this.

The operation on NSPs corresponding to clause 6 of Proposition 13(ii) may now be described as follows:

Proposition 19 *Suppose that $\sigma = \sigma_0, \ldots, \sigma_{r-1} \to \mathtt{N}$ is of level $\leq k$ and that $\Gamma \vdash Y_\sigma M N_0 \ldots N_{r-1}$ in $\mathrm{PCF}^{\Omega}_{k}$, where $[\![M]\!]_\Gamma = \lambda z^\sigma.p = \lambda z^\sigma x_0^{\sigma_0} \cdots x_{r-1}^{\sigma_{r-1}}.e$ and $[\![N_i]\!]_\Gamma = q_i$ for each i. Then*

$$[\![Y_\sigma M N_0 \ldots N_{r-1}]\!]_\Gamma \;=\; \lambda. \ll \Pi_{\Gamma,Z}(e,\xi) \gg$$

where $Z = z, x_0, \ldots, x_{r-1}$, $\xi(z) = p$, and $\xi(x_i) = q_i$ for each i.

Proof Note that in this instance of plugging, the repeated substitutions are needed only for the sake of the term $\xi(z)$ which may contain z free—only a single substitution step is needed for the plugging variables x_i, since the q_i contain no free variables from Z. We may thus rewrite $\Pi_{\Gamma,Z}(e,\xi)$ as

$$\Pi_{\Gamma,\vec{x},Z'}(e,\xi')\;[\vec{x} \mapsto q]\,,$$

where $Z' = \{z\}$ and $\xi'(z) = p$. The proposition will therefore follow easily (with the help of Theorem 4) once we know that

$$[\![Y_\sigma M]\!]_\Gamma \;=\; \lambda\vec{x}.\ll \Pi_{\Gamma,\vec{x},Z'}(e,\xi') \gg\;.$$

To see this, write $Y_\sigma = \lambda g.F_\sigma[g]$ where $F_\sigma[g] = \lambda\vec{x}.\,\mathtt{case}\;g\,(F_\sigma[g])\,\vec{x}^{\,\eta}\;\mathtt{of}\;(i \Rightarrow i)$ as at the start of Section 2.3. Then clearly

$$[\![Y_\sigma M]\!]_\Gamma \;=\; (\lambda g.F_\sigma[g]) \cdot (\lambda z.p) \;=\; \ll F_\sigma[\lambda z.p] \gg\;.$$

Here the meta-term $F_\sigma[\lambda z.p]$ is specified corecursively (up to α-equivalence) by

$$F_\sigma[\lambda z.p] \;=\; \lambda\vec{x}.\,\mathtt{case}\;(\lambda z.p)\,(F_\sigma[\lambda z.p])\,\vec{x}^{\,\eta}\;\mathtt{of}\;(i \Rightarrow i)$$

That is, $F_\sigma[\lambda z.p]$ coincides with the meta-term $G = \bigsqcup_m G^m$, where

$$G^0 \;=\; \bot_\sigma\,, \qquad G^{m+1} \;=\; \lambda\vec{x}.\,\mathtt{case}\;(\lambda z.p)\,G^m\,\vec{x}^{\,\eta}\;\mathtt{of}\;(i \Rightarrow i)\,.$$

We may now compare this with the meta-term $H = \bigsqcup_m H^m$, where

$$H^0 \;=\; \bot_\sigma\,, \qquad H^{m+1} \;=\; \lambda\vec{x}.\,e[z \mapsto H^m]\,.$$

Noting that $(\lambda z.p)\,G^m\,\vec{x}^{\,\eta} \rightsquigarrow e[z \mapsto G^m,\; \vec{x} \mapsto \vec{x}^{\,\eta}]$, we have by Lemmas 5 and 16 that

$$\ll G^{m+1} \gg \;=\; \ll \lambda\vec{x}.\,e[z \mapsto G^m] \gg$$

whence by Theorem 4 and an easy induction we have $\ll G^m \gg = \ll H^m \gg$ for all m. Hence $\ll G \gg = \ll H \gg$.

Moreover, it is immediate from the definition that H coincides with the meta-term $\lambda\vec{x}.\,\Pi_{\Gamma,\vec{x},Z'}(e,\xi')$ mentioned earlier. We thus have

$$[\![Y_\sigma M]\!]_\Gamma \;=\; \ll F_\sigma[\lambda z.p] \gg \;=\; \ll G \gg \;=\; \ll H \gg \;=\; \lambda\vec{x}.\ll \Pi_{\Gamma,\vec{x},Z'}(e,\xi') \gg$$

and the proof is complete. (We have glossed over some fine details of variable renaming here, but these are easily attended to.) $\Box$

Combining Propositions 14, 17 and 19 with Proposition 13, the results of this section may be summarized as follows.

Theorem 20 *The class of* PCF_k^Ω*-denotable procedures-in-environment* $\Gamma \vdash p$ *is the class generated inductively by the following rules:*

1. *If* $\Gamma \vdash q_i$ *is denotable for each* $i < r$ *and* $x \in \Gamma$*, then*
$$\Gamma \vdash \lambda.\,\mathtt{case}\ x q_0 \ldots q_{r-1}\ \mathtt{of}\ (j \Rightarrow j)$$
is denotable.

2. *If* $\Gamma, x \vdash p$ *is denotable, then* $\Gamma \vdash \lambda x.p$ *is denotable.*

3. *Each* $\Gamma \vdash \lambda.n$ *is denotable.*

4. *If* $\Gamma \vdash p$ *is denotable and* $f : \mathbb{N} \rightharpoonup \mathbb{N}$*, then* $\Gamma \vdash p[i \mapsto f(i)]$ *is denotable. (The constructions for suc and pre are special cases of this).*

5. *If* $\Gamma \vdash p$*,* $\Gamma \vdash \lambda.d$ *and* $\Gamma \vdash \lambda.e$ *are denotable, then* $\Gamma \vdash p[0 \mapsto d, i+1 \mapsto e]$ *is denotable.*

6. *If* $\Gamma \vdash \lambda z^\sigma x_0^{\sigma_0} \cdots x_{r-1}^{\sigma_{r-1}}.e$ *is denotable where* $\sigma = \sigma_0, \ldots, \sigma_{r-1} \rightarrow \mathtt{N}$ *is of level* $\leq k$*, and* $\Gamma \vdash q_i : \sigma_i$ *is denotable for each* $i < r$*, then*
$$\Gamma \vdash \lambda. \ll \Pi_{\Gamma,Z}(e,\xi) \gg$$
is denotable, where $Z = z, \vec{x}$ *is disjoint from* Γ*,* $\xi(z) = \lambda\vec{x}.e$*, and* $\xi(x_i) = q_i$ *for each* i*.*

To conclude this section, we introduce a useful constraint on NSPs which, although not satisfied by all PCF_k^Ω-denotable procedures, will hold for all those that we will need to consider in the course of our main proofs. As we shall see, this constraint will interact well with the inductive rules just presented.

Referring back to the examples in Section 1, we see that the recursive definitions of both Y_{k+1} and Φ_{k+1} involved a variable g of type level $k+2$. It is therefore natural that our analysis will involve the consideration of terms in which such a variable g appears free. However, it will turn out that apart from this one designated variable, our terms need never involve any other variables of level $> k$, and this has a pleasant simplifying effect on our arguments. This motivates the following definition:

Definition 21 *Suppose* g *is a variable of type level* $k+2$.

(i) An environment Γ *is* (g-)regular *if* Γ *contains* g *but all other variables in* Γ *are of type level* $\leq k$.

(ii) A meta-term T *is regular if all free and bound variables within* T *are of level* $\leq k$*, except possibly for free occurrences of* g.

(iii) A meta-term-in-environment $\Gamma \vdash T$ *is regular if both* Γ, T *are regular.*

There is a useful alternative characterization of regularity in the case of normal forms:

Proposition 22 *A term-in-environment $\Gamma \vdash t$ is regular iff Γ is regular and t is not a procedure of type level $\geq k+2$.*

Proof The left-to-right implication is trivial, since a procedure of level $\geq k+2$ would have the form $\lambda\vec{x}.\cdots$ where at least one of the x_i was of level $\geq k+1$. For the converse, suppose Γ is regular and t is not a procedure of level $\geq k+2$. Then t contains no free variables of level $\geq k$ other than g, so we just need to show that all variables bound by a λ-abstraction within t are of level $\leq k$. Suppose not, and suppose that $\lambda\vec{x}.e$ 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. $\Box$

Let us now consider how the inductive clauses of Theorem 20 may be used to generate *regular* procedures-in-environment $\Gamma \vdash t$. The following gives a useful property of derivations involving these clauses:

Proposition 23 *If $\Gamma \vdash t$ is regular and $\mathrm{PCF}^{\Omega}_{k}$-denotable, then any inductive generation of the denotability of $\Gamma \vdash t$ via the clauses of Theorem 20 will consist entirely of regular procedures-in-environment.*

Proof It suffices to observe that for each of the six inductive clauses (regarded as rules), if the conclusion is a regular procedure-in-environment then so are each of the premises. For clause 1, this is clearly the case because the q_i are subterms of the procedure in the conclusion. For clause 2, we note that if $\lambda x.p$ is regular then so is p, and moreover x has level $\leq k$ so that Γ, x is regular. Clauses 3 and 4 are trivially handled. For clause 5, we not that if $\Gamma \vdash p[0 \vdash d, i+1 \mapsto e]$ is regular then $\Gamma \vdash p$, $\Gamma \vdash \lambda.d$ and $\Gamma \vdash \lambda.e$ are immediately regular by Proposition 22 (regardless of whether any leaves 0 or $i+1$ appear in p). Likewise, for clause 6, we note that under the given hypotheses, both $\lambda z\vec{x}.e$ and each q_i are of level $\leq k+1$; hence if Γ is regular then immediately $\Gamma \vdash \lambda z\vec{x}.e$ and $\Gamma \vdash q_i$ are regular by Proposition 22. $\Box$

In particular, let us consider again the construction of the procedure Y_{k+1} as $\lambda g.F_{k+1}[g]$, where

$$F_{k+1}[g] \;=\; \lambda x^k.\, \mathtt{case}\; g\,(F_{k+1}[g])\, x^{\eta}\; \mathtt{of}\; (i \Rightarrow i)\;.$$

It is clear by inspection that $g \vdash F_{k+1}[g]$ is regular; hence, if it were $\mathrm{PCF}^{\Omega}_{k}$-denotable, then Proposition 23 would apply. We shall show, however, that a purely regular derivation via the clauses of Theorem 20 cannot generate 'spinal' terms such as $F_{k+1}[g]$; hence $g \vdash F_{k+1}[g]$ is not $\mathrm{PCF}^{\Omega}_{k}$-denotable. This will immediately imply that Y_{k+1} itself is not $\mathrm{PCF}^{\Omega}_{k}$-denotable (Theorem 11), since the only means of generating non-nullary λ-abstractions is via clause 2 of Theorem 20.

4 PCF_k^Ω-denotable procedures are non-spinal

In this section, we will introduce the crucial notion of a *(k+1-)spinal term*, and will show that the clauses of Theorem 20 (in the regular case) are unable to generate spinal terms from non-spinal ones. Since the procedure $F_{k+1}[g]$ will be easily seen to be spinal, this will establish Theorem 11.

