The sequentially realizable functionals*

John Longley†

Abstract

We consider a notion of sequential functional of finite type, more generous than the familiar notion embodied in Plotkin's language PCF. We study both the "full" and "effective" partial type structures arising from this notion of sequentiality. The full type structure coincides with that given by the *strongly stable* model of Bucciarelli and Ehrhard; it has also been characterized by van Oosten in terms of realizability over a certain combinatory algebra.

We survey and relate several known characterizations of these type structures, and obtain some new ones. We show that (in both the full and effective scenarios) every finite type can be obtained as a retract of the pure type $\overline{2}$, and hence that all elements of the effective type structure are definable in PCF extended by a certain *universal* functional H. We also consider the relationship between our notion of sequentially computable functional and other known notions of higher-type computability.

Contents

*LFCS Report ECS-LFCS-98-402. Submitted to *Annals of Pure and Applied Logic.*

†LFCS, Division of Informatics, University of Edinburgh, The King's Buildings, Edinburgh EH9 3JZ. Email: `jrl@dcs.ed.ac.uk`.

1 Introduction

1.1 Background

The classic papers of Scott [51] and Plotkin [45] introduced the language for finite-type computable functionals now widely known as PCF. These authors gave a denotational semantics for this language using complete partial orders, and also pointed out a certain mismatch between the language and its denotational model: because of the "sequential" character of computations in PCF, there are elements of the model (such as "parallel or") that are in some sense computable but are not definable by terms of PCF.

These early papers initiated a search for other denotational models that more precisely clarified the nature of sequential computation, and in particular gave a semantic characterization of a fully abstract model of PCF. This generated an extensive body of research (see e.g. [6]), which culminated around 1993, when several solutions to the full abstraction problem were announced [2, 21, 40, 41].

The general problem of understanding sequentiality is of both conceptual and practical importance, given that most widely used deterministic programming languages are sequential in character. Much of the study of sequentiality to date has centred around PCF—indeed, it is sometimes implicitly assumed that "sequential" computability is synonymous with PCF-computability, at least for functionals of finite type. However, in their paper on game semantics for PCF [21], Hyland and Ong make the following penetrating remarks:[1]

> 'Persisting in the background of these developments is a deeper, more philosophical question of whether there is such a thing as a *canonical* notion of sequential computation at higher type. Clearly, the kind of computation *defined* by PCF is at least a contender for such a standard. But it seems to us that there is no compelling evidence (yet) that PCF-style computation is the only acceptable notion of higher-type sequentiality.'
>
> 'In fact it is unclear whether there are various inequivalent notions of higher-type sequentiality, all of them equally appealing; or whether ... there is just one notion under different guises.'

The purpose of the present paper is to investigate an alternative notion of higher-type sequentiality, more generous than the PCF one. We will show that this alternative notion is both mathematically compelling, in that it admits a large number of *prima facie* independent characterizations, and computationally appealing, in that it arises from a natural and intuitive concept of sequential algorithm.

The class of functionals that we consider is not new. Its discovery is due in essence to Bucciarelli and Ehrhard [9], who constructed a model of PCF involving a category of domains and *strongly stable* functions. Ehrhard later showed that the relevant part of this model could be more simply presented within the framework of *hypercoherences* [15]. The strongly stable model was originally conceived as a

[1] See [21], pages 18–19, 15.

line of attack on the PCF full abstraction problem, and as a step towards a more denotational understanding of certain features of the Berry-Curien *sequential algorithms* model [5]. However, it has since turned out to be of considerable interest in its own right. A crucial step was taken by Ehrhard [16], who showed that the notion of strong stability admits a computational interpretation: every morphism in the strongly stable model of PCF is "sequential" in the sense that it can be computed by a Berry-Curien sequential algorithm. This line of investigation was continued in [17], where Ehrhard showed that the strongly stable model is the *extensional collapse* of the sequential algorithms model. Next, in the autumn of 1996, van Oosten [43] discovered a combinatory algebra embodying an appealing notion of sequential computation, and showed that it yields a realizability model that coincides with the strongly stable model at the finite types. Exactly the same combinatory algebra was rediscovered independently by the present author early in 1997, which gave rise to the investigation reported in the present paper.

A simple example may be helpful at this point. Let $\Sigma = 1_\perp$ be the two-element poset with $\perp \sqsubseteq \top$, and let Σ^Σ be the set of monotone functions $\Sigma \to \Sigma$. Consider the function $F : \Sigma^\Sigma \to 2_\perp$ defined by

$$Fg \;=\; \begin{cases} \mathit{true} & \text{if } g\perp = \top \\ \mathit{false} & \text{if } g\perp = \perp \text{ but } g\top = \top \\ \perp & \text{otherwise.} \end{cases}$$

Intuitively, F tells us whether a function g is able to return a result without looking at its argument. The function F is strongly stable, and it is in some sense sequentially computable. Indeed, the reader familiar with Standard ML [38] (a sequential programming language!) may enjoy verifying that F can be implemented as a function of type `((unit->unit)->unit)->bool`, using either exceptions or references. Likewise, one can implement F using control features such the `call/cc` operator of Scheme [53], or the `catch` operator considered by Cartwright and Felleisen [12]. Although these implementations make internal use of non-functional features, in terms of their external behaviour they are perfectly "functional", in the sense that given equal inputs they yield equal outputs (provided these inputs are themselves functional). However, it is easily seen that F is *not* sequentially computable in the PCF sense, since it is not monotone with respect to the pointwise order. Thus F, although in some sense a functional program, cannot be implemented within the functional fragment of ML.

In this paper the higher-type functionals in question will be called the *sequentially realizable* (or SR) functionals, as distinguished from the *PCF-sequential* functionals. In some sense, the SR functionals include all PCF-sequential functionals, together with the above function F and "all things like it". Indeed, the SR functionals provide an answer to the question of how far one can travel in a language such as PCF+`catch` without sacrificing the functional nature (i.e. extensionality) of programs.

As with other classes of higher-type functionals, one may consider either the *full* (continuous) partial type structure of SR functionals, or the *effective* type structure that arises as its recursive analogue. These two finite type structures will be our main objects of study. The full type structure coincides exactly with that arising from Bucciarelli and Ehrhard's strongly stable model; however, for our present purposes we prefer the name *sequentially realizable* to *strongly stable*, since (a) we wish to emphasize the computational aspect of these functionals; and (b) the *effective* version of sequential realizability unfortunately does not coincide with the effective version of strong stability (this point will be explained in Section 5.3). The word *realizable* here should be understood in a rather loose informal sense: the "realizers" in question might be either intensional objects in a realizability model *à la* Hyland, sequential algorithms in the sense of Berry-Curien, strategies in a game model, or programs in a language such as PCF+catch or Standard ML.

1.2 Outline of the paper

The aims of the present paper are threefold. Firstly, we hope to provide evidence that the notion of SR functional is in some sense a fundamental one. To this end, we collect together a wide variety of characterizations of the (full and effective) SR functionals, surveying various known descriptions and obtaining some new ones. Secondly, we review some of the known results in a more conceptual light, and point out some of the abstract relationships between existing constructions. Thirdly, we study some intrinsic properties of the type structure of SR functionals: for example, that every finite type is a retract of the pure type $\overline{2}$, and that every element of the effective type structure is PCF-definable from a certain *universal* functional H.

The rest of the paper is arranged as follows. In Section 2 we introduce van Oosten's combinatory algebra $\mathcal{B}$, and its effective subalgebra $\mathcal{B}_{\mathit{eff}}$. We prove that these are indeed combinatory algebras, and obtain some basic properties of them. In Section 3 we construct realizability models from these combinatory algebras, in particular the categories of *modest sets* over $\mathcal{B}$ and $\mathcal{B}_{\mathit{eff}}$. These categories give rise to the full and effective SR functionals respectively, and will play a central role in the rest of the paper.

In Sections 4–6 we survey several other characterizations of the SR functionals, and examine the connections between them. In Section 4 we study the relationship between $\mathcal{B}$ and the *concrete data structures* (CDSs) considered by Berry and Curien. We show that $\mathcal{B}$ can be seen as a *universal* (sequential) CDS, and that a good category of CDSs and sequential functions embed fully in $\mathbf{Mod}(\mathcal{B})$. We also show that sequential *algorithms* are essentially equivalent to *realizers* in $\mathbf{Mod}(\mathcal{B})$. In Section 5 we study the hypercoherence model, and give a new proof of van Oosten's theorem that the partial type structure in $\mathbf{Mod}(\mathcal{B})$ coincides with that in Bucciarelli and Ehrhard's strongly stable model. Our proof exploits the connection with sequential algorithms, and invokes Ehrhard's result [17] relating sequential algorithms to the strongly stable model. In Section 6 we consider another characterization of the strongly stable model due to Colson and Ehrhard [13], and recast this as a presheaf construction of the SR functionals.

Section 7 contains the central new result of the paper. In the type structure of (full or effective) SR functionals, every finite type arises as a *retract* of the pure type $\overline{2}$. The key to this result is the construction of a retraction $\overline{3} \lhd \overline{2}$, which is inspired directly by the combinatory algebra $\mathcal{B}$. The next two sections present some applications of this result. In Section 8 we use it to give a fairly simple (and more self-contained) alternative proof of van Oosten's theorem, and to show how one can obtain *recursive types* as retracts of $\overline{2}$. In Section 9 we show that every effective SR functional is definable in the language PCF extended with a single *universal* functional H of type level 3. The language PCF+H gives us yet another handle on the SR functionals, yielding various new results. We also prove a technical fact: there is no universal SR functional of an essentially simpler type than that of H. Sections 7–9 are mostly independent of 4–6, except that Sections 8.1 and 9.4 refer to the material on hypercoherences.

We then point out some of the connections between our results and other ideas in semantics. In Section 10 we review our results from the standpoint of *synthetic domain theory*. In Section 11 we consider the relationship between sequential realizability and other possible notions of higher-type computability. We show that, in a suitable precise sense, there can be no "ultimate" notion of higher-type computability that subsumes all reasonable such notions. One may interpret this latter fact as a kind of "anti-Church's Thesis" for higher types.

We end the paper with a discussion of the significance of our results, and of the claim that they embody a natural notion of "sequential" higher-type functional. We also mention some avenues for further research, including the possible applications of the SR functionals to programming language design.

In the present paper we study the SR functionals mainly from a "denotational" point of view, touching only intermittently on the issue of how they might be implemented operationally. This latter question could cover both the design of particular syntactic systems (programming languages) for implementing the SR functionals, and the use of *games* as an intensional-semantical setting for modelling various operational paradigms. We will survey some of this territory briefly in Section 12.3; a more detailed study of the SR functionals from an operational perspective may appear in a subsequent paper.

1.3 Acknowledgements

I have benefited from discussions and correspondence with very many people, including Samson Abramsky, Thomas Ehrhard, Martín Escardó, Martin Hyland, Jim Laird, Hanno Nickau, Luke Ong, Jaap van Oosten, Gordon Plotkin, Alex Simpson and Thomas Streicher. I also thank Gail Kemp for moral support. This research was funded by the EPSRC Research Grants GR/L89532 "Notions of computability for general datatypes" and GR/J84205 "Frameworks for programming language semantics and logic".

2 A combinatory algebra for sequential computation

2.1 Construction of $\mathcal{B}$ and $\mathcal{B}_{eff}$.

We begin by describing the combinatory algebra $\mathcal{B}$ introduced by van Oosten in [43]. We will follow van Oosten's notation in certain respects.

Let N be the set of natural numbers (including 0), and let $\mathrm{N}_\perp = \mathrm{N} \sqcup \{\perp\}$. We will identify partial functions $\mathrm{N} \rightharpoonup \mathrm{N}$ with total functions $\mathrm{N} \to \mathrm{N}_\perp$; we write $\mathrm{N}_\perp^\mathrm{N}$ for the set of such functions.