More specifically, we will actually introduce the notion of a *g-spinal term*, where g is a free variable which we treat as fixed throughout our discussion. We shall do this first for the case

$$g \; : \; (k+1) \rightarrow (k+1)$$

as appropriate for the analysis of $F_{k+1}[g]$ and hence of Y_{k+1}. Later we will also consider a minor variation for g of type $\mathtt{N} \rightarrow (k+1) \rightarrow (k+1)$, as appropriate to the definition of Φ_{k+1} given in Section 1. In both cases, we shall be able to dispense with the term 'k+1-spinal', since the type level $k+1$ may be read off from the type of g.

Some initial intuition for the concept of spinality was given in Section 1. We now attempt to provide some further motivation by examining a little more closely the crucial difference between Y_{k+1} and Y_k (say) that we are trying to capture.

The most obvious difference between these procedures is that Y_{k+1} involves an infinite sequence of nested calls to a variable $g : (k+1) \rightarrow (k+1)$, 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 using the means at our disposal corresponding to PCF_k^Ω terms.

As it stands, however, this is not the case. Suppose, for example, that $up_k : k \rightarrow k+1$ and $down_k : k+1 \rightarrow k$ are PCF_0 terms defining a standard retraction $k \lhd k+1$. More 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

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

This is essentially just a representation of Y_k modulo our encoding of type k in type $k+1$. 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 $g : (k+1) \rightarrow (k+1)$. (These NSPs are shown schematically in Figure 1 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 $k+1 \rightarrow k+1$ is not exploited, since both the input and output of g are 'funnelled' through the simpler type k. Such funnelling will inevitably entail some loss of information, as Theorem 12 tells us that the type k cannot

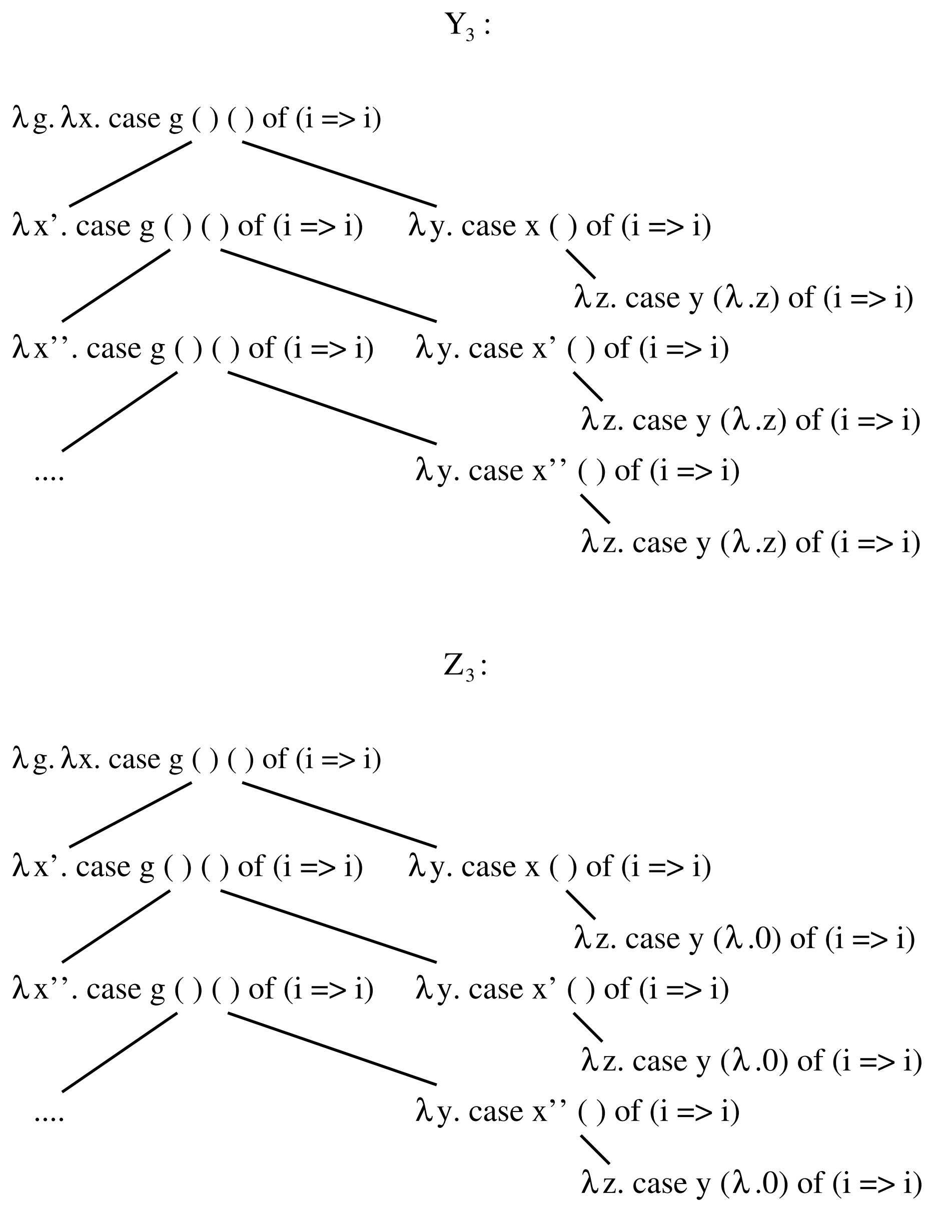
Y3 :
λg. λx. case g ( ) ( ) of (i => i)
λx'. case g ( ) ( ) of (i => i)
λy. case x ( ) of (i => i)
λz. case y (λ.z) of (i => i)
λx''. case g ( ) ( ) of (i => i)
λy. case x' ( ) of (i => i)
λz. case y (λ.z) of (i => i)
....
λy. case x'' ( ) of (i => i)
λz. case y (λ.z) of (i => i)
Z3 :
λg. λx. case g ( ) ( ) 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 $\lambda.\,\texttt{case}\ z\ \texttt{of}\ (i \Rightarrow i)$.

fully represent the structure of the type $k+1$. A useful mental picture here is that of a $(k+1)$-dimensional space being 'flattened' down to a k-dimensional one. Broadly speaking, then, we shall want to define a g-spinal term to be one containing an infinite sequence of nested calls to g but with no essential 'flattening' of the arguments. It will then be the case that Y_{k+1} is g-spinal, but Z_{k+1} is not.

We now approach the formal definition of a g-spinal term, generalizing the structure exhibited by the terms $F_{k+1}[g]$. To get our bearings, let us examine the form of these terms one more time. Note that $F_{k+1}[g]$ has the form $\lambda x^k.H[g,x]$, where

$$H[g,x] \;=\; \texttt{case}\; g(\lambda x'.H[g,x'])x^\eta \;\texttt{of}\; (\cdots)\;.$$

Spinal expressions of this kind, in which the topmost g of the spine appears at the very head of the expression, will be referred to as *head-spinal*. In fact, we shall say that $H[g,x]$ is head-spinal with respect to the variable x, since as noted above, it is significant here that x is passed to g with no 'flattening' (in the form of the procedure x^η). As a first attempt, then, one might hazard that we should define a concept of head-spinality relative to a type k variable coinductively as follows: an expression e is g-head-spinal w.r.t. x if it is of the form

$$\texttt{case}\; g(\lambda x'.e')x^\eta \;\texttt{of}\; (\cdots)$$

where e' is itself g-head-spinal w.r.t. x.

In fact, in order for the set of non-spinal terms to have appropriate closure properties, we shall need to relax this definition in two ways. Firstly, we allow $\lambda x'.e'$ to be replaced by $\lambda x'.E[e']$ where $E[-]$ is any expression context: that is, we allow the head-spinal subterm e' to appear at positions other than the head of this procedure. Secondly, we allow x^η to be replaced by a procedure term o that can be *specialized* to (something close to) x^η: the intuition is that any such o will embody the whole content of x with no flattening.

This leads us, at last, to the following definition. Note that this makes reference to the technical notion of an *x,V-closed substitution*, the explanation of which we shall defer to Definition 25 below. This entails that the notion of head-spinality needs to be defined relative to a certain set V of variables as well as a type k variable x. We shall adopt the convention that any environment denoted by Γ will be g-regular (and hence will contain g); 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).

Definition 24 (Spinal terms) *Suppose g has type $(k+1) \to (k+1)$. Suppose $\Gamma \vdash e$ is g-regular, and that $x^k \in \Gamma$ and $V \subseteq \Gamma$.*

(i) In this situation, we coinductively declare e to be g-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 an expression context, and

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

2. *e' is g-head-spinal with respect to x', V', where V' is the local variable environment for $E[-]$. (Clearly e' will automatically be g-regular in some Γ' that contains both x' and V'.)*

In other words, we take 'e is g-head-spinal w.r.t. x, V' to be the largest relation that satisfies the above statement.

(ii) In the above setting, we may also refer to the application $g(\lambda x'.E[e'])o$ itself as g-head-spinal w.r.t. x, V.

*(iii) We say a term t is g-*spinal *if it contains a subexpression that is g-head-spinal w.r.t. some x, V.*

Whilst this definition makes use of local variable environments which in principle pertain to concrete terms, it is easily seen that the notion of g-spinal term is α-invariant. Since we are taking g to be fixed throughout the discussion, we will usually omit mention of it and speak simply of spinal and head-spinal terms, and of regular (meta-)terms and environments.

In condition 1, one might have expected to see $o^\circ \approx x^\eta$, 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 somewhat non-trivial to show and is not needed for our main proof.

It remains to define the notion of an x, V-closed substitution. Suppose that $^\circ = [\vec{w} \mapsto \vec{r}]$ is some substitution proposed for use in condition 1 of Definition 24(i). Since we are wishing to compare o° with x^η, it is natural to require that the $\vec{r}$ contain no free variables other than x. However, what we want to ensure here is intuitively that the whole unflattened content of x is present in o itself rather than simply being introduced by the substitution. This can be ensured if we allow x as a free variable only in procedures r_i of type level $< k$: such procedures can only introduce 'flattened' images of x, since the x is here being funnelled through a type of level $\leq k-1$.

For technical reasons, we furthermore need to restrict such occurrences of x to those r_i substituted for variables w_i in a certain set V, which in practice will consist of variables bound between one spinal occurrence of g and the next (as can be seen from the specification of V' above). The necessity for the set V is admittedly difficult to motivate at this point: it is simply what the details of the proof seem to demand (see the last page of the proof of Lemma 27).

Definition 25 *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.*

It is worth remarking that if we were only interested in showing the non-definability of Y_{k+1} as an element of SP^0, one could do without the notion of x,V-closedness altogether, and more simply require in Definition 24 that $^\circ$ is closed (and moreover that $\ll o^\circ \gg = x^\eta$ on the nose). The weaker definition we have given is designed with the proof of non-definability in SF in mind: we will be able to show in Section 5 that every (simple) procedure representing the functional $\Phi_{k+1} \in \mathsf{SF}$ is spinal in this weaker sense.[6]

We now digress briefly to explain the small modification of this machinery that we will need in Section 5. Since our purpose there will be to analyse the functional Φ_{k+1} which we defined in Section 1, we shall be working in a setup in which the global variable g has the slightly different type $0 \rightarrow (k+1) \rightarrow (k+1)$. In this setting, we may vary the above definition by coinductively declaring e to be g-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 $g \vdash F_{k+1}[g]$ is spinal. The main result of this section will be that every $\mathrm{PCF}^{\Omega}_{k}$-denotable procedure $\Gamma \vdash p$ is non-spinal (Theorem 28). We shall establish this by induction on the generation of denotable terms as in Theorem 20, the only challenging case being the one for rule 6, which involves plugging. Here we require some technical machinery whose purpose is to show that if the result of a plugging operation is spinal, then a spinal structure must already have been present in one of the components of the plugging: there is no way to 'assemble' a spinal structure from material in non-spinal fragments.

The core of the proof will consist of some lemmas developing the machinery necessary for tackling rule 6. We start with some technical but essentially straightforward facts concerning evaluation and the tracking of subterms and variable substitutions.

Lemma 26 *Suppose that*

$$\Gamma \vdash \ll K[d] \gg =_{\alpha} K'[c]$$

where $K[-], K'[-]$ are concrete meta-term contexts with local environments $\vec{v}, \vec{v}\,'$ respectively, and $\Gamma, \vec{v} \vdash d = \texttt{case}\ gpq\ \texttt{of}\ (\cdots)$, $\Gamma, \vec{v}\,' \vdash c = \texttt{case}\ gp'q'\ \texttt{of}\ (\cdots)$ are concrete expressions. 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.*

[6]It can be shown using Theorem 12 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.

Then:

(i) There is a substitution $^\dagger = [\vec{v} \mapsto \vec{s}]$ *of level* $\leq k$ *arising from the* β*-reductions in the above evaluation, with* $\Gamma, \vec{v}\,' \vdash \vec{s}$ *regular, such that* $\Gamma, \vec{v}\,' \vdash gp'q' =_\alpha \ll (gpq)^\dagger \gg$*, whence* $\ll d^\dagger \gg$ *is of form* `case` $gp'q'$ `of` $(\cdots)$ *up to* $=_\alpha$.[7]

(ii) If furthermore c *is head-spinal w.r.t. some* x, V*, then also* $\ll d^\dagger \gg$ *is head-spinal w.r.t.* x, V.

(iii) If $K[-]$ *contains no* β*-redexes* $P\vec{Q}$ *with* P *of type level* $k+1$*, then* † *is* trivial for level k variables*: that is, there is an injection* ι *mapping each level* k *variable* $v_i \in \vec{v}$ *to a variable* $\iota(v_i) \in \vec{v}\,'$ *such that* $s_i = \iota(v_i)^\eta$.

In reference to part (iii), recall that substitutions $v \mapsto v^\eta$ have no effect on the meaning of a term, as established by Lemma 5. Note that the environments $\vec{v}, \vec{v}\,'$, and hence the injection ι, will in general depend on the concrete choice of $K[d]$ and $K'[c]$. However, for the purpose of proving the theorem, it is clearly harmless to assume that $K'[c]$ is, on the nose, the concrete term obtained by evaluating $K[d]$. In this case, we will see from the proof below that each $\iota(v_i)$ will be either v_i itself or a renaming of v_i arising from the evaluation.

Proof (i) We first formulate a suitable property of terms that is preserved under all individual reduction steps. Let $K[-], d, p, q$ 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 $\ll gP^0Q^0 \gg =_\alpha \ll (gpq)^{\dagger 0} \gg$.

We claim that K^1, P^1, Q^1 enjoy these same properties w.r.t. the local environment $\vec{v}^{\,1}$ for $K^1[-]$. For property 1, clearly $K^1[\texttt{case}\ gP^1Q^1\ \texttt{of}\ (\cdots)]$ cannot contain variables of level $> k$ other than g, because $K^0[\texttt{case}\ gP^0Q^0\ \texttt{of}\ (\cdots)]$ does not. For property 2, we define the required substitution $^{\dagger 1} = [\vec{v} \mapsto \vec{s}^{\,1}]$ by cases according to the nature of the reduction step:

- If the subexpression `case` gP^0Q^0 `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

$$\begin{gathered}\texttt{case}\ (\texttt{case}\ gP^0Q^0\ \texttt{of}\ (i \Rightarrow E_i^0))\ \texttt{of}\ (j \Rightarrow F_j) \quad \rightsquigarrow \\ \texttt{case}\ gP^0Q^0\ \texttt{of}\ (i \Rightarrow \texttt{case}\ E_i^0\ \texttt{of}\ (j \Rightarrow F_j))\end{gathered}$$

  then the conclusion is immediate, noting that $\vec{v}^{\,1} = \vec{v}^{\,0}$ and taking $^{\dagger 1} = {}^{\dagger 0}$.

[7]Note that although both $\ll d^\dagger \gg$ and c have the form `case` $gp'q'$ `of` $(\cdots)$, they will in general have different case-branches, for instance when $K[-]$ is of the form `case` $-$ `of` $(\cdots)$.

- 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 $^{\dagger 1}$ to be $^{\dagger 0}$, with obvious adjustments to compensate for any renaming of bound variables within R_i or $\vec{s}^{\,0}$. In this case $\vec{v}^{\,1}$ may contain more variables than $\vec{v}^{\,0}$, but we will still have that $\Gamma, \vec{v}^{\,1} \vdash \vec{s}^{\,1}$ once these renamings have been effected.

- 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 $Q^1 = Q^0[\vec{x} \mapsto \vec{R}\,']$ for some $\vec{R}\,' =_\alpha \vec{R}$. In this case, the local environment $\vec{v}^{\,1}$ for $K^1[-]$ will be $\vec{v}^{\,0} - \vec{x}$ (perhaps modulo renamings of the v_i^0), so that the conclusion follows if we take $^{\dagger 1} = [\vec{v} \mapsto \vec{s}^{\,1}]$ where $s_i^1 = \ll s_i^0[\vec{x} \mapsto \vec{R}\,'] \gg$ for each i (modulo the same renamings). Note here that $\Gamma, \vec{v}^{\,1} \vdash \vec{s}^{\,1}$ is regular since $\vec{R}$ is regular by condition 1 of the hypothesis.

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, intuitively, $K''[-]$ is fully evaluated down as far as the hole. More formally, there is a finite normal-form context $t[-] \sqsubseteq K''[-]$ containing the hole in $K''[-]$ such that $t[-] \sqsubseteq K'[-]$; from this we may also see that $\ll P' \gg = p'$, $\ll Q' \gg = q'$ and $\ll K''[-] \gg = K'[-]$. Moreover, we now see that K, p, q themselves trivially satisfy the above invariants if we take $^\dagger = [\vec{v} \mapsto \vec{v}^{\,\eta}]$ (Lemma 5 is used here). We therefore infer by iterating the argument above that K'', P', Q' also satisfy these invariants with respect to some $^\dagger = [\vec{v} \mapsto \vec{s}]$ with $\Gamma, \vec{v}\,' \vdash \vec{s}$ regular. (The environment $\Gamma, \vec{v}\,'$ is correct here, as $K'[-], K''[-]$ have the same local environment.) We now have $gp'q' = \ll gP'Q' \gg =_\alpha \ll (gpq)^\dagger \gg$. That the $\vec{v}$ are of level $\leq k$ is automatic, 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 24 that $gp'q'$ and hence $\ll d^\dagger \gg = \texttt{case}\ gp'q'\ \texttt{of}\ (\cdots)$ 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 a substitution variable for a β-reduction. For those v_i that never serve as such a variable, it is clear from the construction that v_i gives rise to some variable $\iota(v_i) \in \vec{v}\,'$ (either v_i itself or a renaming thereof), and that $s_i = v_i^\dagger = \iota(v_i)^\eta$. We wish to show that all $v_i \in \vec{v}$ of level k are in this category.

Recalling that $\vec{v}$ is the local environment for $K[-]$, any $v_i \in \vec{v}$ of level k is bound by the leading λ of some meta-procedure P within $K[-]$ of level $k+1$. By hypothesis, this P does not occur in operator position; nor can it occur as an argument to another λ-abstraction within $K[-]$, since 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 residual $g(\lambda v_i'.E'[-])\cdots$ in each of the intermediate reducts (where v'_i is either v_i or a renaming thereof), and thus that v_i and renamings thereof never serve as substitution variables for β-reductions. $\Box$

Thus, in the setting of the above lemma, if c is head-spinal then 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 will give what we need in order to show that k-pluggings cannot manufacture spinal terms out of non-spinal ones.) This is shown by the next lemma, whose proof forms the most complex and demanding part of the 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 as regards which variables can appear free where, and for this reason we shall make a habit of explicitly recording the variable environment for practically every term or meta-term that we mention.

Lemma 27 *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 $\Gamma, \vec{v}\,' \vdash \ll d[\vec{v} \mapsto \vec{s}] \gg$ is head-spinal with respect to some x, V. Then d itself is spinal.

Proof We begin with some informal intuition. The term $\ll d[\vec{v} \mapsto \vec{s}] \gg$, being head-spinal, 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 $\circ$ (and likewise for o and x). Here the head g of t clearly originates from that of d; likewise, the $\lambda x'$ originates from the leading λ of p within d, rather than from $\vec{s}$. Suppose, however, that the second displayed 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 content 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 12 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 interaction to occur. (The situation is made slightly more complex by the fact that some components of $\circ$ might also involve x', but the same idea applies.) It follows that the second spinal g in t originates from d after all. 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. By renaming variables if necessary, we may assume for clarity that the same variable is never bound in two places within the entire list of terms $d, \vec{s}$, and that all bound variables within $d, \vec{s}$ are distinct from those of $\vec{v}$ and $\vec{v}\,'$.

Let $^{\dagger} = [\vec{v} \mapsto \vec{s}]$, and let us write the subterm p appearing within d as $\Gamma, \vec{v} \vdash \lambda x'^{k}.e$, where $\Gamma, \vec{v}, x' \vdash e$ is regular. Then

$$\ll d^{\dagger} \gg \;\;=\;\; \texttt{case}\; g\,(\lambda x'.\!\ll e^{\dagger} \gg) \ll q^{\dagger} \gg \;\texttt{of}\; (\cdots)\;,$$

and since $\ll d^\dagger \gg$ is head-spinal by hypothesis, $\ll e^\dagger \gg$ will be some 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 itself head-spinal with respect to x' and $\vec{y}\,'$. (Here $\vec{y}\,'$ denotes the local environment for $E[-]$, so that $\Gamma, x', \vec{v}\,', \vec{y}\,'$ contains no repetitions.) 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 $L[-]$; note that $\vec{z}$ is disjoint from $\Gamma, x', \vec{v}\,'$, but may well overlap with $\vec{y}\,'$.) 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 26, 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, whence so is $\Gamma, x', \vec{v} \vdash e^\dagger = D[L[d']]$; condition 2 is immediate in the present setup.