As a first step, let us consider informally the possible behaviours of a "sequential" algorithm or strategy σ for computing a partial function $F : \mathrm{N}_\perp^\mathrm{N} \rightharpoonup \mathrm{N}$. Suppose σ is presented with a function $g : \mathrm{N} \to \mathrm{N}_\perp$ as an argument. There are three possibilities: Firstly, σ might simply diverge, in which case F is the everywhere undefined function. Secondly, σ might return a natural number straightaway, in which case F is a constant function. Thirdly, σ might ask for the value of g on some argument n, say. If $g(n)$ is undefined, then σ can do nothing; however, if $g(n)$ returns an answer m, then the subsequent behaviour of σ may be conditional on m. Indeed, for each value of m we have again the above three possibilities for the behaviour of σ, and so on.

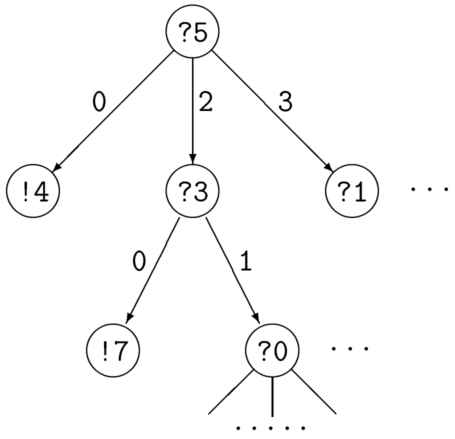
We may represent the behaviour of such a strategy σ as a (finite or infinite) *decision tree*. This consists of a set of nodes, each carrying a label which may be either a *question* $?n$ or an *answer* $!n$ ($n \in \mathrm{N}$). At each node, the children are indexed by (a subset of the) natural numbers. The way in which such a decision tree represents a strategy should be clear from Figure 1, which shows part of a decision tree together with pseudocode for the corresponding part of the strategy. (Decision trees of this kind will be very familiar to readers of e.g. [5, 12].)

Formally, we may identify the nodes of a decision tree with finite sequences of natural numbers: the root node is the empty sequence ϵ, and the child m of the node $[m_1, \ldots, m_k]$ (if it exists) is the node $[m_1, \ldots, m_k, m]$. (This is the notation we shall use for displaying finite sequences.) We write Seq(N) for the set of finite sequences of natural numbers (including ϵ), and use $\alpha, \beta, \ldots$ to range over Seq(N). (Note that Roman letters will stand for natural numbers and Greek letters for sequences.) We write $m; \alpha$ or $\alpha; m$ for the result of adjoining a new element at the beginning or end of a sequence, and $\alpha; \beta$ for concatenation of sequences.

Likewise, we may identify the labels in a decision tree with elements of $\mathrm{N}+\mathrm{N}$, where the left and right summands correspond to the tags ? and ! respectively. This leads us to the following definition:

Definition 2.1 *A* decision tree *is formally a partial function* $\sigma : \mathrm{Seq}(\mathrm{N}) \to \mathrm{N} + \mathrm{N}$. *The domain of* σ *is the set of* nodes *in the tree.*

One might ask whether one ought also to impose hygiene conditions on σ: for example, that the set of nodes of the tree is prefix-closed, and that labels $!n$ may appear only on leaves of the tree. In fact such conditions will not be necessary, although trees that do not satisfy them will contain "dead" nodes that will never be reached in any play of the strategy. Note also that the same question may




```
fn g =>
  case g(5) of
    0 => output 4
  | 1 => diverge
  | 2 => (case g(3) of
        0 => output 7
      | 1 => (case g(0) of ...)
        ...)
  | 3 => (case g(1) of
        _ => diverge)
    ...
```

Figure 1: Part of a decision tree for a sequential strategy

be asked twice on the same path through the tree, and this too may give rise to inaccessible nodes.

The key observation in the construction of $\mathcal{B}$ is that nodes and labels can themselves be coded as natural numbers—that is, both Seq(N) and $N+N$ admit injections into N. Let $\langle\cdots\rangle : \mathrm{Seq}(\mathrm{N}) \rightarrowtail \mathrm{N}$ be some effective coding for nodes: for example, take

$$\langle\epsilon\rangle = 0, \qquad \langle m_1, \ldots, m_k\rangle = 2^{m_1}3^{m_2}\ldots p_k^{m_k+1} - 1,$$

where $\langle m_1, \ldots, m_k\rangle$ abbreviates $\langle[m_1, \ldots, m_k]\rangle$. Likewise, let $[?,!] : \mathrm{N}+\mathrm{N} \rightarrowtail \mathrm{N}$ be some effective coding for labels, for example:

$$?(n) = 2n, \qquad !(n) = 2n+1.$$

In general, we will require simply that these encoding operations are effective. It is automatic that the images of these encodings are semidecidable subsets of N, and that the corresponding decoding operations are effective. (The particular codings given above are of course bijective, but this is not a necessary property and we will not assume it below.)

Using these codings, we can represent a decision tree $\sigma : \mathrm{Seq}(\mathrm{N}) \rightharpoonup \mathrm{N}+\mathrm{N}$ simply by a partial function $f : \mathrm{N} \rightharpoonup \mathrm{N}$ (i.e. by an element $f \in \mathrm{N}_\perp^{\mathrm{N}}$). Explicitly, we say f *represents* σ if for all $\alpha \in \mathrm{Seq}(\mathrm{N})$ and $l \in \mathrm{N}+\mathrm{N}$ we have

$$f(\langle\alpha\rangle) = [?,!](l) \quad \text{iff} \quad \sigma\alpha = l.$$

Clearly, every partial function $f \in \mathrm{N}_\perp^{\mathrm{N}}$ represents a unique decision tree σ_f, and for every decision tree σ there is a least partial function f_σ that represents it.

If f represents σ, a play of σ against a function g may be viewed as a "dialogue" between f and g. Thus, the procedure for playing a strategy against an argument gives rise to an operation $| : \mathrm{N}_\perp^{\mathrm{N}} \times \mathrm{N}_\perp^{\mathrm{N}} \to \mathrm{N}_\perp$. This operation is intuitively sequentially computable; the following informal ML-style recursive definition gives the idea. (A more formal definition in call-by-value PCF will be given in Section 9.1.)

fun *play* $f\ g\ \alpha\ =$
  case $f\langle\alpha\rangle$ of
    $!n \Rightarrow n$
    $|\ ?n \Rightarrow$ (case $g(n)$ of $m \Rightarrow$ *play* $f\ g\ (\alpha; m)$)
fun $|\ f\ g\ =$ *play* $f\ g\ \epsilon$

A minor modification of this gives us an operation $\bullet : \mathrm{N}_\perp^\mathrm{N} \times \mathrm{N}_\perp^\mathrm{N} \to \mathrm{N}_\perp^\mathrm{N}$: in effect, we use f to represent an infinite forest of decision trees rather than just a single one.

fun $\bullet\ f\ g\ =$ (fn $n \Rightarrow$ *play* $f\ g\ [n]$)

These ideas may be expressed more formally as follows:

Definition 2.2 *(i) Let play* : $\mathrm{N}_\perp^\mathrm{N} \times \mathrm{N}_\perp^\mathrm{N} \times \mathrm{Seq}(\mathrm{N}) \to \mathrm{N}_\perp$ *be the smallest partial function such that, for all* $f, g \in \mathrm{N}_\perp^\mathrm{N}, \alpha \in \mathrm{Seq}(\mathrm{N})$ *and* $n, m \in \mathrm{N}$,

- *if* $f\langle\alpha\rangle = !n$ *then* $play(f, g, \alpha) = n$,
- *if* $f\langle\alpha\rangle = ?n$ *and* $g(n) = m$ *then* $play(f, g, \alpha) = play(f, g, (\alpha; m))$.

(ii) Let $|, \bullet$ *be the operations defined by*

$$f \mid g \ = \ play(f, g, \epsilon), \qquad f \bullet g \ = \ \lambda n.play(f, g, [n]).$$

Let $\mathcal{B}$ *be the applicative structure* $(\mathrm{N}_\perp^\mathrm{N}, \bullet)$.

Clearly the function *play* can be constructed by an iteration up to ω. As usual, we take $\bullet$ to be left-associative in expressions such as $f \bullet g \bullet h$. By abuse of notation, we sometimes write $\mathcal{B}$ for the underlying set $\mathrm{N}_\perp^\mathrm{N}$.

As is observed in [43], the construction of $\mathcal{B}$ is reminiscent of Kleene's second model K_2, the partial combinatory algebra for "function realizability" (see [25]). In both cases, the elements are functions from N to N, and at each stage in the application of f to g, f either returns a result or requests further information about g. However, $\mathcal{B}$ differs from K_2 in two respects: firstly, we use partial rather than total functions on N; and secondly, at each stage f is allowed to specify the *particular* piece of further information about g that it would like to see.

We will write $\sqsubseteq$ for the pointwise ordering on $\mathcal{B}$. It is clear that $(\mathcal{B}, \sqsubseteq)$ is a CPO and that the functions *play*, $|$ and $\bullet$ are monotone and continuous.

It is easy to check that if $f, g \in \mathrm{N}_\perp^\mathrm{N}$ are *partial recursive* then so is $f \bullet g$. This justifies the following definition:

Definition 2.3 *Let* $\mathrm{N}_{\perp\mathit{eff}}^\mathrm{N}$ *be the subset of* $\mathrm{N}_\perp^\mathrm{N}$ *consisting of the partial recursive functions, and let* $\mathcal{B}_\mathit{eff}$ *be the applicative structure* $(\mathrm{N}_{\perp\mathit{eff}}^\mathrm{N}, \bullet)$. *The elements of* $\mathcal{B}_\mathit{eff}$ *are called* effective *elements of* $\mathcal{B}$.

2.2 Combinatory completeness

The main result of this section is that $\mathcal{B}$ and $\mathcal{B}_{\mathit{eff}}$ are both combinatory algebras: in fact, there are elements $\mathbf{k}, \mathbf{s} \in \mathcal{B}_{\mathit{eff}}$ such that for all $x, y, z \in \mathcal{B}$ we have

$$\mathbf{k} \bullet x \bullet y \ = \ x, \qquad \mathbf{s} \bullet x \bullet y \bullet z \ = \ (x \bullet z) \bullet (y \bullet z).$$

(The fact that $\mathcal{B}$ is a combinatory algebra was stated without proof in [43].)

Although the definition of application in $\mathcal{B}$ is fairly simple, the proof that it is a combinatory algebra seems to require a significant effort, however it is tackled. This is perhaps surprising: intuitively it seems fairly clear that for both $\mathbf{k}$ and $\mathbf{s}$ there are sequential algorithms that mediate the appropriate interactions between x, y, z. However, a direct definition of $\mathbf{s}$ in particular would be very cumbersome, and it seems more illuminating to obtain $\mathbf{k}, \mathbf{s}$ indirectly via Curry's *combinatory completeness* property. We use the following standard notions from the theory of combinatory algebras (cf. [3, chapter 5]).