We conclude that there is a substitution $[\vec{y} \mapsto \vec{t}\,]$ (called $[\vec{v} \mapsto \vec{s}\,]$ in the statement of Lemma 26) with $\vec{y}$ the local environment for $D[L[-]]$ and $\Gamma, x', \vec{v}\,', \vec{y}\,' \vdash \vec{t}$ (recalling that $\vec{y}\,'$ are the local variables for $E[-]$), such that $\ll d'[\vec{y} \mapsto \vec{t}\,] \gg$ is head-spinal and indeed of the form $\texttt{case}\ gF'o'\ \texttt{of}\ (\cdots)$ with F', o' as above. 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 in $e^\dagger$ with a substitution variable of level k, so by Lemma 26(iii), the substitution $[\vec{y} \mapsto \vec{t}\,]$ is trivial for variables of level k. Note also that $\vec{y}$ (the environment for $D[L[-]]$) subsumes $\vec{z}$ (the environment for $L[-]$); it is disjoint from $\Gamma, x, \vec{v}\,'$ but may well overlap with $\vec{y}\,'$.

Let us now split the substitution $[\vec{y} \mapsto \vec{t}\,]$ as $[\vec{y}^{\,+} \mapsto \vec{t}^{\,+}, \vec{y}^{\,-} \mapsto \vec{t}^{\,-}]$, where $\vec{y}^{\,+}$ consists of the variables in $\vec{y}$ of level k, and $\vec{y}^{\,-}$ consists of those of level $< k$. As we have noted, the substitution for $\vec{y}^{\,+}$ is trivial: that is, there is a mapping associating with each $y_j \in \vec{y}^{\,+}$ a variable $\iota(y_j) \in \vec{y}\,'$ such that $t_j = \iota(y_j)^\eta$.

Taking stock, we have that

$$\begin{array}{rcl} \Gamma, x', \vec{v}\,', \vec{y}\,' & \vdash & \ll d'[\vec{y} \mapsto \vec{t}\,] \gg \ = \ \texttt{case}\ gF'o'\ \texttt{of}\ (\cdots) \ , \\ \Gamma, \vec{v}\,', \vec{z} & \vdash & d' = \texttt{case}\ gp'q'\ \texttt{of}\ (\cdots) \ , \\ \Gamma, x', \vec{v}\,', \vec{y}\,' & \vdash & \vec{t} \ , \end{array}$$

where $[\vec{y} \mapsto \vec{t}\,]$ is trivial for level k variables, and $gF'o'$ is head-spinal w.r.t. $x, \vec{y}\,'$. From this we may read off that

$$\Gamma, x', \vec{v}\,', \vec{y}\,' \vdash \ll q'[\vec{y} \mapsto \vec{t}\,] \gg \ = \ o' \ .$$

Since $\vec{y}$ subsumes $\vec{z}$, we may henceforth regard q' as a term in environment $\Gamma, \vec{v}\,', \vec{y}$. (This is compatible with the no-variable-hiding condition: our conventions ensure that the variables of $\vec{y} - \vec{z}$ come from d rather than s_i and so do not appear bound in q'.) We may harmlessly write $q'[\vec{y} \mapsto \vec{t}\,]$ as above, even though there are variables of $\vec{y}$ that cannot appear in q'.

Since x' does not occur free in q', each free occurrence of x' in o' above must originate from some $t_j \in \vec{t}$, which must furthermore have some type ρ_j of level $< k$, since if t_j had level k then we would have $t_j = \iota(y_j)^\eta$ which does not contain x' free. In fact, we may decompose the substitution $[\vec{y} \mapsto \vec{t}\,]$ as $[\vec{y}^{\,+} \mapsto \iota(\vec{y}^{\,+})^\eta]$ followed by $[\vec{y}^{\,-} \mapsto \vec{t}^{\,-}]$, since none of variables of $\vec{y}^{\,-}$ appear free in the $\iota(y_j)^\eta$ for $y_j \in \vec{y}^{\,+}$. Setting $q'^* = \ll q[\vec{y}^{\,+} \mapsto \iota(\vec{y}^{\,+})^\eta] \gg$ (so that q'^* is just q' with the variables in $\vec{y}^{\,+}$ rewritten via ι), we therefore have $\ll q'^*[\vec{y}^{\,-} \mapsto \vec{t}^{\,-}] \gg = o'$. Thus:

$$\begin{array}{rcl} \Gamma, \vec{v}\,', \iota(\vec{y}^{\,+}), \vec{y}^{\,-} & \vdash & q'^* : \overline{k}\ , \\ \Gamma, x, \vec{v}\,', \vec{y}\,' & \vdash & t_j : \rho_j \quad \text{for } t_j \in \vec{t}^{\,-}\ , \\ \Gamma, x, \vec{v}\,', \vec{y}\,' & \vdash & \ll q'^*[\vec{y}^{\,-} \mapsto \vec{t}^{\,-}] \gg \;\; = \;\; o'\ . \end{array}$$

Since $o'^\circ \succeq x'^\eta$ for a suitable $x', \vec{y}\,'$-closed substitution $\circ$ (as part of the fact that $\ll d^\dagger \gg$ is head-spinal), the above already comes close to exhibiting $\overline{k}$ as a pseudo-retract of a level $< k$ product type, contradicting Theorem 12. To complete the argument, we must take account of the effect of $\circ$, which we here write as $[\vec{w} \mapsto \vec{r}\,]$ (we may assume that $\vec{w}$ is exactly $\Gamma, \vec{v}\,', \vec{y}\,'$). Reordering our variables, we may now write $x, \vec{w} \vdash \vec{t}^{\,-}$.

Next, let us split $\circ$ into two independent parts: a substitution $[\vec{w}^{\,+} \mapsto \vec{r}^{\,+}]$ covering the variables in $\Gamma, \vec{v}\,', \vec{y}\,'$ of level $\geq k$, and $[\vec{w}^{\,-} \mapsto \vec{r}^{\,-}]$ covering those of level $< k$. Since $\circ$ is $x', \vec{y}\,'$-closed, we have $\vdash \vec{r}^{\,+}$ and $x' \vdash \vec{r}^{\,-}$. Now set $q'^\wr = \ll q'^*[\vec{w}^{\,+} \mapsto \vec{r}^{\,+}] \gg$, so that $\vec{u}^{\,-} \vdash q'^\wr$ where $\vec{u}^{\,-}$ consists of the variables of $\Gamma, \vec{v}\,', \vec{y}$ of level $< k$. The idea is that $\vec{u}^{\,-} \vdash q'^\wr$ may now serve as one half of a suitable pseudo-retraction. For the other half, let $[\vec{u}^{\,-} \mapsto \vec{a}^{\,-}]$ denote the effect of the substitution $[\vec{y}^{\,-} \mapsto \vec{t}^{\,-}]$ followed by $\circ = [\vec{w} \mapsto \vec{r}\,]$ (the order is important here as $\vec{y}^{\,-}$ and $\vec{w}$ may overlap). Since $\vec{u}^{\,-} \subseteq \vec{y} \cup \vec{w}$ and $x, \vec{w} \vdash \vec{t}^{\,-}$ and $x \vdash \vec{r}$, this substitution does indeed cover at least the variables of $\vec{u}^{\,-}$ and we have $x \vdash \vec{a}^{\,-}$. We may now verify that $\vec{u}^{\,-} \vdash q'^\wr$ and $x \vdash \vec{a}^{\,-}$ constitute a pseudo-retraction as follows:

$$\begin{array}{rcl} x' & \vdash & \ll q'^\wr[\vec{u}^{\,-} \mapsto \vec{a}^{\,-}] \gg \\ & = & \ll q'^*[\vec{w}^{\,+} \mapsto \vec{r}^{\,+}][\vec{y}^{\,-} \mapsto \vec{t}^{\,-}][\vec{w} \mapsto \vec{r}\,] \gg \\ & = & \ll (q'^*[\vec{y}^{\,-} \mapsto \vec{t}^{\,-}])[\vec{w} \mapsto \vec{r}\,] \gg \\ & = & \ll o'^\circ \gg \;\; \succeq \;\; x'^\eta\ . \end{array}$$

As regards the second equation here, the first substitution $[\vec{w}^{\,+} \mapsto \vec{r}^{\,+}]$ may be safely omitted as $\vec{w}^{\,+}$ and $\vec{y}^{\,-}$ are disjoint and the terms $\vec{r}^{\,+}$ do not contain any of the $\vec{w}^{\,+}$ or $\vec{y}^{\,-}$ free. We therefore have $\overline{k}$ as a pseudo-retract of a product of level $< k$ types. This contradicts Theorem 12, so the proof of Claim 1 is complete.

We may therefore suppose that in the evaluation $\ll e^{\dagger} \gg = E[c]$, the originating occurrence of the head g in c is as indicated by $\Gamma, x', \vec{v} \vdash e = C[d']$, where $\Gamma, x', \vec{v}, \vec{v}\,'' \vdash d' = \texttt{case}\ gp'q'\ \texttt{of}\ (\cdots)$. (Here $\vec{v}\,''$ is the local environment for $C[-]$. The symbols d', p', q' are available for recycling now that the proof of Claim 1 is complete.)

In order to continue our analysis to greater depth, note that we may write

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

where c is head-spinal w.r.t. $x', \vec{y}\,'$, and the head g of d' is the origin of the head g of c. (Recall that $\vec{y}\,'$ is the local environment for $E[-]$ and that $\Gamma, x', \vec{v} \vdash e = C[d']$ is regular, whence $\mathrm{lv}(\vec{v}\,'') \leq k$.)

We claim that once again we are in the situation of Lemma 26, 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. Condition 2 of the lemma is 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 22 contain no bound variables of level $> k$; hence the same is true for $(\lambda \vec{v}.\, C[d'])\vec{s}$. Applying Lemma 26, 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 $\Gamma, x', \vec{v}\,', \vec{y}\,' \vdash \vec{s}\,^{+}$ are regular, such that

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

is head-spinal w.r.t. $x', \vec{y}\,'$, and indeed of the form $\texttt{case}\ gF'o'\ \texttt{of}\ (\cdots)$ with F', o' as above. (We may in fact write just $\Gamma, x', \vec{v}\,^{+} \vdash d'$ since, by assumption, the variables of $\vec{v}\,'$ do not overlap with the free or bound variables of d so do not appear in d.) We may also read off that $\ll (p')^{\dagger'} \gg = F'$ and $\ll (q')^{\dagger'} \gg = o'$. As regards the substitution ${}^{\dagger'} = [\vec{v}\,^{+} \mapsto \vec{s}\,^{+}]$, it is clear that this extends $[\vec{v} \mapsto \vec{s}]$ since the evaluation of $(\lambda \vec{v}.\, C[d'])\vec{s}$ starts by β-reducing this term. Moreover, the argument of Lemma 26(iii) shows that ${}^{\dagger'}$ 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 $d', \vec{v}\,^{+}, \vec{s}\,^{+}$ themselves satisfy the hypotheses of Lemma 27, with (Γ, x') now playing the role of Γ and $(\vec{v}\,', \vec{y}\,')$ that of $\vec{v}\,'$. Explicitly, we have regular terms

$$\Gamma, x, \vec{v}\,^{+} \vdash d' = \texttt{case}\ gp'q'\ \texttt{of}\ (\cdots)\ , \qquad \Gamma, x, \vec{v}\,', \vec{y}\,' \vdash \vec{s}\,^{+}$$

(so that $\mathrm{lv}(\vec{v}\,^{+}), \mathrm{lv}(\vec{v}\,', \vec{y}\,') \leq k$) where $\Gamma, x, \vec{v}\,', \vec{y}\,' \vdash \ll d'[\vec{v}\,^{+} \mapsto \vec{s}\,^{+}] \gg$ is head-spinal w.r.t. $x', \vec{y}\,'$. We can therefore iterate the whole of 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 †, ${}^{\dagger'}$, ${}^{\dagger''}, \ldots$ applicable to $d, d', d'', \ldots$ respectively, such that $\ll p^{\dagger} \gg$, $\ll (p')^{\dagger'} \gg$, $\ll (p'')^{\dagger''} \gg, \ldots$ coincide with the successive procedure subterms $F, F', F'', \ldots$ from the spine of the original term $\ll d^\dagger \gg$, and likewise $\ll q^\dagger \gg$, $\ll (q')^{\dagger'} \gg$, $\ll (q'')^{\dagger''} \gg, \ldots$ coincide with $o, o', o'' \ldots$.

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 bound locally within p. We will in fact show that d' is head-spinal w.r.t. $x', \vec{v}\,''$, where $\vec{v}\,''$ is the local environment for $C[-]$; this will imply that d is spinal. In the light of Definition 24, it will be sufficient to show that $(q')^{\circ'} \succeq x'^{\eta}$ for some $x', \vec{v}\,''$-closed specialization ${}^{\circ'}$ covering the free variables of q' except x' (namely those of $\Gamma, \vec{v}, \vec{v}\,''$); the same argument will then obviously apply also to $q'', q''', \ldots$.

Recall that $\Gamma, \vec{v}, \vec{v}\,'', x' \vdash q'$ and $\Gamma, \vec{v}\,', \vec{y}\,', x' \vdash o'$. Since $gF'o'$ is head-spinal w.r.t. $x, \vec{y}\,'$, we may as before take ${}^\circ = [\vec{w} \mapsto \vec{r}]$ $x', \vec{y}\,'$-closed such that $o'^{\circ} \succeq x'^{\eta}$, where $\vec{w} = \Gamma, \vec{v}\,', \vec{y}\,'$ and $x' \vdash \vec{r}$. Now define

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

(where we write $\vec{s}^{\,+}$ as $\vec{s}, \vec{s}\,''$, and $\vec{w}^{\,\Gamma} \mapsto \vec{r}^{\,\Gamma}$ denotes the restriction of ${}^\circ$ to Γ). This covers the free variables of q' except x', and we have $x' \vdash \vec{s}^{\,\circ}, (\vec{s}\,'')^{\circ}, \vec{r}^{\,\Gamma}$ because $\vec{w} \vdash \vec{s}, \vec{s}\,''$ and $x \vdash \vec{r}$. Moreover, we have

$$
q'^{\circ'} \;=\; (q'[\vec{v}^{\,+} \mapsto (\vec{s}^{\,+})^{\circ}])[\vec{w}^{\,\Gamma} \mapsto \vec{r}^{\,\Gamma}] \;=\; (q'^{\dagger'})^{\circ} \;\approx\; o^{\circ} \;\succeq\; x'^{\eta}
$$

since $(\vec{s}^{\,+})^{\circ}$ contains no free variables except x. To check that ${}^{\circ'}$ is $x', \vec{v}\,''$-closed, it remains to show that that $u^{\circ'}$ may contain x' free only when $u \in \vec{v},''$ and u is of level $< k$. (Indeed, it is because of the possibility of x' occurring free in these terms that the machinery of x, V-closed substitutions is necessary at all.) The remaining cases are handled as follows:

- The terms $\vec{s}$ exist in environment $\Gamma, \vec{v}\,'$, so do not involve x' or any of the variables of $\vec{y}\,'$. Since ${}^\circ$ is $x', \vec{y}\,'$-closed, it follows that the terms $\vec{s}^{\,\circ}$ do not involve x'.

- For any variables $v \in \vec{v}\,''$ of level k, we have $v^{\dagger'} = v^{\eta}$ which contains no free variables of level $< k$, so that $(v^{\dagger'})^{\circ}$ cannot involve x'.

- The $\vec{r}^{\,\Gamma}$ cannot involve x', since ${}^\circ$ is $x', \vec{y}\,'$-closed and Γ is disjoint from $\vec{y}\,'$.

This completes the verification that d is spinal. $\Box$

From the above lemma we may immediately conclude, for example, that in the setting of Lemma 26(ii) the term d is spinal.

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

Theorem 28 *Every* $\mathrm{PCF}^{\Omega}_{k}$*-denotable procedure* $\Gamma \vdash p$ *is non-*g*-spinal where* $g : (k+1) \to (k+1)$.

Proof In the light of Section 3, it will suffice to show that the clauses of Theorem 20 cannot generate spinal terms from non-spinal ones. For clauses 1–5 this is very easily seen. For clause 6, it will be sufficient to show that non-spinal terms are closed under k-plugging, and it is here that the machinery of Lemmas 26 and 27 comes into play.

Suppose that $t = \ll \Pi_{\Gamma,Z}(e,\xi) \gg$ where $\Gamma, Z \vdash e$ and $\Gamma, Z \vdash \xi(z)$ for each $z \in Z$. 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.

It will be natural to frame the argument contrapositively. Suppose that t is spinal, i.e. t contains some head-spinal expression c at position $K[-]$. We shall focus on the head occurrence of g in c. Clearly this occurrence must originate from one of the ingredients $\Gamma, Z \vdash e$ or $\Gamma, Z \vdash \xi(z)$ of the plugging $\Pi_{\Gamma,Z}(e,\xi)$; let us denote this ingredient by $\Gamma, Z \vdash t_0$. We will show that t_0 itself is spinal.

Suppose that the relevant 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 26, 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 condition 2 is immediate in the present setup.

By Lemma 26, 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 27, so may conclude that d itself is spinal, and hence that t_0 is spinal. We have thus shown that k-plugging cannot assemble spinal terms from non-spinal ones, and this completes the proof. $\square$

In particular, since the procedure $g \vdash F_{k+1}[g]$ mentioned at the start of the section is spinal, we may conclude that this procedure is not PCF^Ω_k-denotable, and hence neither is the procedure Y_{k+1}. This establishes Theorem 11.

We conclude the section by mentioning some minor variations on Theorem 28 that we will require below. First, as already indicated, the whole of the above proof goes through for the modified notion of spinal term appropriate to a variable $g : 0 \to (k+1) \to (k+1)$. Secondly, the theorem also holds for an innocuous extension of $\mathrm{PCF}^{\Omega}_{k}$ with a constant *byval* : $(\mathtt{N} \to \mathtt{N}) \to \mathtt{N} \to \mathtt{N}$, whose denotation in SP^0 we take to be

$$\lambda f x.\, \mathtt{case}\ x\ \mathtt{of}\ (i \Rightarrow \mathtt{case}\ fi\ \mathtt{of}\ (j \Rightarrow j))\ .$$

To see that the proof of Theorem 28 goes through in the presence of *byval*, it suffices simply to add an extra clause to the inductive proof noting that the procedure for *byval* is non-spinal. This mild extension will allow a significant simplification of the forms of procedures that we need to consider in Section 5.[8]

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 $\mathrm{PCF}^{\Omega}_{k}$-denotable, 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, we may present the canonical procedure Y_{k+1} as $\lambda gx.C[g,x]$, where

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

However, another candidate for the fixed point operator is

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

Intuitively, this computes the desired value twice, discarding the first result.

As a slightly more subtle example, consider the procedure

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

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 $\lambda x'.\mathtt{case}\ A[g,x]\ \mathtt{of}\ (i \Rightarrow C[g,x'])$ 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

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

[8]The operator *byval* plays a major role in [17, Section 7.1], where it is shown that every element of SP^0 is denotable in PCF^{Ω} + *byval*. The sense in which it is innocuous is that its denotation in SF coincides with that of $\lambda fx.\, \mathit{ifzero}\, x\, (fx)(fx)$; thus *byval* adds nothing to the expressivity of $\mathrm{PCF}^{\Omega}_{k}$ as regards SF.

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} is spinal; from this it would follow easily by Theorem 28 that the fixed point functional Y_{k+1} in SF is not denotable in PCF_k^Ω.

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 $Y_{0\to(k+1)}$, although even here, it is simplest to concentrate not on $Y_{0\to(k+1)}$ itself, but on a certain functional that is readily definable from it, namely the functional Φ_{k+1} introduced in Section 1. Nonetheless, the above examples of 'time-wasting' procedures illustrate some of the situations that our proof will need to deal with, and they may help to motivate some of the technical machinery that follows.

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

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

so that informally

$$\Phi_{k+1}\, g\, n \ = \ g\, n\, (g\, (n+1)\, (g\, (n+2)\, (\cdots)))\ .$$

For the rest of this section we will write Φ_{k+1} simply as Φ. For each $n \in \mathbb{N}$, let $g \vdash \Phi^n[g] = \Phi\, g\, \widehat{n} : k+1$, and let $p_n \in \mathsf{SP}(k+1)$ be 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:

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

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 24). Using Theorem 28 (adapted 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_k^Ω-denotable 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 $\lambda x.\, \mathtt{case}\ \mathit{garo}\ \mathtt{of}\ (\cdots)$, 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. We achieve this 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 $pq \approx \lambda.n$ 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 $q \succeq' p_n$, where p_n is the canonical procedure for $\Phi^n[g]$ as above.

We shall say a procedure $g \vdash q$ is *simple* if for every application $garo$ appearing within q, the first argument a is just a numeral $\lambda.n$. The following observation simplifies our analysis of terms considerably; it uses the operator *byval* and its NSP interpretation, as introduced at the end of Section 4.

Proposition 29 *If there is a procedure $g \vdash q \succeq' p_n$-denotable in* PCF^Ω_k, *then* *there is a simple procedure* $g \vdash q' \succeq' p_n$ *denotable in* $\mathrm{PCF}^{\Omega}_{k} + \mathit{byval}$.

Proof Suppose $g \vdash q$ is $\mathrm{PCF}^{\Omega}_{k}$-denotable where $q \succeq' p_n$, and write

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

It is easy to see that

$$S[g] \;=\; [\![\lambda izx.\,\mathit{byval}\,(\lambda j.gjzx)\,i\,]\!]_g\ .$$

Take $g \vdash q' = \ll q[g \mapsto S[g]] \gg$, so that q' is denotable in $\mathrm{PCF}^{\Omega}_{k} + \mathit{byval}$. Then $q \approx' q'$ 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 $\ll q[g \mapsto S[g]] \gg$. $\Box$

For any $n \leq n'$, let us define the *masking* $\mu_{n,n'}(g)$ of g to be the following procedure term (here $\rho = 0 \rightarrow (k+1) \rightarrow (k+1)$):

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

(The wildcard symbol '$-$' covers all branch indices not covered by the preceding clauses.) 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 closed meta-term $\vdash P : \rho$ 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 closed 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 30 *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* $\vdash G$ *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

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

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

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

contradicting $q \succeq' p_n$ (note that G' is strict). 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

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

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

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

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 almost 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). However, there still remains a small mismatch between the hypothesis and the conclusion, in that claim 3 concerns not r itself but rather $\ll r[g \mapsto \mu_n(g)] \gg$. (In the light of claim 3, the variable x may be safely ignored here.) This mismatch is repaired by the following lemma, which lends itself to iteration to any depth. Note the entry of a term context $E[-]$ here.