Definition 2.4 *(i) Let V be an infinite supply of formal* variables*, and suppose we also have a formal* constant *c for each element $c \in \mathcal{B}$. The* formal expressions *over $\mathcal{B}$ are freely constructed from variables and constants via formal juxtaposition (a binary operation). If $x_0, \ldots, x_r$ are distinct variables, we write $x_0, \ldots, x_r \vdash e$ to mean that e is a formal expression whose variables are among $x_0, \ldots, x_r$.*

(ii) A valuation *ν for $x_0, \ldots, x_r$ associates to each x_i an element $a_i = \nu(x_i)$ of $\mathcal{B}$. Formally, a valuation is an ordered list $(x_0 \mapsto a_0, \ldots, x_r \mapsto a_r)$. If $x_0, \ldots, x_r \vdash e$ and ν is a valuation for $x_0, \ldots, x_r$, we define the* interpretation *$[\![e]\!]_\nu$ of e relative to ν inductively as follows:*

- *If c is a constant then $[\![c]\!]_\nu = c$.*
- *If x is a variable then $[\![x]\!]_\nu = \nu(x)$.*
- *If $e_1 e_2$ is a juxtaposition then $[\![e_1 e_2]\!]_\nu = [\![e_1]\!]_\nu \bullet [\![e_2]\!]_\nu$.*

Our goal is to show the following: if $x_0, \ldots, x_r \vdash e$ then there exists $f \in \mathcal{B}$ such that for all valuations $\nu = (x_0 \mapsto a_0, \ldots, x_r \mapsto a_r)$ we have $f \bullet a_0 \bullet \cdots \bullet a_r = [\![e]\!]_\nu$. As a first step, we prove an "uncurried" version of this, in which f takes the arguments $a_0, \ldots, a_r$ all together rather than separately. As a means of lumping arguments together, let us say that an element $b \in \mathcal{B}$ *represents* a valuation $(x_0 \mapsto a_0, \ldots, x_r \mapsto a_r)$ if $b\langle i, n\rangle = a_i(n)$ whenever $0 \leq i \leq r$ and $n \in N$.

Proposition 2.5 *Suppose $\vec{x} \vdash e$, where $\vec{x}$ abbreviates $x_0, \ldots, x_r$. Then there is a (canonical) element $(\Lambda\vec{x}.e) \in \mathcal{B}$ such that whenever ν is a valuation for $\vec{x}$ and b represents ν, we have $(\Lambda\vec{x}.e) \bullet b = [\![e]\!]_\nu$. Moreover, if all the constants occurring in e are effective then so is $(\Lambda\vec{x}.e)$.*

PROOF By induction on the structure of e.

- For *constants* $c \in \mathcal{B}$, let $(\Lambda\vec{x}.c)$ be the (least) partial function f such that for all n, m, $c(n) = m$ implies $f\langle n\rangle =!m$. Clearly $(\Lambda\vec{x}.c) \bullet b = c$ for all $b \in \mathcal{B}$. Moreover, if c is effective then so is $(\Lambda\vec{x}.c)$.

- For *variables* x_i, let $(\Lambda\vec{x}.x_i)$ be the partial function f given by

$$f\langle n\rangle = ?\langle i,n\rangle, \qquad f\langle n,m\rangle = !m.$$

  Clearly, if b represents $(x_0 \mapsto a_0, \ldots, x_r \mapsto a_r)$ then $(\Lambda\vec{x}.x_i) \bullet b = a_i$. Moreover, $(\Lambda\vec{x}.x_i)$ is always effective.

- For *juxtapositions* e_1e_2, suppose $f_1 = (\Lambda\vec{x}.e_1)$ and $f_2 = (\Lambda\vec{x}.e_2)$. We will construct $f \in \mathcal{B}$ such that for all b we have $f \bullet b = (f_1 \bullet b) \bullet (f_2 \bullet b)$. Let $play_1 : \mathrm{N} \times \mathrm{Seq}(\mathrm{N}) \to \mathrm{N}_\perp$ and $play_2 : \mathrm{N} \times \mathrm{N} \times \mathrm{Seq}(\mathrm{N}) \to \mathrm{N}_\perp$ be the smallest partial functions satisfying the following conditions for all $m, p, q, \alpha, \beta, \gamma$:

  – If $f_1\langle\alpha\rangle = ?q$ then $play_1(\langle\alpha\rangle, \epsilon) = ?q$
  and $play_1(\langle\alpha\rangle, (m,\gamma)) = play_1(\langle\alpha, m\rangle, \gamma)$.

  – If $f_1\langle\alpha\rangle = !(!p)$ then $play_1(\langle\alpha\rangle, \epsilon) = !p$.

  – If $f_1\langle\alpha\rangle = !(?q)$ then $play_1(\langle\alpha\rangle, \gamma) = play_2(\langle\alpha\rangle, \langle q\rangle, \gamma)$.

  – If $f_2\langle\beta\rangle = ?q$ then $play_2(\langle\alpha\rangle, \langle\beta\rangle, \epsilon) = ?q$
  and $play_2(\langle\alpha\rangle, \langle\beta\rangle, (m,\gamma)) = play_2(\langle\alpha\rangle, \langle\beta, m\rangle, \gamma)$.

  – If $f_2\langle\beta\rangle = !p$ then $play_2(\langle\alpha\rangle, \langle\beta\rangle, \gamma) = play_1(\langle\alpha, p\rangle, \gamma)$.

  (Clearly $play_1$, $play_2$ can be constructed by simultaneous iteration.) Now define f by $f\langle n;\alpha\rangle = play_1(\langle n\rangle, \alpha)$. By some tedious inductions, it is straightforward to verify that f has the desired property.

  Set $(\Lambda\vec{x}.e_1e_2) = f$; then $(\Lambda\vec{x}.e_1e_2) \bullet b = [\![\, e_1e_2 \,]\!]_\nu$ provided both $(\Lambda\vec{x}.e_1) \bullet b = [\![\, e_1 \,]\!]_\nu$ and $(\Lambda\vec{x}.e_2) \bullet b = [\![\, e_2 \,]\!]_\nu$. Moreover, it is easy to see that if both $(\Lambda\vec{x}.e_1)$ and $(\Lambda\vec{x}.e_2)$ are effective then so is $(\Lambda\vec{x}.e_1e_2)$. $\Box$

The next step is to show that $(\Lambda\vec{x}.e)$ can be transformed into a representation of e that takes its arguments one at a time. Some further notation will be helpful. If $b, c \in \mathcal{B}$ and $i \in \mathrm{N}$, define $b[c/x_i] \in \mathcal{B}$ as follows: if $m = \langle i, n\rangle$ for some n then $b[c/x_i](m) = c(n)$; otherwise $b[c/x_i](m) = b(m)$. Note that if b represents $(x_0 \mapsto a_0, \ldots, x_r \mapsto a_r)$ and $0 \le i \le r$, then $b[c/x_i]$ represents $(x_0 \mapsto a_0, \ldots, x_i \mapsto c, \ldots, x_r \mapsto a_r)$.

Proposition 2.6 *Given any $f \in \mathcal{B}$ and $i \in \mathrm{N}$, there is a (canonical) element $curry_i(f) \in \mathcal{B}$ such that for all $b, c \in \mathcal{B}$ we have $curry_i(f) \bullet b \bullet c = f \bullet b[c/x_i]$. Moreover, if f is effective then so is $curry_i(f)$.*

Proof Let $play' : \mathrm{N} \times \mathrm{Seq}(\mathrm{N}) \times \mathrm{Seq}(\mathrm{N}) \to \mathrm{N}_\perp$ be the smallest partial function satisfying the following conditions for all $m, n, p, q, \alpha, \beta, \gamma$:

– If $f\langle\alpha\rangle = !p$ then $play'(\langle\alpha\rangle, \epsilon, \epsilon) = !(!p)$.

– If $f\langle\alpha\rangle = ?\langle i, n\rangle$ then $play'(\langle\alpha\rangle, \epsilon, \epsilon) = !(?n)$
and $play'(\langle\alpha\rangle, (m,\beta), \gamma) = play'(\langle\alpha, m\rangle, \beta, \gamma)$.

– If $f\langle\alpha\rangle = ?q$ where $\neg\exists n.\, q = \langle i, n\rangle$, then $play'(\langle\alpha\rangle, \beta, \epsilon) = ?q$
and $play'(\langle\alpha\rangle, \beta, (m,\gamma)) = play'(\langle\alpha, m\rangle, \beta, \gamma)$.

Now define $curry_i(f)$ by $curry_i(f)\langle\langle n,\beta\rangle,\gamma\rangle = play'(\langle n\rangle,\beta,\gamma)$. It is straightforward to show that $curry_i(f)$ has the required property, and the effectivity condition is obvious. $\Box$

It now follows readily that $\mathcal{B}$, $\mathcal{B}_{\mathit{eff}}$ are combinatory algebras.

Theorem 2.7 (Combinatory completeness) *(i) Suppose that $x_0,\ldots,x_r \vdash e$. Then there is an element $(\lambda^*\vec{x}.e) \in \mathcal{B}$ such that for all valuations $\nu = (\vec{x} \mapsto \vec{a})$ we have $(\lambda^*\vec{x}.e) \bullet a_0 \bullet \cdots \bullet a_r = [\![e]\!]_\nu$. Moreover, if all constants appearing in e are effective then so is $(\lambda^*\vec{x}.e)$.*

(ii) There exist $\mathbf{k},\mathbf{s} \in \mathcal{B}_{\mathit{eff}}$ such that for all $x,y,z \in \mathcal{B}$ we have $\mathbf{k} \bullet x \bullet y = x$, $\mathbf{s} \bullet x \bullet y \bullet z = (x \bullet z) \bullet (y \bullet z)$.

PROOF (i) Define $(\lambda^*\vec{x}.e) = (curry_0(\ldots(curry_r(\Lambda\vec{x}.e))\ldots)) \bullet (\lambda n.\bot)$. Note that for any $\nu = (\vec{x} \mapsto \vec{a})$, the element $(\lambda n.\bot)[a_0/x_0]\ldots[a_r/x_r]$ represents ν. So by Propositions 2.6 and 2.5 we have

$$(\lambda^*\vec{x}.e) \bullet a_0 \bullet \cdots \bullet a_r \;=\; (\Lambda\vec{x}.e) \bullet (\lambda n.\bot)[a_0/x_0]\ldots[a_r/x_r] \;=\; [\![e]\!]_\nu.$$

The effectivity condition follows from those in Propositions 2.6 and 2.5.

(ii) Take $\mathbf{k} = (\lambda^* xy.x)$, $\mathbf{s} = (\lambda^* xyz.(xz)(yz))$. $\Box$

Remarks 2.8 (i) The construction of $\mathcal{B}$ can be carried out more generally starting from any infinite set X equipped with injective coding functions $\mathrm{Seq}(X) \rightarrowtail X$ and $X + X \rightarrowtail X$.

(ii) A combinatory algebra $\mathcal{A}$ closely related to $\mathcal{B}$ has recently been constructed by Samson Abramsky, using ideas from Girard's Geometry of Interaction and the history-free games model of [2]. The construction of $\mathcal{A}$ makes use of coding operations $[L,R] : \mathrm{N} + \mathrm{N} \rightarrowtail \mathrm{N}$ and $pair : \mathrm{N} \times \mathrm{N} \to \mathrm{N}$. The underlying set of $\mathcal{A}$ is $\mathrm{N}_\bot^{\mathrm{N}}$, and application is defined by the formula

$$f \bullet g \;=\; L^{-1} \circ \bigcup_{r \geq 0} h^r \circ f \circ L,$$

where $h = f \circ R \circ !g \circ R^{-1}$ and $!g \;=\; pair \circ (\mathrm{id} \times g) \circ pair^{-1}$. Further details and motivation are given in [32].

It turns out that $\mathcal{A}$ and $\mathcal{B}$ are *equivalent* in the sense of [31]. More particularly, they have the same underlying set, and application in each is representable in the other. It follows that $\mathcal{A},\mathcal{B}$ give rise to exactly the same category of modest sets (see Section 3). However, the construction of $\mathcal{A}$ seems to be of independent interest for several reasons—for instance, subject to some conditions on the coding operations, one can identify a subalgebra $\mathcal{A}_{wb}$ of *well-bracketed* strategies, which yields a realizability model embodying the PCF notion of sequentiality (see [32]).

We will see in Section 4 that $\mathcal{B}$ and $\mathcal{B}_{\mathit{eff}}$ are λ-*algebras*, though they are not λ-*models*. An interesting question, suggested to us by Luke Ong, is whether one can find a good syntactic calculus (such as an untyped λ-calculus with control features or a suitable extension of combinatory logic) in which all the effective elements of $\mathcal{B}$ are definable.

In the remainder of this section we establish some technical facts about $\mathcal{B}$ and $\mathcal{B}_{\mathit{eff}}$ which give further insight into their nature.

2.3 Irredundant realizers

We have already mentioned that our decision trees may contain redundancy in that the same question may be asked more than once along a single path, and may also contain inaccessible nodes for other reasons. We show here that any decision tree may be "pruned" to yield an equivalent tree not suffering from any of these kinds of redundancy, and that moreover this pruning operation may be performed within $\mathcal{B}$ itself.

In fact we will consider two notions of irredundant element, according to whether we are thinking of single decision trees or infinite forests.

Definition 2.9 (Irredundant elements) *Let f be an element of $\mathcal{B}$.*

(i) f is $|$-irredundant *if, for all $\alpha, \beta \in \mathrm{Seq}(\mathrm{N})$,*

- *if $f\langle\alpha\rangle$ is defined then it is a question or an answer;*
- *if $f\langle\alpha\rangle$ is defined and β is a proper prefix of α then $f\langle\beta\rangle$ is a question;*
- *if $f\langle\alpha\rangle$ is defined and β is a proper prefix of α then $f\langle\beta\rangle \neq f\langle\alpha\rangle$.*

(ii) f is $\bullet$-irredundant *if $f\langle\epsilon\rangle = \bot$ and f_n is $|$-irredundant for each n, where $f_n\langle\alpha\rangle = f\langle n;\alpha\rangle$.*

The important facts about irredundant elements are given by the following:

Proposition 2.10 *(i) To each $f \in \mathcal{B}$ we may associate a $|$-irredundant element $irr_|(f)$ such that $irr_|(f) \mid g = f \mid g$ for all g, and if f is already $|$-irredundant then $irr_|(f) = f$. Moreover, there is an element $\mathbf{irr}_| \in \mathcal{B}_{\mathit{eff}}$ such that $\mathbf{irr}_| \bullet f = irr_|(f)$ for all f.*

(ii) Likewise, to each $f \in \mathcal{B}$ we may associate a $\bullet$-irredundant $irr_\bullet(f)$ such that $irr_\bullet(g) \bullet g = f \bullet g$ for all g, and if f is $\bullet$-irredundant then $irr_\bullet(f) = f$. Moreover, there is an element $\mathbf{irr}_\bullet \in \mathcal{B}_{\mathit{eff}}$ such that $\mathbf{irr}_\bullet \bullet f = irr_\bullet(f)$ for all f.

Proof (i) The informal idea behind the construction of $irr_|(f)$ from f is clear: we simply drop all inaccessible nodes of f, and omit all repetitions of previously asked questions, knowing that they must receive the same answer as before. Thus, to find out what $irr_|(f)\langle m_1,\ldots,m_l\rangle$ should be, we may engage in a dialogue with f, using $m_1,\ldots,m_l$ as a "script" giving the answers to the first l distinct questions asked by f. Formally, we define $irr_|(f)$ to be the smallest partial function f_0 such that $f_0\langle m_1,\ldots,m_l\rangle = f\langle t_1,\ldots,t_j\rangle$ whenever the following conditions hold for some $q_1,\ldots,q_j,r_1,\ldots,r_l$:

- $f\langle t_1,\ldots,t_{i-1}\rangle = ?q_i$ for $1 \leq i \leq j$;
- $r_1,\ldots,r_l$ are the distinct elements of the sequence $q_1,\ldots,q_j$ in order of appearance, and whenever $q_i = r_h$ we have $t_i = m_h$;
- $f\langle t_1,\ldots,t_j\rangle$ is of the form $!p$ or $?q$ where $q \notin \{q_1,\ldots,q_j\}$.

It is routine to verify that $irr_|(f) \mid g = n$ iff $f \mid g = n$, that $irr_|(f)$ is $|$-irredundant, and that if f is $|$-irredundant then $irr_|(f) = f$.

It remains to show that there exists $\mathbf{irr}_| \in \mathcal{B}_{\mathit{eff}}$ that computes $irr_|(f)$ from f. Formally, let $\mathbf{irr}_|$ be the least partial function satisfying the following for all $\alpha = m_1,\ldots,m_l$ and β:

- If $\beta = ?q_1, \ldots, ?q_j$ and $r_1, \ldots, r_k$ are the distinct elements of $q_1, \ldots, q_j$ in order of appearance, where $k \leq l$, then $\mathbf{irr}_{|}\langle\langle\alpha\rangle, \beta\rangle = ?\langle m_{h(1)}, \ldots, m_{h(j)}\rangle$, where each $h(i)$ is determined by $q_i = r_{h(i)}$.
- If $\beta = ?q_1, \ldots, ?q_j$ contains exactly l distinct questions and u is of the form $!p$ or $?q$ where $q \notin \{q_1, \ldots, q_j\}$, then $\mathbf{irr}_{|}\langle\langle\alpha\rangle, \beta, u\rangle = !u$.

Clearly $\mathbf{irr}_{|}$ is effective, and it is easy to verify that $\mathbf{irr}_{|} \bullet f = irr_{|}(f)$ for any f.

(ii) We may now define $\mathbf{irr}_{\bullet} \in \mathcal{B}_{\mathit{eff}}$ as the least partial function such that

- if $\mathbf{irr}_{|}\langle\langle\alpha\rangle, \beta\rangle = ?\langle\gamma\rangle$ then $\mathbf{irr}_{\bullet}\langle n, \alpha, \beta\rangle = ?\langle n, \gamma\rangle$;
- if $\mathbf{irr}_{|}\langle\langle\alpha\rangle, \beta\rangle = !u$, then $\mathbf{irr}_{\bullet}\langle n, \alpha, \beta\rangle = !u$,

and take $irr_{\bullet}(f) = \mathbf{irr}_{\bullet} \bullet f$. It is easy to see that $irr_{\bullet}(f)$ has the required properties. □

There is an analogy here with the Scott graph model $\mathcal{P}\omega$ [50]. In both $\mathcal{B}$ and $\mathcal{P}\omega$ we may identify a class of "canonical" realizers we may associate to each element x a canonical element that represents the same endofunction, and moreover this canonical element can be computed from x within the combinatory algebra itself. However, there is an important difference: in $\mathcal{P}\omega$ there is a unique canonical element representing any given continuous function (namely its *graph*), but in $\mathcal{B}$ there are usually many irredundant elements that represent a given sequential function, corresponding to different sequential *algorithms* (cf. Section 4).

2.4 A fixed point combinator

In any combinatory algebra $(A, \cdot)$, we say that an element $\mathbf{y}$ is a *fixed point combinator* if $\mathbf{y} \cdot x = x \cdot (\mathbf{y} \cdot x)$ for every x. It is well known that every combinatory algebra has a fixed point combinator: for instance, we may define one by Curry's formula

$$\mathbf{y} \;=\; (\lambda^* xy.y(xxy))(\lambda^* xy.y(xxy)).$$

On the other hand, for every $x \in \mathcal{B}$ we know that the function $\alpha_x : y \mapsto x \bullet y$ is continuous with respect to the CPO structure of $\mathcal{B}$, so it has a *least* fixed point.

Perhaps surprisingly, there are fixed point combinators in $\mathcal{B}$ that do not always compute the least fixed point. Our aim here is to show that there is one that does. The combinator defined above would probably suffice, but it is easier to show the required property for a more concretely defined element $\mathbf{y}$.

Lemma 2.11 *There is an element $\mathbf{y} \in \mathcal{B}_{\mathit{eff}}$ such that for all $x \in \mathcal{B}$, $\mathbf{y} \bullet x$ is the least fixed point of α_x.*

Proof Let $\mathbf{i} = \mathbf{s} \bullet \mathbf{k} \bullet \mathbf{k}$, so that $\mathbf{i} \bullet x = x$ for all x. Let $\mathbf{y}_0$ be the element $\perp \in \mathcal{B}$ (i.e. the function $\lambda m.\perp$), and define inductively $\mathbf{y}_{k+1} = \mathbf{s} \bullet \mathbf{i} \bullet \mathbf{y}_k$. Since $\mathbf{y}_0 \subseteq \mathbf{y}_1$, by induction we have $\mathbf{y}_k \subseteq \mathbf{y}_k + 1$ for every k. So take $\mathbf{y} = \bigsqcup \mathbf{y}_k$. Since the $\mathbf{y}_k$ are effective uniformly in k, clearly $\mathbf{y}$ is effective.

Now take any $x \in \mathcal{B}$. We have $\mathbf{y}_0 \bullet x = \perp$, and so by induction we have $\mathbf{y}_k \bullet x = \alpha_x^k(\perp)$ for each k. Since application is continuous, we have $\mathbf{y} \bullet x = \bigsqcup \alpha_x^k(\perp)$, and this is the least fixed point of α_x. □

It follows immediately that if $x \in \mathcal{B}_{\mathit{eff}}$ then the least fixed point of α_x is also an element of $\mathcal{B}_{\mathit{eff}}$. For the rest of the paper, $\mathbf{y}$ will denote the element of $\mathcal{B}_{\mathit{eff}}$ given by the above proof.

3 Realizability models

3.1 Realizability models over $\mathcal{B}$ and $\mathcal{B}_{\mathit{eff}}$

We now review the construction of the standard realizability models corresponding to $\mathcal{B}$ and $\mathcal{B}_{\mathit{eff}}$. We will focus almost entirely on the categories of *modest sets* over these combinatory algebras; these are full subcategories of the corresponding *realizability toposes*, but in this paper we shall only occasionally refer to the latter. We will use the notation and terminology of [31], where further information on realizability models may be found.

The following definition makes sense for any combinatory algebra $(A, \cdot)$:

Definition 3.1 (Modest sets) *(i) A* modest set *X over A consists of a set $|X|$ (called the* underlying set *of X) together with a function assigning to each $x \in |X|$ a non-empty set $\|x\| \subseteq A$ (called the set of* realizers *for x), such that*

$$a \in \|x\| \wedge a \in \|x'\| \implies x = x'.$$

We sometimes write $\|x\|_X$ for $\|x\|$ to avoid ambiguity.

(ii) Suppose X, Y are two modest sets over A. A function $f : |X| \to |Y|$ is said to be tracked *by $r \in A$ if for all $x \in |X|$ and $a \in \|x\|_X$ we have $r \cdot a \in \|fx\|_Y$. A* morphism *$f : X \to Y$ is a function $f : |X| \to |Y|$ that is tracked by some $r \in A$. We write $\mathbf{Mod}(A)$ for the category of modest sets over A and morphisms between them.*

Remarks 3.2 (i) The fact that $\mathbf{Mod}(A)$ is a category follows easily from the combinatory completeness of $(A, \cdot)$. It is easy to see that $\mathbf{Mod}(A)$ is equivalent to the category $\mathbf{PER}(A)$ of *partial equivalence relations* on A.

(ii) Typically we are only interested in properties of modest sets up to isomorphism, and we will often switch freely between different representatives of the same isomorphism class, using whichever is most convenient. We may think of isomorphic modest sets as different *presentations* of essentially the same object.

One can think of a modest set X as a "datatype", where for each value $x \in |X|$ we have a set $\|x\|$ of "intensional representations" of x. The category of modest sets turns out to have very good properties (for more details see [31]):

Proposition 3.3 *The category $\mathbf{Mod}(A)$ is cartesian closed and regular, and it has finite sums and a natural number object.* $\Box$

We recall the construction of exponentials in $\mathbf{Mod}(A)$. If X and Y are modest sets, then Y^X is defined as follows: $|Y^X|$ is the set of morphisms $f : X \to Y$, and $\|f\|_{Y^X}$ is the set of elements $r \in A$ that track f. It is easy to check that Y^X

is indeed a modest set. The evaluation morphism $Y^X \times X \to Y$ is the obvious function $|Y^X| \times |X| \to |Y|$.