Lemma 31 *Suppose $g \vdash q$ is simple and $q[g \mapsto \mu_{n,n'}(g)] \succeq' p_{n'}$. Then q has the form $\lambda x^k.\,E[\mathtt{case}\ garo\ \mathtt{of}\ (\cdots)]$, where:*

1. *$E[-]$ has empty local variable environment,*

2. $a = \lambda.n'$,

3. $o[g \mapsto G] \succeq x^\eta$ *whenever* $\vdash G$ *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 30 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 26(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 $r'[g \mapsto \mu_{n'}(g)] \succeq' p_{n'+1}$. □

Corollary 32 *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 24 (subject to the relevant adjustments for $g : 0 \to (k+1) \to (k+1)$). Note that at each level, a suitable substitution $^\circ$ will be the closed one that specializes g to $\lambda izx.0$ (which is trivial at all n), and all variables x^k other than the innermost-bound one to $\bot$. □

Thus, if there exists a PCF^Ω_k-denotable procedure $t \approx \lambda g.p_0$, for instance, then by Proposition 29 there is also a simple such procedure t' denotable in $\mathrm{PCF}^\Omega_k + \mathit{byval}$, and by Corollary 32, this t' will be spinal in the modified sense. But this contradicts Theorem 28 (understood relative to the modified setting, and applied to $\mathrm{PCF}^\Omega_k + \mathit{byval}$ as indicated at the end of Section 4). We conclude that no $t \approx \lambda g.p_0$ can be PCF^Ω_k-denotable. 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 denotable in PCF^Ω_k. On the other hand, this element is obviously denotable relative to $Y_{0\to(k+1)} \in \mathsf{SF}$ even in PCF_1, so the proof of Theorems 1(i) and 2(i) is complete. This establishes Berger's conjecture, and also suffices for the proof of Corollary 3.

6 Extensional non-definability of Y_{k+1}

We have so far shown that the element $Y_{0\to(k+1)} \in \mathsf{SF}$ is not $\mathrm{PCF}^{\Omega}_{k}$-denotable. We shall now refine our methods slightly to show that even $Y_{k+1} \in \mathsf{SF}$ is not $\mathrm{PCF}^{\Omega}_{k}$-denotable. Since $\overline{k+1}$ is clearly a PCF_0-definable retract of every level $k+1$ type, this will establish that no $Y_\sigma \in \mathsf{SF}$ where $\mathrm{lv}(\sigma) = k+1$ is denotable in $\mathrm{PCF}^{\Omega}_{k}$.

The idea is as follows. Within each type level $k \geq 1$, we can stratify the types into *sublevels* (k,l) where $l = 1,2,\ldots$, essentially by taking account of the 'width' of the type as well as its depth. We thus obtain a sublanguage $\mathrm{PCF}^{\Omega}_{k,l}$ of $\mathrm{PCF}^{\Omega}_{k}$ by admitting Y_σ only for types σ of sublevel (k,l) or lower. We show that, roughly speaking, all our previous methods can be adapted to show that for a given k, the languages $\mathrm{PCF}^{\Omega}_{k,l}$ for $l = 1,2,\ldots$ form a strict hierarchy. (This is true as regards definability in SP^0; for SF, we will actually show only that $\mathrm{PCF}^{\Omega}_{k,l}$ is strictly weaker than $\mathrm{PCF}^{\Omega}_{k,l+2}$.) This more refined hierarchy is of some interest in its own right, and illustrates that the structure of SF is much richer than that manifested by the pure types.[9]

Any term of $\mathrm{PCF}^{\Omega}_{k}$ will be a term of some $\mathrm{PCF}^{\Omega}_{k,l}$. Previously we showed only that no such term could define $Y_{0\to(k+1)} \in \mathsf{SF}$; however, we now see that there is actually plenty of spare headroom between $\mathrm{PCF}^{\Omega}_{k,l}$ and the pure type $k+1$. Specifically, there are operators Y_σ in $\mathrm{PCF}^{\Omega}_{k,l+2}$ that are not $\mathrm{PCF}^{\Omega}_{k,l}$-denotable; and since all such σ are of level $\leq k$ and are thus easily seen to be retracts of $\overline{k+1}$, we may conclude that $Y_{k+1} \in \mathsf{SF}$ is not $\mathrm{PCF}^{\Omega}_{k}$-denotable.

The following definition sets out the more fine-grained stratification of types that we shall use.

Definition 33 *(i) The* width $w(\sigma)$ *of a type σ is defined inductively as follows:*

$$w(\mathtt{N}) \;=\; 0\,, \qquad w(\sigma_0,\ldots,\sigma_{r-1}\to\mathtt{N}) \;=\; \max(r, w(\sigma_0),\ldots,w(\sigma_{r-1}))\,.$$

For $k,l \geq 1$, we say σ has sublevel (k,l) *if $\mathrm{lv}(\sigma) = k$ and $w(\sigma) = l$. If $\sigma = \mathtt{N}$, we simply say that σ has* sublevel 0. *We order sublevels in the obvious way: 0 is the lowest sublevel, and $(k,l) < (k',l')$ if either $k < k'$ or $k = k'$ and $l < l'$.*

(ii) For each k,l we define a type $\rho_{k,l}$ by:

$$\rho_{0,l} \;=\; \mathtt{N}\,, \qquad \rho_{k+1,l} \;=\; \rho_{k,l},\ldots,\rho_{k,l}\to\mathtt{N} \;\;\textit{(l arguments)}\,.$$

When $k,l \geq 1$, we may call $\rho_{k,l}$ the homogeneous *type of sublevel (k,l).*

The following facts are easily established. Here we shall say that σ is a *simple retract* of τ if there is a PCF_0-definable retraction $\sigma \lhd \tau$ within SP^0.

Proposition 34 *(i) For $k \geq 1$, every type of sublevel (k,l) or lower is a simple retract of $\rho_{k,l}$. Hence, for all $k \geq 0$, for every finite list of types σ_i of level $\leq k$, there is some l such that each σ_i is a simple retract of $\rho_{k,l}$.*

[9]This contrasts with the situation for extensional *total* type structures over $\mathbb{N}$, for example. There, under mild conditions, every simple type is isomorphic to a pure type: see Theorem 4.2.9 of [17].

(ii) Every finite product of level $\leq k$ types is a simple retract of $\overline{k+1}$.
(iii) If σ is a simple retract of τ, then Y_σ is PCF_0-definable from Y_τ in SP_0.

Proof (i) The first claim (for $k \geq 1$) is easy by induction on k, and the second claim (which is trivial when $k = 0$) follows easily.

(ii) By induction on k. The case $k = 0$ is easy. For $k \geq 1$, suppose $\sigma_0, \ldots, \sigma_{m-1}$ are level $\leq k$ types, where $\sigma_i = \tau_{i0}, \ldots, \tau_{i(n_i-1)} \to \mathtt{N}$ for each i. Here the τ_{ij} are of level at most $k-1$, so by (i), we may choose l such that each τ_{ij} is a simple retract of $\rho_{k-1,l}$. Taking $n = \max_i n_i$, we then have that each σ_i is a simple retract of $\rho_{k-1,l}, \ldots, \rho_{k-1,l} \to \mathtt{N}$ (with n arguments). The product $\Pi_i \sigma_i$ is therefore a simple retract of the type $\sigma = \mathtt{N}, \rho_{k-1,l}, \ldots, \rho_{k-1,l} \to \mathtt{N}$. But by the induction hypothesis, the product of $\mathtt{N}, \rho_{k-1,l}, \ldots, \rho_{k-1,l}$ is a simple retract of $\overline{k}$ whence σ itself is a simple retract of $\overline{k+1}$.

We leave (iii) as an exercise. $\Box$

Next, we adapt the proof of Theorem 12 to establish the crucial gap between $\rho_{k,l}$ and $\rho_{k,l+1}$. This gives an indication of how our methods may be used to map out the embeddability relation between types in finer detail, although we leave an exhaustive investigation of this to future work.

Theorem 35 *Suppose $k \geq 1$. Within SF, the type $\rho_{k,l+1}$ is not a pseudo-retract of any finite product of types of sublevel $\leq (k, l)$ or lower.*

Proof In view of Proposition 34(i), it will suffice to show that $\rho_{k,l+1}$ is not a pseudo-retract of a finite power of $\rho_{k,l}$. We argue by induction on k. The arguments for both the base case $k = 1$ and the step case $k > 1$ closely parallel the argument for the step case in Theorem 12, so we treat these two cases together as far as possible, omitting details that are easy adaptations of those in the earlier proof.

Suppose for contradiction that there were procedures

$$z^\rho \vdash t_i : \sigma \ \ (i < m) \ , \qquad x_0^\sigma, \ldots, x_{m-1}^\sigma \vdash r : \rho \ ,$$

where $\rho = \rho_{k,l+1}$, $\sigma = \rho_{k,l}$, such that $z \vdash r[\vec{x} \mapsto \vec{t}\,] \succeq z^\eta$. Let $v = \ll r[\vec{x} \mapsto \vec{t}\,] \gg$, so that $z \vdash v \succeq z^\eta$. As in the proof of Theorem 12, one may show that v has the form $\lambda f_0 \ldots f_l.\, \mathtt{case}\ z p_0 \ldots p_l\ \mathtt{of}\ (\cdots)$ where $p_i[z \mapsto \lambda \vec{w}.0] \succeq f_i$ for each i. Next, we note that $r[\vec{x} \mapsto \vec{t}\,]$ reduces in finitely many steps to a head normal form $\lambda f_0 \ldots f_l.\, \mathtt{case}\ z P_0 \ldots P_l\ \mathtt{of}\ (\cdots)$ where $\ll P_i \gg = p_i$ for each i; moreover, the ancestor of the leading z here must lie within some t_i, say at the head of some subterm $z\vec{P}\,'$, where $\vec{P}$ is an instance of $\vec{P}\,'$ via some substitution †.

At this point, the arguments for the base case and step case part company. In the base case $k = 1$, we have that $z^\rho \vdash t_i : \sigma$ where $\rho = \mathtt{N}^{l+1} \to \mathtt{N}$ and $\sigma = \mathtt{N}^l \to \mathtt{N}$; thus there are no bound variables within t_i except the top-level ones—say $w_0, \ldots, w_{l-1}$, all of type $\mathtt{N}$. So in fact † has the form $[\vec{w} \mapsto \vec{W}]$ for certain meta-terms $f_0^{\mathtt{N}}, \ldots, f_l^{\mathtt{N}}, z \vdash W_j : \mathtt{N}$. Now consider the terms $f_0, \ldots, f_l \vdash \vec{W}^*$ and $\vec{w} \vdash \vec{P}\,'^*$, writing * for the substitution $[z \mapsto \lambda w.0]$. These compose to yield $\vec{f} \vdash \ll \vec{P}^* \gg = \ll \vec{p}^* \gg \succeq f$, so we have expressed $\mathtt{N}^{l+1}$ as a pseudo-retract of

$\mathtt{N}^l$ within SF. As already noted in the course of the proof of Theorem 12, this is impossible.