We now identify some important structure in $\mathbf{Mod}(\mathcal{B})$ and $\mathbf{Mod}(\mathcal{B}_{\mathit{eff}})$. (Note at once that $\mathbf{Mod}(\mathcal{B}_{\mathit{eff}})$ is a non-full subcategory of $\mathbf{Mod}(\mathcal{B})$.) For the terminal object 1 in both categories, we may take $|1| = \{*\}$, $\| * \| = \{\lambda m.\bot\}$. For the natural number object N, we take $|N| = \mathrm{N}$, $\|n\| = \{\overline{n}\}$, where $\overline{(n)} : 0 \mapsto n,\ m+1 \mapsto \bot$. It is easy to check that N (equipped with the obvious zero and successor morphisms) is indeed a natural number object in both $\mathbf{Mod}(\mathcal{B})$ and $\mathbf{Mod}(\mathcal{B}_{\mathit{eff}})$.

Next we define a *lift* operation $-_\bot$ on objects of $\mathbf{Mod}(\mathcal{B})$. Let $\mathit{up} : \mathcal{B} \to \mathcal{B}$ be the function defined by

$$\mathit{up}(r)(0) \;=\; 0, \qquad \mathit{up}(r)(n+1) \;=\; r(n),$$

and for any object X, let $X_\bot$ be the object given by

$$|X_\bot| \;=\; |X| \sqcup \{\bot\}, \qquad \|x\|_{X_\bot} \;=\; \{\mathit{up}(r) \mid r \in \|x\|_X\}, \qquad \|\bot\|_{X_\bot} \;=\; \{\lambda n.\bot\}.$$

The operation $-_\bot$ clearly extends to a functor, and indeed a monad, on $\mathbf{Mod}(\mathcal{B})$, which restricts well to a monad on $\mathbf{Mod}(\mathcal{B})$. In the case of $N_\bot$, we will usually work with the following simpler presentation: $|N_\bot| = \mathrm{N}_\bot$, $\|x\| = \{\overline{x}\}$ (where $\overline{\bot} = \lambda m.\bot$). It is easy to check that this is isomorphic to the object given by the definition of $-_\bot$.

In both categories, the object $N_\bot^N$ has a particularly simple presentation. This will be very useful in Section 7 when we consider call-by-value PCF and various extensions of it.

Proposition 3.4 *(i) Let B be the "object of realizers" in $\mathbf{Mod}(\mathcal{B})$, defined by $|B| = \mathrm{N}_\bot^{\mathrm{N}}$, $\|f\| = \{f\}$. Then $B \cong N_\bot^N$ in $\mathbf{Mod}(\mathcal{B})$.*

(ii) Let B_{eff} be the object of realizers in $\mathbf{Mod}(\mathcal{B}_{\mathit{eff}})$: $|B_{\mathit{eff}}| = \mathrm{N}_{\bot\,\mathit{eff}}^{\mathrm{N}}$, $\|f\| = \{f\}$. Then $B_{\mathit{eff}} \cong N_\bot^N$ in $\mathbf{Mod}(\mathcal{B}_{\mathit{eff}})$. $\square$

Note that the operation $\bullet$ is representable in $\mathcal{B}_{\mathit{eff}}$ by the element $\lambda^* xy.xy$. Hence it is easy to see that the operation $\mid$ is also representable in $\mathcal{B}_{\mathit{eff}}$ in the following sense: there is an element $\mathbf{bar} \in \mathcal{B}_{\mathit{eff}}$ such that for all $f, g \in \mathcal{B}$ and $x \in \mathrm{N}_\bot$, $\mathbf{bar} \bullet f \bullet g = \overline{x}$ iff $f \mid g = x$. It follows that in both $\mathbf{Mod}(\mathcal{B})$ and $\mathbf{Mod}(\mathcal{B}_{\mathit{eff}})$ there are morphisms $\mathit{dot} : N_\bot^N \times N_\bot^N \to N_\bot^N$ and $\mathit{bar} : N_\bot^N \times N_\bot^N \to N_\bot$ corresponding to $\bullet$ and $\mid$ respectively.

3.2 The SR functionals

We are now ready to interpret the simple types in $\mathbf{Mod}(\mathcal{B})$. This will lead to our definition of the sequentially realizable functionals, which are the primary objects of study in this paper.

Definition 3.5 (Simple types) *(i) The* simple types *(or finite types) are freely generated from a single ground type $\overline{0}$ via the binary type constructor $\to$:*

$$\sigma \;:=\; \overline{0} \;\mid\; \sigma_1 \to \sigma_2.$$

The pure types $\overline{0}, \overline{1}, \overline{2}, \ldots$ *are defined by* $\overline{k+1} = \overline{k} \to \overline{0}$.

(ii) For each type σ*, its* interpretation *in* $\mathbf{Mod}(\mathcal{B})$ *is the object* $[\![\,\sigma\,]\!]$ *given by*

$$[\![\,\overline{0}\,]\!] = N_\perp, \qquad [\![\,\sigma_1 \to \sigma_2\,]\!] = [\![\,\sigma_2\,]\!]^{[\![\,\sigma_1\,]\!]}.$$

The interpretation $[\![\,\sigma\,]\!]_{\mathit{eff}}$ *in* $\mathbf{Mod}(\mathcal{B}_{\mathit{eff}})$ *is defined analogously.*

Note that the above gives the "call-by-name" interpretation of the simple types, i.e. the one used for modelling the call-by-name version of PCF, as in [45]. We will content ourselves with this for the time being, but in Section 7 we shall also consider the call-by-value interpretation.

The sequentially realizable functionals themselves are introduced via a general notion of (abstract) type structure. For now, we choose a definition of type structure that matches our call-by-name interpretation of types.

Definition 3.6 *(i) A* (call-by-name, partial) type structure T *consists of the following:*

- *a set* T^σ *for each type* σ*, where* $T^{\overline{0}} = \mathrm{N}_\perp$*,*
- *for each* σ, τ *a total "application" function* $\cdot_{\sigma\tau} : T^{\sigma\to\tau} \times T^\sigma \to T^\tau$*.*

(ii) The type structures R, R_{eff} *are defined by* $R^\sigma = |[\![\,\sigma\,]\!]|$, $R^\sigma_{\mathit{eff}} = |[\![\,\sigma\,]\!]_{\mathit{eff}}|$*, with the application functions given by the evaluation morphisms. We call the elements of* R *the* sequentially realizable (SR) *functionals, and the elements of* R_{eff} *the* effective SR *functionals.*

Clearly the type structures R, R_{eff} are both *extensional*, in the sense that if $f \cdot x = g \cdot x$ for all x then $f = g$.

Remarks 3.7 (i) It should be clear already that both R and R_{eff} are in some sense models of PCF. One can easily exhibit realizers for the basic arithmetical operations; application and abstraction are given by the CCC structure of the category of modest sets; and for each type σ one can verify that the element $\mathbf{y} \in \mathcal{B}_{\mathit{eff}}$ realizes fixed point operators $Y \in R^{(\sigma\to\sigma)\to\sigma}$ and $Y_{\mathit{eff}} \in R_{\mathit{eff}}^{(\sigma\to\sigma)\to\sigma}$. One could prove directly at this stage that this gives adequate interpretations of PCF in R and R_{eff}, but we will defer a formal discussion of PCF until Section 9.

(ii) The reader may enjoy verifying that $R^{\overline{2}}$ and $R^{\overline{2}}_{\mathit{eff}}$ also contain a non-PCF-definable element F with the following specification (this is essentially the example given in the Introduction):

$$Fg \;=\; \begin{cases} 0 & \text{if } g\perp = 0 \\ 1 & \text{if } g\perp = \perp \text{ but } g0 = 0 \\ \perp & \text{otherwise.} \end{cases}$$

Unfortunately, many of the basic properties of R and R_{eff} are difficult to prove directly from the above definitions. For instance, it will turn out that R_{eff} can be identified with a substructure of R (that is, there are inclusions $R^\sigma_{\mathit{eff}} \rightarrowtail R^\sigma$ commuting with application), and that each R^σ is CPO with the ordering defined by

$$f \sqsubseteq g \quad \Leftrightarrow \quad \exists q \in \|f\|, r \in \|g\|.\ q \sqsubseteq r,$$

We will defer the proofs of these facts until we have gathered more information about these type structures.

3.3 Modified realizability

The construction given above will be the central one used in this paper. However, we now briefly consider another realizability construction that also gives rise to the SR functionals: the *modified realizability* model over $\mathcal{B}$. The notion of modified realizability is due in essence to Kreisel [26]; modified realizability toposes have been extensively studied e.g. in [20, 44]. For our purposes, it suffices to consider a much smaller category, analogous to the category of modest sets.

In the following definition, A is any total combinatory algebra, and we suppose moreover that we have an element $\mathbf{0} \in A$ such that $\mathbf{0} \cdot a = \mathbf{0}$ for all $a \in A$. (The element $\mathbf{0}$ plays a technical role in the construction of modified realizability toposes; it is not too important for the purposes of this paper.)

Definition 3.8 (Modified modest sets) *(i) A* modified modest set[2] X *over a combinatory algebra A consists of a modest set X over A in the usual sense, together with a subset $P_X \subseteq A$ (called the set of* potential realizers *for X) such that $\mathbf{0} \in P_X$ and $\|x\| \subseteq P_X$ for all $x \in |X|$.*

(ii) Suppose X, Y are modified modest sets over A. A function $f : |X| \to |Y|$ is said to be tracked in the modified sense *by $r \in A$ if r tracks f in the usual sense, and furthermore for all $a \in P_X$ we have $r \cdot a \in P_Y$. A morphism $f : X \to Y$ is a function $f : |X| \to |Y|$ that is tracked in the modified sense by some $r \in A$. We write $\mathbf{mMod}(A)$ for the category of modified modest sets.*

Modified modest sets may be thought of as modest sets with some extra "type information" built in: we may think of P_X as giving a "type" for realizers of elements of X. It is easy to show that $\mathbf{mMod}(A)$ is a cartesian closed category. The exponential Y^X is defined as follows: $|Y^X|$ is the set of morphisms $X \to Y$ in $\mathbf{mMod}(A)$; $\|f\|_{Y^X}$ is the set of elements r that track f in the modified sense; and P_{Y^X} is the set $(P_X \Rightarrow P_Y)$ (in general we write $(A \Rightarrow B)$ for the set $\{r \mid \forall a \in A.\ r \bullet a \in B\}$).

In the case $A = \mathcal{B}$ or $\mathcal{B}_{\mathit{eff}}$, we may take $\mathbf{0} = \lambda m.\bot$. Both $\mathbf{mMod}(\mathcal{B})$ and $\mathbf{mMod}(\mathcal{B}_{\mathit{eff}})$ have a natural number object N which may be described as follows: the underlying modest set of N is the object N described in Section 3.1, and $P_N = \{\overline{\bot}, \overline{0}, \overline{1}, \ldots\}$. Likewise, there is an obvious candidate for $N_\bot$ in these categories: the underlying modest set is $N_\bot$, and $P_{N_\bot} = P_N$. This means that as before we can give interpretations $[\![-]\!]_m$, $[\![-]\!]_{m,\mathit{eff}}$ of the simple types in these categories, and hence obtain type structures R_m, $R_{m,\mathit{eff}}$ from them.

Proposition 3.9 *For $A = \mathcal{B}$ or $\mathcal{B}_{\mathit{eff}}$, the interpretation of the simple types in $\mathbf{mMod}(A)$ agrees with that in $\mathbf{Mod}(A)$. That is, for each type σ we have $U[\![\sigma]\!]_m = [\![\sigma]\!]$ and $U[\![\sigma]\!]_{m,\mathit{eff}} = [\![\sigma]\!]_{\mathit{eff}}$. Hence $R_m = R$ and $R_{m,\mathit{eff}} = R_{\mathit{eff}}$.*

PROOF For simplicity, we just consider the case $A = \mathcal{B}$; the effective case is similar. First observe that the object $N_\bot$ in $\mathbf{mMod}(\mathcal{B})$ is isomorphic to the

[2] Jaap van Oosten has pointed out to me that, in terms of the categorical properties of these objects within the modified realizability topos, the term "discrete" is perhaps more appropriate than "modest" in this context. However, here we will stick with the term "modest" to maintain a sense of analogy with Section 3.1.

object $N'_\perp$ described as follows: the underlying modest set of $N'_\perp$ is again $N_\perp$, but $P_{N'_\perp}$ is the whole of $\mathcal{B}$. We can define an interpretation $[\![-]\!]'_m$ of the simple types using $N'_\perp$ instead of N', and hence obtain a type structure R'_m; it is easy to show by induction on types that $R'_m = R_m$.

We now show by induction on σ that $U[\![\sigma]\!]'_m = [\![\sigma]\!]$ and $P_{[\![\sigma]\!]'_m} = \mathcal{B}$. The base case is trivial by definition of $N'_\perp$. So suppose the hypotheses hold for σ and τ. By the definition of exponentials we have $P_{[\![\sigma \to \tau]\!]'_m} = \{r \mid \forall a \in \mathcal{B}.\, r \cdot a \in \mathcal{B}\} = \mathcal{B}$, since $\mathcal{B}$ is total. It is now clear that $U[\![\sigma \to \tau]\!]'_m = [\![\sigma \to \tau]\!]$. $\Box$

Remark 3.10 Our category $\mathbf{mMod}(A)$ is restricted enough that it embeds fully in both the modified realizability topos $\mathbf{mRT}(A)$ and Streicher's category $\mathbf{MAss}(A)$ of *modified assemblies* over A (see [54]). Both these embeddings preserve the cartesian closed structure, but unfortunately (in the case $A = \mathcal{B}$ or $\mathcal{B}_{\mathit{eff}}$) neither of them preserves the natural number object.

One reason for being particularly interested in our object $N_\perp$ will become apparent in Section 5. But we would also expect that in both $\mathbf{mRT}(\mathcal{B})$ and $\mathbf{MAss}(\mathcal{B})$, the finite types over the true natural number object yield the SR functionals.

4 Concrete data structures

In this section we consider the concrete data structures (CDSs) of Kahn and Plotkin [22], together with the sequential algorithms and sequential functions studied by Berry and Curien [5]. We will see that $\mathcal{B}$ itself be regarded as a *universal* object for a certain class of CDSs. It follows that a certain category of CDSs and sequential *functions* can be identified with a full subcategory of $\mathbf{Mod}(\mathcal{B})$, and that moreover there is a strong connection between sequential *algorithms* of cds and *realizers* in $\mathbf{Mod}(\mathcal{B})$.

4.1 Basic definitions

We begin by recalling some definitions from [5]. We refer the reader to this paper or to the book [14] for further details and motivation.

Definition 4.1 *A* concrete data structure (CDS) M *consists of the following:*

- *a countable set* C_M *of* cells*;*
- *a countable set of* V_M *of* values*;*
- *a set* $E_M \subseteq C_M \times V_M$ *of* events*, such that for all* $c \in C_M$ *there is at least one* v *such that* $(c, v) \in E_M$*;*
- *an* enabling *relation* $\vdash_M$ *between finite subsets of* E_M *and elements of* C_M*, which is* well-founded *in the following sense: there is no infinite sequence* $c_0, c_1, \ldots$ *of cells such that for each* i *there is a finite set* $t \subseteq E_M$ *such that* $t \vdash_M c_i$ *and* t *contains some element* (c_{i+1}, v)*.*

A state x *of* M *is a subset of* E_M *such that*

- *if* $(c,v) \in x$ *and* $(c,v') \in x$ *then* $v = v'$*;*
- *if* $(c,v) \in x$ *then there exists a finite set* $t \subseteq x$ *such that* $t \vdash c$*.*

We write $(D_M, \sqsubseteq)$ *for the set of states of* M *ordered by inclusion. We also write* D^0_M *for the set of* finite *states of* M*.*

If $t \vdash c$ we say that t is an *enabling* of c. We say that c is *filled* in a state x if $(c,v) \in x$ for some v, and that c is *accessible* from x if c is not filled in x but x contains an enabling of c.

The well-foundedness condition implies that any non-empty CDS has at least one cell c such that $\emptyset \vdash c$; such a cell is called *initial*. The second condition in the definition of state ensures that every state is in some sense reachable. Note that the poset of states is always a CPO.

We will restrict our attention mainly to CDSs with the following property:

Definition 4.2 *A CDS is* deterministic*, or is a* DCDS*, if for any state* x *and cell* c*,* x *contains at most one enabling of* c*.*

Two different notions of morphism between DCDSs are of interest: the "extensional" notion of sequential *function*, and the more "intensional" notion of sequential *algorithm*. Sequential functions are simply certain functions between the relevant sets of states:

Definition 4.3 *A* sequential function $f : M \to M'$ *between DCDSs is a monotone and continuous function* $f : D_M \to D_{M'}$ *such that, for any* $x \in D_M$ *and any* $c' \in C_{M'}$ *accessible from* $f(x)$*, if there exists a state* $y \sqsupseteq x$ *such that* c' *is filled in* $f(y)$*, then there is a cell* $c \in C_M$ *accessible from* x *such that* c *is filled in all such states* y*. Such a cell* c *is called a* sequentiality index *of* f *for* c' *at* x*.*

Note that the sequentiality index c need not be unique: indeed, a sequential *algorithm* can be regarded as telling us at each stage which cell c is to be filled next. Several equivalent definitions of sequential algorithm can be given; the following definition (adapted from [14]) is in some sense intermediate between the "concrete" and "abstract" definitions given in [5].

Definition 4.4 *If* M, M' *are DCDSs, the DCDS* $N = M'^M$ *is defined as follows:*

- $C_N = D^0_M \times C_{M'}$ *(we will write the cell* (x,c') *just as* xc'*).*
- $V_N = C_M + V_{M'}$ *(we denote the left and right inclusions by* $?, !$ *respectively).*
- *Events of* N *are of two kinds:*
  - $(xc', ?c) \in E_N$ *iff* c *is accessible from* x *in* M*;*
  - $(xc', !v') \in E_N$ *iff* $(c', v') \in E_{M'}$*.*
- *Instances of the enabling relation of* N *are of two kinds:*
  - $(yc', ?c) \vdash_N xc'$ *iff* $x = y \sqcup \{(c,v)\}$ *for some* v*;*

– $(x_1c'_1, !v'_1), \ldots, (x_nc'_n, !v'_n) \vdash_N xc'$ *iff* $x = x_1 \cup \cdots \cup x_n$ *and* $(c'_1, v'_1), \ldots, (c'_n, v'_n) \vdash_{M'} c'$.

A sequential algorithm $a : M \to M'$ *is simply a state of* M'^M. *The* application *operation* $\cdot : D_{M'^M} \times D_M \to D_{M'}$ *is defined by*

$$a \cdot x = \{(c', v') \mid \exists y \sqsubseteq x. (yc', !v') \in a\}.$$

It is shown in [14] that if M and M' are CDSs [resp. DCDSs] then so is M'^M. Moreover, for any sequential algorithm $a : M \to M'$ between DCDSs, the mapping $a_* : x \mapsto a \cdot x : D_M \to D'_M$ is a sequential function $M \to M'$. Conversely, every sequential function between DCDSs is represented by at least one sequential algorithm in this way. (Accordingly, we will sometimes say things like "Let $a_* : M \to M'$ be a sequential function".)

Given sequential algorithms $a : M \to M'$ and $b : M' \to M''$, their composition $ba : M \to M''$ may be defined directly (if somewhat opaquely) as follows.

- $(xc'', !v'') \in ba$ iff for some $y' = \{(c'_1, v'_1), \ldots, (c'_n, v'_n)\}$ and $x_1, \ldots, x_n \in D^0_M$ with $x = x_1 \cup \cdots \cup x_n$, we have $(x_ic'_i, !v_i) \in a$ for each i, and $(y'c'', !v'') \in b$.
- $(xc'', ?c) \in ba$ iff for some $y' = \{(c'_1, v'_1), \ldots, (c'_n, v'_n)\}$, $x_0, x_1, \ldots, x_n \in D^0_M$ with $x = x_0 \cup x_1 \cup \cdots \cup x_n$, and $c' \in C_{M'}$, we have $(x_ic'_i, !v_i) \in a$ for each $i \geq 1$, $(y'c'', ?c') \in b$ and $(x_0c', ?c) \in a$.

This agrees with the indirect definition of composition given in [14, Section 2.6]. One can check that $ba{\cdot}x = b{\cdot}(a{\cdot}x)$, so composition for sequential algorithms agrees with composition for sequential functions. We write **SeqAlg** for the category of DCDSs and sequential algorithms. A major result of [5] is that **SeqAlg** is a cartesian closed category, with the exponential M'^M defined as above. However, it is well known that the category **SeqFun** of DCDSs and sequential *functions* is not cartesian closed.

The following condition was recognized by Curien as defining an interesting class of CDSs (see [14, Def. 2.1.10]):

Definition 4.5 *A CDS M is called* sequential *if, for any cell d and any state x such that d is not filled in x, if there exists a state $y \sqsupseteq x$ such that d is filled in y, there is a cell c accessible from x such that c is filled in all such states y. Such a cell c is called a* sequentiality index *of M for d at x. We write* **SSeqFun** *for the category of sequential DCDSs with sequential functions, and* **SSeqAlg** *for the category of sequential DCDSs with sequential algorithms.*

We will sometimes implicitly assume that a sequential CDS M comes equipped with a choice of sequentiality index $c_{d,x}$ for each d and x to which the above conditions apply. Curien pointed out that sequential DCDSs are closed under finite products and exponentials—that is, **SSeqAlg** is a sub-CCC of **SeqAlg**. In fact, one can show that every sequential CDS is isomorphic to a sequential DCDS, and so the categories of sequential CDSs and sequential DCDSs are equivalent.

Remark 4.6 The above definitions all have a fairly evident *effective* analogue. For this, we need to work with CDSs M equipped with *enumerations* of C_M and V_M—that is, identifications of C_M and V_M with (not necessarily r.e.) subsets of N. An *effective state* of M is one that is given by a partial recursive function $C_M \rightharpoonup V_M$ (more precisely, by a partial recursive function $\mathrm{N} \rightharpoonup V_M$ whose domain is contained in C_M). If M, M' are DCDSs, a sequential function $f : M \to M'$ is *effective* if, for all effective states $x \in D_M$ and $c' \in C_{M'}$ satisfying the conditions of Definition 4.3, a sequentiality index of f for c' at x can be recursively computed from c' and a recursive index for x (one need not assume that different indices for x yield the same sequentiality index). An *effective sequential algorithm* $M \to M'$ is just an effective state of M'^M (note that enumerations for M, M' induce an enumeration for M'^M in a standard way). One may then check, by a routine effectivization of the standard proofs, that the effective sequential functions are precisely the maps induced by effective sequential algorithms. Such functions clearly map effective states to effective states.

We write $\mathbf{SeqAlg}_{\mathit{eff}}$ [resp. $\mathbf{SeqFun}_{\mathit{eff}}$] for the category of enumerated DCDSs and effective sequential algorithms [resp. functions]. It is clear that $\mathbf{SeqAlg}_{\mathit{eff}}$ is cartesian closed, with the same exponentials as **SeqAlg**.