For the step case $k > 1$, we proceed much as in the proof of Theorem 12, using the substitution † to express $\rho_{k-1,l+1}$ as a retract of a finite product of types of sublevel $(k-1, l)$ or lower, contrary to the induction hypothesis. We leave the remaining details as an exercise. □

We now outline how the ideas of Sections 3, 4 and 5 may be adapted to show that $Y_{\rho_{k,l+1}} \in \mathsf{SP}^0$ is not $\mathrm{PCF}^{\Omega}_{k,l}$-denotable. We assume that $k \geq 2$ for the time being (the case $k = 1$ will require special treatment). The adaptations are mostly quite systematic: the type $\rho_{k,l+1}$ now plays the role of $\overline{k+1}$; types of sublevel $\leq (k, l)$ play the role of types of level $\leq k$; $\rho_{k-1,l+1}$ plays the role of $\overline{k}$; and types of sublevel $\leq (k-1, l)$ play the role of types of level $< k$. Since the proof we are adapting is quite lengthy, we leave many routine details to be checked by the reader.

First, the evident adaptation of Definition 18 yields a notion of *k, l-plugging* where the plugging variables are required to be of sublevel $\leq (k, l)$, and we thus obtain an inductive characterization of the $\mathrm{PCF}^{\Omega}_{k,l}$-denotable procedures analogous to Theorem 20. We also adapt the notion of *regular* meta-term as follows:

Definition 36 *Suppose $g : \rho_{k,l+1} \to \rho_{k,l+1}$ for $k, l \geq 1$. An environment Γ is g-$(k,l$-$)$*regular *if Γ contains g but no other variables of sublevel $> (k, l)$. A meta-term T is g-regular if it contains no variables of sublevel $> (k, l)$ except possibly for free occurrences of g. We say $\Gamma \vdash T$ is g-regular if both Γ and T are g-regular.*

Next, we proceed to the ideas of Section 4. Our convention here will be that Γ ranges over regular environments, and Roman capitals V, Z range over sets of variables of sublevel $\leq (k, l)$. The notions of x, V-closed substitution and spinal term carry over as follows:

Definition 37 *Suppose $g : \rho_{k,l+1} \to \rho_{k,l+1}$ where $k \geq 2$ and $l \geq 1$.*

*(i) If $\vec{x}$ is a list of variables of type $\rho_{k-1,l+1}$ and V a set of variables of sublevel $\leq (k, l)$, a substitution $^\circ = [\vec{w} \mapsto \vec{r}]$ is called $\vec{x}, V$-*closed *if the r_i contain no free variables, except that if $w_i \in V$ and w_i is of sublevel $\leq (k-1, l)$ then r_i may contain the $\vec{x}$ free.*

*(ii) Suppose $\Gamma \vdash e$ is g-k, l-regular and $\vec{x}, V \subseteq \Gamma$. We coinductively declare e to be g-*head-spinal *w.r.t. $\vec{x}, V$ iff e has the form $\mathtt{case}\ g(\lambda \vec{x}\,'.E[e'])\vec{o}\ \mathtt{of}\ (\cdots)$, where $E[-]$ is an expression context, and*

- *for some $\vec{x}, V$-closed specialization $^\circ$ covering the free variables of $\vec{o}$ other than those of $\vec{x}$, we have $\vec{o}^{\,\circ} \succeq \vec{x}^{\,\eta}$,*
- *e' is g-head-spinal w.r.t. $\vec{x}\,', V'$, where V' is the local variable environment for $E[-]$.*

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

Lemma 26 and its proof go through with the above adaptations; here the local environments $\vec{v}, \vec{v}\,'$ are now of sublevel $\leq (k,l)$, and part (iii) of the lemma now states that if $K[-]$ contains no redexes with operator of sublevel $> (k,l)$, then the substitution † is trivial for variables of sublevel $\geq (k-1, l+1)$. The crucial Lemma 27, which forms the heart of our proof, now translates as follows:

Lemma 38 *Suppose $g : \rho_{k,l+1} \rightarrow \rho_{k,l+1}$ and we have g-regular terms*

$$\Gamma, \vec{v} \vdash \ d \ = \ \mathtt{case}\ gpq\ \mathtt{of}\ (\cdots)\ , \qquad \Gamma, \vec{v}\,' \vdash \vec{s},$$

where $\vec{v}, \vec{v}\,'$ are of sublevel $\leq (k,l)$, and $\Gamma, \vec{v}\,' \vdash \ll d[\vec{v} \mapsto \vec{s}] \gg$ is g-head-spinal with respect to some $\vec{x}, V$. Then d itself is g-spinal.

The entire proof of this lemma translates systematically according to the correspondences we have indicated, invoking Theorem 35 for the fact that $\rho_{k-1,l+1}$ is not a pseudo-retract of a product of sublevel $\leq (k-1,l)$ types. The analogue of Theorem 28 now goes through readily, so we obtain:

Theorem 39 *If $k \geq 2$ and $l \geq 1$, every $\mathrm{PCF}^{\Omega}_{k,l}$-denotable procedure is non-g-spinal where $g : \rho_{k,l+1} \rightarrow \rho_{k,l+1}$.* $\Box$

As in our original proof, we will actually use a version of this theorem for the modified notion of spinal term that incorporates the extra argument b, and for the extension of $\mathrm{PCF}^{\Omega}_{k,l}$ with the operator *byval*.

To adapt the material of Section 5, we now take g to be a variable of type $0 \rightarrow \rho_{k,l+1} \rightarrow \rho_{k,l+1}$, and argue that the $\mathrm{PCF}_{k,l+2}$-denotable element $\Phi = \lambda g.\, Y_{0\rightarrow\rho_{k,l+1}}(\lambda f n.\, g\, n\, (f\, (suc\, n)))$ within SF is not $\mathrm{PCF}^{\Omega}_{k,l}$-denotable. The proof is a completely routine adaptation of that in Section 5. Since $Y_{0\rightarrow\rho_{k,l+1}}$ is readily definable from Y_{k+1} by Proposition 34, this implies that $Y_{k+1} \in \mathsf{SF}$ is not $\mathrm{PCF}^{\Omega}_{k,l}$-denotable. We have thus shown:

Theorem 40 *(i) For $k \geq 2$ and $l \geq 1$, the element $Y_{0\rightarrow\rho_{k,l+1}} \in \mathsf{SF}$ is denotable in $\mathrm{PCF}_{k,l+2}$ but not in $\mathrm{PCF}^{\Omega}_{k,l}$.*

(ii) For $k \geq 2$, the element $Y_{k+1} \in \mathsf{SF}$ is not denotable in $\mathrm{PCF}^{\Omega}_{k}$.

A slightly different approach is needed for the case $k = 1$. This is because at level 0 our only type is `N`, so we are unable to make a distinction between sublevels l and $l+1$. To establish Lemma 38 in this case, we again wish to show that we cannot pass in the content of the relevant $\vec{x}\,'$ to the relevant s_i, but now the idea is to appeal to the fact that $\vec{x}\,'$ consists of $l+1$ variables of type `N`, whereas s_i accepts at most l arguments of type `N`. (We have already seen that $\mathtt{N}^{l+1}$ cannot be a retract of $\mathtt{N}^{l}$.) However, we also need to exclude the possibility that the substitution $^\circ$ is being used to import some components of $\vec{x}\,'$. We can achieve this if we require $^\circ$ to be actually closed rather than just $\vec{x}\,', V'$-closed, and it turns out that this is permissible if we also tighten our notion of spinal term slightly, essentially to ensure that no intermediate applications of g appear in between those declared to constitute the spine of the term:

Definition 41 *Suppose $g : \rho_{1,l+1} \to \rho_{1,l+1}$ where $l \geq 1$.*

Suppose $\Gamma \vdash e$ is g-1, l-regular and $\vec{x} \subseteq \Gamma$. We coinductively declare e to be strongly g-head-spinal *w.r.t. $\vec{x}$ iff e has the form* $\texttt{case}\ g(\lambda\vec{x}\,'.E[e'])\vec{o}\ \texttt{of}\ (\cdots)$, *where $E[-]$ is an expression context, and*

- *the hole within $E[-]$ does not itself occur within an application gpq,*
- *for some closed substitution $^\circ$ covering the free variables of $\vec{o}$ other than those of $\vec{x}$, we have $\vec{o}^\circ \succeq \vec{x}^{\,\eta}$,*
- *e' is strongly g-head-spinal w.r.t. $\vec{x}\,'$.*

The notion of strongly g-spinal term follows suit.

The counterpart of Lemma 26 goes through as expected, although without part (iii): the relevant sublevel distinction does not exist at type level 0, and we cannot conclude that the substitution in question is trivial for all variables of type $\mathtt{N}$. We may now indicate the required changes to Lemma 38 and its proof:

Lemma 42 *Suppose $g : \rho_{1,l+1} \to \rho_{1,l+1}$ and we have $1, l$-regular terms*

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

where $\vec{v}, \vec{v}\,'$ are of sublevel $\leq (1,l)$, and $\Gamma, \vec{v}\,' \vdash \ll d[\vec{v} \mapsto \vec{s}] \gg$ is strongly g-head-spinal with respect to some $\vec{x}$. Then d itself is strongly g-spinal.

Proof The proof of Lemma 27 up to the end of the proof of Claim 1 adapts straightforwardly, and is somewhat simplified by the fact that the substitution $^\circ$ is closed. As sketched above, the crucial contradiction is provided by the fact that $\mathtt{N}^{l+1}$ is not a pseudo-retract of $\mathtt{N}^l$ in SF.

For the remainder of the proof, the key point to note is that $\vec{v}\,''$ (the local environment for $C[-]$) is actually empty in this case. This is because $C[-]$ is in normal form and contains no free variables of level ≥ 2 except g, so any λ-term containing the hole would need to appear as an argument to g. It would then follow that the hole within $E[-]$ lay within an argument to some occurrence of g, as precluded by the definition of strongly spinal term.

It follows trivially that the $\vec{x}\,', \vec{v}\,''$-closed substitution $^{\circ'}$ constructed at the very end of the proof is actually closed. Moreover, the spinal structure of d' identified by the proof cannot contain any intermediate applications of g, since these would give rise under evaluation to intermediate applications of g in the spine of $\ll d[\vec{v} \mapsto \vec{s}] \gg$ as precluded by Definition 41. Thus, the identified spinal structure in d' is actually a strongly spinal structure, and the argument is complete. $\Box$

A trivial adaptation of the proof of Theorem 28 now yields:

Theorem 43 *No $\mathrm{PCF}^{\Omega}_{1,l}$-denotable procedure can be strongly g-spinal where $g : \rho_{1,l+1} \to \rho_{1,l+1}$.* $\Box$

As before, this adapts easily to a variable g of type $0 \to \rho_{1,l+1} \to \rho_{1,l+1}$. From here on, we again follow the original proof closely. The only additional point to note is that in place of Corollary 32 we now require that any simple $g \vdash q$ with $q \succeq' p_n$ must be *strongly* spinal, but this is already clear from the proof of Lemma 31. We therefore have everything we need for:

Theorem 44 *(i) For any $l \geq 1$, $Y_{\mathtt{N}^{l+2} \to \mathtt{N}} \in \mathsf{SF}$ is not denotable in* $\mathrm{PCF}^{\Omega}_{1,l}$.
(ii) $Y_2 \in \mathsf{SF}$ is not denotable in $\mathrm{PCF}^{\Omega}_{1}$. $\square$

The proof of Theorems 1 and 2 is now complete.