We say an enumerated DCDS M is *effectively sequential* for all cells d and effective states x satisfying the requirements in Definition 4.5, a sequentiality index for d at x can be recursively computed from d and an index for x (again, we need not assume extensionality). We write $\mathbf{SSeqFun}_{\mathit{eff}}$ and $\mathbf{SSeqAlg}_{\mathit{eff}}$ for the respective full subcategories of $\mathbf{SeqFun}_{\mathit{eff}}$ and $\mathbf{SeqAlg}_{\mathit{eff}}$ consisting of effectively sequential DCDSs.

4.2 $\mathcal{B}$ as a universal object

The key observation of this section is that the combinatory algebra $\mathcal{B}$ can be viewed as (the set of states of) a concrete data structure. The cells of this CDS are just the natural numbers; the values are also the natural numbers; every pair (c, v) is an event; and the enabling relation is given by $\emptyset \vdash c$ for all c (all cells are initial). The states are then precisely the partial functions from cells to values, that is, elements of $\mathrm{N}_\perp^\mathrm{N}$. We will denote this CDS also by $\mathcal{B}$.

It is easy to see that $\mathcal{B}$ is a sequential DCDS. Note that $\mathcal{B}$ can also be described as a countably infinite product (in either **SeqFun** or **SeqAlg**) of copies of a single-cell CDS N whose values are the natural numbers.

Any CDS M can be "embedded" in $\mathcal{B}$ in such a way that the states of M are identified with a subset of the states of $\mathcal{B}$. Specifically, we pick some identifications of C_M, V_M with subsets of N (or we suppose such identifications are given by the hypothesis that C_M, V_M are countable). States of M are then identified with certain partial functions $\mathrm{N} \rightharpoonup \mathrm{N}$, i.e. states of $\mathcal{B}$. (We take these partial functions to be undefined outside the subset of N corresponding to C_M.) This gives us a continuous inclusion $\iota_M : D_M \hookrightarrow \mathcal{B}$ between CPOs.

For convenience, we will assume henceforth that C_M, V_M actually *are* subsets of N for every M that we consider. Furthermore, in Definition 4.4, we identify $V_N = C_M + V_{M'}$ with the set $\{?c \mid c \in C_M\} \cup \{!v' \mid v' \in V_{M'}\}$ where $?, ! : \mathrm{N} \to \mathrm{N}$ are the coding functions introduced in Section 2.

The following tells us when the embedding ι_M is well-behaved.

Proposition 4.7 *For a DCDS M, the following conditions are equivalent:*
(i) M is sequential.
(ii) ι_M is a sequential function $i_* : M \to \mathcal{B}$.
(iii) M is a retract of $\mathcal{B}$ in **SeqFun**.
(iv) M is a retract of $\mathcal{B}$ in **SeqAlg**.

Proof (i) $\Rightarrow$ (ii): Any sequentiality index of M for c at x is also a sequentiality index of ι_M for c at x. (Note that if $c \notin C_M$ then a sequentiality index of ι_M for n at x is not needed.) So $\iota_M = i_*$ for some sequential algorithm i.

(ii) $\Rightarrow$ (iii): It suffices to construct a sequential function $j_* : \mathcal{B} \to M$ such that $j_* i_* = \mathrm{id}_M$. Let us say that a cell d is *supported* in a state y of $\mathcal{B}$ if there is a state x of M such that $i_*(x) \sqsubseteq y$ and d is filled in $i_*(x)$. Write Σ_y for the set of cells supported in y, and define $j_*(y)$ to be the restriction of y to Σ_y. Clearly j_* is monotone and continuous, and $j_* i_* = \mathrm{id}_M$. Moreover, it is routine to check that j_* is sequential.

(iii) $\Rightarrow$ (i): Assume $M \xrightarrow{f_*} \mathcal{B} \xrightarrow{g_*} M$ is any retraction in **SeqFun**, and suppose we are given $x \in D(M)$ and d not filled in x such that d is filled in some $y \sqsupseteq x$. Since $g_* f_*(x) = x$, we may take c' to be a sequentiality index of g_* for d at $f_*(x)$; note that c' is filled in $f_*(y)$. We can now take c a sequentiality index of f_* for c' at x, and check that c is a sequentiality index of M for d at x.

(iii) $\Rightarrow$ (iv): It suffices to show that the only sequential algorithm representing the sequential function id_M is the identity. Clearly, for any cell c and state x, the only sequentiality index of id_M for c at x (if one exists) is c itself; and by the relationship between sequential algorithms and abstract algorithms established in [14, Section 2.6], this is enough.

(iv) $\Rightarrow$ (iii) is trivial. $\Box$

The above proposition shows that $\mathcal{B}$ is a *universal object* in the categories **SSeqFun** and **SSeqAlg**. We henceforth suppose that each sequential DCDS M comes equipped with some choice of sequential algorithms i_M, j_M representing i_*, j_* respectively.

Since **SSeqAlg** is cartesian closed, we have that $\mathcal{B}^{\mathcal{B}}$ is a retract of $\mathcal{B}$—that is, $\mathcal{B}$ is a *reflexive object* in **SSeqAlg**. In fact, one can choose a retraction $\mathcal{B}^{\mathcal{B}} \lhd \mathcal{B}$ that gives rise in the standard way to precisely the applicative structure $(\mathcal{B}, \bullet)$. This is not the case for the particular retraction given by Proposition 4.7, but the following proposition establishes the relevant retraction. The essential force of this proposition is that *irredundant realizers are sequential algorithms.* (Throughout this section, "irredundant" will mean $\bullet$-irredundant.)

Proposition 4.8 *(i) To any sequential algorithm $a : \mathcal{B} \to \mathcal{B}$ there corresponds an irredundant element $r_a \in \mathcal{B}$ such that $r \bullet b = a \cdot b$ for all $b \in \mathcal{B}$.*

(ii) The map $I_* : a \mapsto r_a$ constitutes a sequential function $\mathcal{B}^{\mathcal{B}} \to \mathcal{B}$.

(iii) The map I_* is a bijection between sequential algorithms $\mathcal{B} \to \mathcal{B}$ and irredundant elements of $\mathcal{B}$.

(iv) The map $J_* : r \mapsto I_*^{-1}(irr_\bullet(r))$ constitutes a sequential function $\mathcal{B} \to \mathcal{B}^{\mathcal{B}}$.

PROOF (i) Given a sequential algorithm a, we may construct a realizer $r = r_a$ as follows. For each $n \in \mathrm{N}$, let $\rho_n : \mathrm{Seq}(\mathrm{N}) \rightharpoonup \mathcal{B}^0$ be the (unique) smallest partial function with the following properties:

- $\rho_n(\epsilon) = \emptyset$.
- If $\rho_n(\alpha) = x$ and $(xn, ?m) \in a$, then $\rho_n(\alpha, v) = x \cup \{m \mapsto v\}$.

We now take r to be the smallest partial function such that, for all n, α, if $\rho_n(\alpha) = x$ and $(xn, l) \in a$ (where l may be $?c$ or $!v$) then $r\langle n, \alpha \rangle = l$.

We first check that r is irredundant. If $r\langle n, \alpha\rangle = l$ then from the definition l is a question or an answer. If moreover β is a proper prefix of α then $\rho_n(\beta, v)$ is defined for some v, so if $y = \rho_n(\beta)$ then $(yn, ?m) \in a$ for some m, whence $r\langle n, \beta\rangle =?m$. Finally, we have $(m, v) \in \rho_n(\beta_v) \sqsubseteq \rho_n(\alpha)$, and so if $x = \rho_n(\alpha)$ then $(xn, ?m) \notin a$, so $r\langle n, \alpha\rangle \neq ?m$.

Next we show that r realizes $b \mapsto a \cdot b$, that is, $(r \bullet b)(n) = v$ iff $\exists x \sqsubseteq b.\, (xn, !v) \in a$. By definition, $(r \bullet b)(n) = v$ iff there is a sequence $c_1 v_1 \ldots c_k v_k$ such that $(c_i, v_i) \in b$ for $1 \leq i < k$, and if x_i denotes $\rho_n\langle v_1, \ldots, v_i\rangle$ then $(x_i n, ?c_{i+1}) \in a$ for $0 \leq i < k$ and $(x_k n, !v) \in a$. So for the implication $\Rightarrow$ above we may take $y = x_n$, since $x_n \subseteq x$. Conversely, given $x \subseteq b$ with $(xn, !v) \in a$, let $y_0 = x$, and while $y_i \neq \emptyset$ let $(y_{i+1}n, ?d_i)$ be the unique enabling of $y_i n$ in a (recall that $\mathcal{B}^{\mathcal{B}}$ is a DCDS), where $y_i = y_{i+1} \sqcup \{(d_i, w_i)\}$. By the well-foundedness of the enabling relation we have $y_k = \emptyset$ for some k. But then the sequence $d_{k-1} w_{k-1} \ldots d_0 w_0$ satisfies the above requirements for $c_1 v_1 \ldots c_k v_k$.

(ii) Suppose given $a : \mathcal{B} \to \mathcal{B}$ and c accessible from r_a such that c is filled in r_b for some $b \sqsupseteq a$. Since r_b is irredundant, c must have the form $\langle n, \alpha\rangle$. Let $\beta = m_1, \ldots, m_k$ be the shortest prefix of α such that $\langle n, \beta\rangle$ is not filled in r_a; then $r_a\langle n, \gamma\rangle$ is a question for any proper prefix γ of β. So let $x = \rho_n(\beta)$ as above. It is easy to check that xn is a sequentiality index of I_* for c at a.

(iii) If $r \in \mathcal{B}$ is irredundant, then for each n and $\alpha = m_1, \ldots, m_k$ such that $r\langle n, \alpha\rangle$ is defined, let $\sigma_{n,\alpha}$ be the (least) finite partial function such that for $0 \leq i < k$, if $r\langle n, m_1, \ldots, m_i\rangle =?q_i$ then $\sigma_{n,\alpha}(q_i) = m_{i+1}$. Now define

$$a_r = \{(\sigma_{n,\alpha} n, l) \mid n \in \mathrm{N},\ \alpha \in \mathrm{Seq}(\mathrm{N}),\ l = r\langle n, \alpha\rangle\}$$

(here l may be either a question or an answer). Clearly a_r is a sequential algorithm $\mathcal{B} \to \mathcal{B}$, and it is straightforward to check that $a = a_r$ iff $r = r_a$.

(iv) Suppose given $r \in \mathcal{B}$, and xn accessible from $a = a_{r^\bullet}$ such that xn is filled in $a_{s^\bullet}$ for some $s \sqsupseteq r$. Let $\{(yn, ?c)\}$ be an enabling of xn in a, where $x = y \cup \{(c, v)\}$ and $y = \sigma_{n,\alpha}$ as above. Then for any s as above, we have that $\langle n, \alpha, v\rangle$ is accessible from $irr_\bullet(r)$ and filled in $irr_\bullet(s)$. Next, one can see by applying (iii) above to the realizer $irr_\bullet(\mathbf{irr}_\bullet)$ that the map $irr_\bullet$ is a sequential function $\mathcal{B} \to \mathcal{B}$. So take d a sequentiality index of this map for $\langle n, \alpha, v\rangle$ at r. Then d is clearly a sequentiality index of J_* for xn at r. $\square$

Since $J_* I_* = \mathrm{id}$, by the proof of Proposition 4.7 we have that $JI = \mathrm{id}$.

Clearly, morphisms $1 \to \mathcal{B}$ in **SeqAlg** correspond precisely to elements (or states) of $\mathcal{B}$. So by the above proposition, applying the functor $\mathrm{Hom}(1, -)$ to the composite

$$\mathcal{B} \times \mathcal{B} \xrightarrow{J \times \mathrm{id}} \mathcal{B}^{\mathcal{B}} \times \mathcal{B} \xrightarrow{ev} \mathcal{B}$$

gives precisely the function $\bullet : \mathcal{B} \times \mathcal{B} \to \mathcal{B}$.