7 Related and future work

7.1 Other hierarchies of Y-combinators

There have been a number of previous results from various research traditions showing that in some sense the power of level k recursions increases strictly with k. Whilst many of these results look tantalizingly similar to ours, it turns out on inspection that their mathematical substance is quite different, and we do not expect any substantial technical connections with our own work to be forthcoming. Nonetheless, it is interesting to see how strikingly different ideas and methods arising in other contexts can lead to superficially similar results.

Previous results to the effect that Y-combinators for level $k+1$ are not definable from those for level k have been obtained by Damm [6] and Statman [27]. 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 with 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,[10] 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-denotability in SP^0, an approach along Statman's lines would be unlikely to yield much information,

[10]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.

since there is no reason why the number of unfoldings of Y_k required to generate the NSP for Y_{k+1} to depth n should not grow dramatically as a function of n.

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. (Actually, Damm's λ-terms involve a restriction to *derived types*, which has the effect of limiting attention to what are elsewhere called *safe* recursion schemes.) 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 if programs are considered up to *tree equality*, then safe level $k+1$ recursions give more expressive power than safe level k ones. Damm's result is thus distinguished from Statman's in two ways: by the restriction to safe recursion schemes, and by the use of tree equality in place of the stricter computational equality. This latter point brings Damm's work somewhat closer in spirit to our work: indeed, in the case of pure λY, tree equality agrees with equality of innocent strategies if the base type is interpreted as a certain trivial game—or equivalently with equality in a variant of our SP^0 with no ground type values. However, in the case of a signature for PCF, tree equality will still be considerably more fine-grained than equality in SP^0, let alone in SF, since in effect the PCF constants are left uninterpreted. It therefore seems unlikely that an approach to our theorems via Damm's methods is viable.

The strictness of the recursion scheme hierarchy was further investigated by Ong [22], who used innocent game semantics to show that the complexity of certain model checking problems for trees generated by level k recursions increases strictly with k. It was also shown by Hague, Murawski, Ong and Serre [7] that the trees generated by level k recursion schemes (with no safety restriction) were precisely those that could be generated by *collapsible pushdown automata* of order k. Later, Kartzow and Parys [10] used pumping lemma techniques to show that the collapsible pushdown hierarchy was strict, hence so was the recursion scheme hierarchy with no safety restriction. Again, despite the intriguing parallels to our work, these results appear to be manifesting something quite different: in our setting, the power of level k recursion has nothing to do with the 'difficulty of computing' the relevant NSPs in the sense of automata theory, since the full power of Turing machines is required even at level 1.

Also of interest is the work of Jones [9] from the functional programming community. Jones's motivation is close to ours in that he seeks to give mathematical substance to the programming intuition that some (combinations of) language features yield 'more expressive power' than others. As Jones notes, an obvious obstacle to obtaining such results is that all programming languages of practical interest are Turing complete, so that no such expressivity distinctions are visible at the level of the (first-order) computable functions. Whereas we have responded to this by considering the situation at higher types, where genuine expressivity differences do manifest themselves, Jones investigates the power of (for example) recursion at different type levels in the context of a restricted language of 'read-only' or 'cons-free' programs. Amongst other results, he shows that in such a language, if data values and general recursion of type level $k \geq 1$ are admitted, then the computable functions from lists of booleans to booleans are exactly the $\textsc{exp}^k\textsc{time}$ computable ones. These results inhabit a mathematical territory very different from ours, although yet again, the basic moral that the power of recursion increases strictly with its type level shines through.

7.2 Relationship to game semantics

Next, we comment briefly how our work relates to the known *game models* of PCF [1, 8]. 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 (see [17, Section 7.4]). This raises the obvious question of whether our proofs could be conducted equally well, or better, in a game-semantical setting.

Whilst a direct translation is presumably possible, our present impression is that the sequential procedure presentation, and our calculus of meta-terms in particular, allows one to see the wood for the trees more clearly, and also to draw more easily on familiar intuitions from λ-calculus. However, a closer look at game-semantical approaches would be needed in order to judge whether either approach really offers some genuine mathematical or conceptual advantage over the other.

7.3 Sublanguages of PCF: further questions

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

So far, we have worked mainly with a coarse stratification of types in terms of their levels, although we have illustrated in Section 6 how finer stratifications are also possible. Naturally, there is scope for a still more fine-grained analysis of types and the relative strength of their Y-combinators; this is of course 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. Our analysis in Section 6 has shown that, for $l \geq 1$,

- the element $Y_{\mathtt{N}^{l+1}\to\mathtt{N}} \in \mathsf{SP}^0$ is not PCF_0^Ω-definable from $Y_{\mathtt{N}^l\to\mathtt{N}}$,
- the element $Y_{\mathtt{N}^{l+2}\to\mathtt{N}} \in \mathsf{SF}$ is not PCF_0^Ω-definable from $Y_{\mathtt{N}^l\to\mathtt{N}}$.

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 according to classical logic, at least one of these must be the case. (This is reminiscent of some well-known situations from complexity theory.) We expect that a more delicate analysis will allow us to fill this gap. One can also envisage an 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 [17, 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 [17, 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 some further reworking of our arguments.

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

$$\begin{aligned} \mathit{rec}_\sigma &: \sigma \to (\sigma \to \mathtt{N} \to \sigma) \to \mathtt{N} \to \sigma \;, \\ \mathit{iter}_{\sigma\tau} &: (\sigma \to (\sigma + \tau)) \to \sigma \to \tau \;. \end{aligned}$$

The idea behind the latter is to embody the behaviour of a *while* construct for imperative-style loops with state σ and exit type τ.[11]

It is shown in [17, Section 6.3] that each rec_σ may be interpreted by a *left-well-founded* procedure (cf. Subsection 2.5), and it is not hard to check that the same is true for each $\mathit{iter}_{\sigma\tau}$. Furthermore, it is clear that all left-well-founded procedures are non-spinal, so it requires only the addition of an easy base case in the proof of Theorem 28 to show that Y_{k+1} cannot be defined even in PCF_k^Ω extended with all primitive recursors and iterators. (Actually, we can dispense with the primitive recursors here, as it is straightforward to define them from suitable iterators.)

The dual question, roughly speaking, is whether any or all of the recursors rec_σ or iterators $\mathit{iter}_{\sigma\mathtt{N}}$ for types σ of level $k+1$ are definable in PCF_k. We conjecture that they are not, and that this could be shown by suitably choosing a substructure of SP^0 so as to exclude such $\mathit{iter}_{\sigma\mathtt{N}}$. (This would incidentally 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 denotable in $\mathcal{L}$? Or to look at it another way: given a type

[11]The sum type $\sigma + \tau$ is not officially part of our system, but can (for any given σ, τ) be represented as a retract of some existing simple 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 [17, 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. As in [17], we write $\mathsf{Klex}^{\mathrm{min}}$ of the language of *Kleene μ-recursion*: this is equivalent (in its power to denote elements of SF) to PCF_0 extended with a strict primitive recursor for ground type and a strict iterator for ground type, but with no form of general recursion.

- For first-order types σ, even $\mathsf{Klex}^{\mathrm{min}}$ suffices for defining all elements of $\mathsf{SF}^{\mathrm{eff}}(\sigma)$; likewise, the oracle version $\mathsf{Klex}^{\mathrm{min}^{\Omega}}$ suffices for $\mathsf{SF}(\sigma)$.
- For second-order types σ of the special form $(\mathtt{N} \to \mathtt{N})^r \to \mathtt{N}$, $\mathsf{Klex}^{\mathrm{min}^{\Omega}}$ still suffices for $\mathsf{SF}(\sigma)$; however, this result is non-constructive, and $\mathsf{Klex}^{\mathrm{min}}$ does not suffice for $\mathsf{SF}^{\mathrm{eff}}(\sigma)$. (We draw here on some recent work of Dag Normann [20].)
- For general second-order types, $\mathsf{Klex}^{\mathrm{min}^{\Omega}}$ 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.
- 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.

7.4 Other languages and models

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, where we consider expressibility modulo observational equivalence in Idealized Algol. This would constitute an interesting variation on 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 [19] that every effective element of Ct is represented by a PCF-*denotable* element of PC, and the proof actually shows that PCF_1 suffices here. (The further generalization of these ideas by Longley [15] makes some 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 [17, 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 45 *(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 (i) The cases $k = 0, 1$ are easy, so let us assume $k \geq 2$. 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 $\bot$.

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 $f'' \neq f$ so $f \prec f''$).

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 $f'' \neq f$ so $f \prec f''$).

(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 will 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

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

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] Hague, M., Murawski, A.S., Ong, C.-H.L., Serre, O.: Collapsible pushdown automata and recursion schemes. In: Proceedings of the 23th Annual IEEE Symposium on Logic in Computer Science, pp. 452-461. IEEE Computer Society (2008)

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

[9] Jones, N.D.: The expressive power of higher-order types, or, life without CONS. Journal of Functional Programming **11**, 5–94 (2001)

[10] Kartzow, A., Parys, P.: Strictness of the collapsible pushdown hierarchy. In: B. Rovan, V. Sassone and P. Widmayer (eds.), Mathematical Foundations of Computer Science 2012, pp. 566-577. Lecture Notes in Computer Science 7464, Springer (2012)

[11] 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)

[12] Kristiansen, L.: Higher types, finite domains and resource-bounded Turing machines. Journal of Logic and Computation **22(2)**, 281–304 (2012)

[13] Loader, R.: Finitary PCF is not decidable. Theoretical Computer Science **266(1-2)**, 341–364 (2001)

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

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

[16] Longley, J.R.: Bar recursion is not computable via iteration. Available from `arxiv.org/abs/1804.07277`; submitted for publication (2018). Earlier version as Informatics Research Report EDI-INF-RR-1420, University of Edinburgh (2015)

[17] Longley, J. and Normann, D.: Higher-Order Computability. Theory and Applications of Computability, Springer-Verlag (2015)

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

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

[20] Normann, D.: The sequential functionals of type $(\iota \to \iota)^n \to \iota$ form a dcpo for all $n \in \mathbb{N}$. CoRR, `abs/1607.02970` (2016)

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

[22] Ong, C.-H.L.: On model-checking trees generated by higher-order recursion schemes. In: Proceedings of the 21st Annual IEEE Symposium on Logic in Computer Science, pp. 81-90. IEEE Computer Society (2006)

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

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

[25] 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)

[26] 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).

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