Remarks 4.9 (i) By standard facts about reflexive objects in CCCs (see [3]), the above gives an alternative proof that $(\mathcal{B}, \bullet)$ is a combinatory algebra. In fact it proves more: namely, that $\mathcal{B}$ is a λ-*algebra*. That is, if M, N are formal meta-expressions over $\mathcal{B}$ (possibly involving λ^*) and $M =_\beta N$, then $[\![M]\!]_\nu = [\![N]\!]_\nu$ for all valuations ν. It would be interesting to know what λ-theory is induced by the interpretation of pure λ-terms in $\mathcal{B}$.

It is clear, however, that $\mathcal{B}$ is not a λ-*model*, since different irredundant elements can realize the same function $\mathcal{B} \to \mathcal{B}$.

(ii) An alternative way of viewing the above results is the following: given the monoid $\mathcal{M}$ of realizable endofunctions on $\mathcal{B}$ (i.e. sequential endofunctions on $\mathrm{N}_\perp^{\mathrm{N}}$), one can obtain **SSeqFun** as a full subcategory of its *Karoubi envelope* $K(\mathcal{M})$. This is interesting because the definition of $\mathcal{M}$ is much less cumbersome than that of **SSeqFun**. However, $K(\mathcal{M})$ is strictly larger than **SSeqFun**, as it is easy to find realizable idempotents on $\mathcal{B}$ (and even projections) whose kernel is not a concrete domain (see [22]).

Remark 4.10 One can easily obtain an effective analogue of Proposition 4.7, showing that $\mathcal{B}$ is a universal object in both $\mathbf{SSeqFun}_{\mathit{eff}}$ and $\mathbf{SSeqAlg}_{\mathit{eff}}$. One can also show that the retraction $(I, J) : \mathcal{B}^{\mathcal{B}} \lhd \mathcal{B}$ described by Proposition 4.8 lives in $\mathbf{SeqAlg}_{\mathit{eff}}$, and induces a bijection between effective sequential algorithms $\mathcal{B} \to \mathcal{B}$ and irredundant elements of $\mathcal{B}_{\mathit{eff}}$. Clearly, morphisms $1 \to \mathcal{B}$ in $\mathbf{SeqAlg}_{\mathit{eff}}$ correspond to elements of $\mathcal{B}_{\mathit{eff}}$, and so the retraction (I, J) gives rise to exactly the combinatory algebra $\mathcal{B}_{\mathit{eff}}$.

4.3 Sequential DCDSs and realizability

Armed with the above results, we can formulate a precise connection between CDSs and modest sets over $\mathcal{B}$: specifically, **SSeqFun** can be fully embedded in $\mathbf{Mod}(\mathcal{B})$.

We define a functor $E : \mathbf{SSeqFun} \to \mathbf{Mod}(\mathcal{B})$ as follows. Given M a sequential DCDS, we take $|EM|$ to be the set D_M of states of M, and for each $x \in D_M$ let $\|x\|_{EM}$ be the singleton set $\{i_{M*}(x)\}$. Given $f : M \to M'$ a sequential function, take $Ef = f : D_M \to D_{M'}$. Note that if $a : M \to M'$ is any sequential algorithm representing f then $r_a \equiv I(i_{M'} \circ a \circ j_M)$ is a realizer for Ef.

Likewise, we have a functor $E_{\mathit{eff}} : \mathbf{SSeqFun}_{\mathit{eff}} \to \mathbf{Mod}(\mathcal{B}_{\mathit{eff}})$ given as follows: $|E_{\mathit{eff}}M|$ is the set of effective states of M, and $\|x\|_{E_{\mathit{eff}}M} = \{i_{M*}(x)\}$ (note that if x is an effective state then $i_{M*}(x) \in \mathcal{B}_{\mathit{eff}}$). If $f : M \to M'$ is an effective sequential function then $E_{\mathit{eff}}f$ is the restriction of f to effective states. Note that if a is an effective sequential algorithm representing f then the realizer r_a defined as above is effective, since (we may assume) $i_{M'}$ and j_M are effective.

Proposition 4.11 *The functors E and E_{eff} are full and faithful.*

PROOF The faithfulness of E is trivial; the faithfulness of E_{eff} follows from the fact that any continuous map $D_M \to D_{M'}$ is determined by its action on effective states. To see that E is full, let g be any morphism $EM \to EM'$, realized by $r \in \mathcal{B}$ say. Then $j_{M'} \circ J_*(r) \circ i_M$ is a sequential algorithm $M \to M'$, representing a sequential function g^* such that $Eg^* = g$. Similarly for E_{eff}. □

Remarks 4.12 (i) Note that if $f : M \to M'$ is effective then the realizers for $E_{\mathit{eff}} f$ are precisely the *effective* realizers for Ef. This is because an arbitrary state of M, M' is a least upper bound of effective states, and so any realizer that takes effective states of M to effective states of M' also does the same for arbitrary states.

(ii) The above is an instance of a general piece of folklore about realizability models: given a category $\mathcal{C}$ with a universal and reflexive object X, giving rise to a combinatory algebra A_X, there is a full functor $E : \mathcal{C} \to \mathbf{Mod}(A_X)$ such that $Ef = Eg$ iff $\mathrm{Hom}(1, f) = \mathrm{Hom}(1, g)$.

(iii) All the modest sets in the image of either E or E_{eff} have the special property that every element has just one realizer. Such objects are (up to isomorphism) precisely the *projective* modest sets (see e.g. [31]).

(iv) It is immediate from Propositions 4.7 and 4.11 that the object $B \cong N_\perp^N$ is a universal object in the subcategory of $\mathbf{Mod}(\mathcal{B})$ corresponding to **SSeqFun**; similarly for B_{eff} in $\mathbf{Mod}(\mathcal{B}_{\mathit{eff}})$.

The above results establish the correspondence between *sequential functions* $M \to M'$ and *morphisms* $EM \to EM'$. There is also a close relationship between *sequential algorithms* and *realizers* in $\mathcal{B}$. The correspondence is not a perfect bijection—intuitively there are more realizers than sequential algorithms—but it is fairly strong:

Proposition 4.13 *Let M, M' be sequential DCDS. Then there are sequential algorithms $V : M'^M \to \mathcal{B}$ and $W : \mathcal{B} \to M'^M$ such that*

- *if $a : M \to M'$ is a sequential algorithm representing a sequential function f, then $V \cdot a$ tracks Ef;*
- *if $r \in \mathcal{B}$ tracks Ef, then $W \cdot r$ is a sequential algorithm representing f;*
- *$W \circ V = \mathrm{id}$;*
- *$V \circ W$ is realized by an element of $\mathcal{B}$.*

If moreover M, M' are effectively sequential, then V, W are effective and induce a similar correspondence between effective sequential algorithms and realizers in $\mathcal{B}_{\mathit{eff}}$; furthermore, $V \circ W$ is realized by an element of $\mathcal{B}_{\mathit{eff}}$.

Proof Since **SeqAlg** is cartesian closed, composition and application of sequential algorithms are themselves given by sequential algorithms. So there are sequential algorithms V, W defined by

$$V \;=\; \lambda a.\, I \cdot (i_{M'} \circ a \circ j_M), \qquad W \;=\; \lambda r.\, j_{M'} \circ (J \cdot r) \circ i_M.$$

The first two conditions are then clear (cf. the proof that E is a functor and is full). The third condition holds because $JI = \mathrm{id}$, $j_M i_M = \mathrm{id}$ and $j_{M'} i_{M'} = \mathrm{id}$.

The fourth condition is immediate since, by Proposition 4.8(i), every sequential algorithm $\mathcal{B} \to \mathcal{B}$ is realized by an element of $\mathcal{B}$. The effective versions of these conditions are straightforward given that $i_M, j_M, i_{M'}, j_{M'}$ are effective (an appeal to Remark 4.12(i) is needed for the second condition.) $\Box$

This gives us a particular retraction $(V, W) : M'^M \lhd \mathcal{B}$ in **SSeqAlg**. In general this is not identical to the chosen retraction (i, j) for M'^M, although the "translations" $i \circ W$ and $V \circ j$ are of course realizable by elements of $\mathcal{B}$. We thus have the following relationship between EM'^{EM} and $E(M'^M)$:

Proposition 4.14 *The exponential EM'^{EM} in* $\mathbf{Mod}(\mathcal{B})$ *is isomorphic to an obvious quotient Q of $E(M'^M)$, where $|Q|$ is the set of sequential* functions $M \to M'$*; and $\|f\|_Q$ is the union of the (singleton) sets $\|a\|_{E(M'^M)}$ for a a sequential algorithm representing f. Similarly in the effective case.* $\Box$

Since in general there is no *canonical* sequential algorithm representing a sequential function, the object Q (and hence EM'^{EM}) is not usually projective. We therefore have some kind of an explanation for why **SSeqFun** fails to be cartesian closed: the desired exponentials do live naturally in $\mathbf{Mod}(\mathcal{B})$, but they take us outside the subcategory corresponding to **SSeqFun**. This is an example of a common phenomenon: by passing from some category of objects to a category of partial equivalence relations on them, we often obtain a much richer structure. An abstract account of this phenomenon is given in [7].

5 The hypercoherence model

In the last section we explored the connections between the realizability model over $\mathcal{B}$ and the sequential algorithms model. In this section we relate both of these to the *strongly stable* model due to Bucciarelli and Ehrhard, as embodied in Ehrhard's category of *hypercoherences.* In [43], van Oosten gave a direct proof that the SR functionals coincide exactly with the strongly stable functionals of finite type. Here we follow a more scenic route to this result via the modified realizability model and the sequential algorithms model. Our proof invokes a theorem of Ehrhard [17] which states that the hypercoherence model may be obtained as an *extensional collapse* of the sequential algorithms model. (We will give our own proof of Ehrhard's theorem in Section 7.)

We discuss some of the insights that the hypercoherence model provides, particularly with regard to the stable order on the SR functionals. We also point out a curiosity, namely the failure of the *effective* analogue of Ehrhard's result.

5.1 An extensional collapse construction

We start by considering the extensional collapse of the sequential algorithms model, as studied by Ehrhard in [17]. We give a cheap proof that this construction yields precisely the SR functionals, by relating it to the modified realizability model described in Section 3.3.

It will be convenient to recast the definitions of Section 3.3 in terms of partial equivalence relations (PERs). By a *modified PER* X on $\mathcal{B}$ we will mean a set $P_X \subseteq \mathcal{B}$ with $\mathbf{0} \in P_X$, together with a partial equivalence relation $\sim_X$ on P_X (that is, a symmetric, transitive relation on P_X). We say that $r \in \mathcal{B}$ *realizes a morphism* $X \to Y$ of modified PERs if for all $a, b \in \mathcal{B}$ we have

$$a \in P_X \implies r \bullet a \in P_Y, \qquad a \sim_X b \implies r \bullet a \sim_Y r \bullet b.$$

A morphism of modified PERs is an equivalence class of such realizers, under the equivalence relation

$$q \sim r \iff \forall a \in \mathcal{B}.\, q \bullet a \sim_Y r \bullet a.$$

The category $\mathbf{MPER}(\mathcal{B})$ of modified PERs on $\mathcal{B}$ is clearly equivalent to $\mathbf{mMod}(\mathcal{B})$. The modified modest sets $[\![\, \sigma \,]\!]_m$ thus correspond to the modified PERs $M_\sigma = (P_\sigma, \sim_\sigma)$ defined as follows:

$$\begin{array}{lcl} P_{\overline{0}} & = & \{\overline{\bot}, \overline{0}, \overline{1}, \ldots\}, \\ x \sim_{\overline{0}} y & \iff & x = y, \\ P_{\sigma\to\tau} & = & (P_\sigma \Rightarrow P_\tau), \\ q \sim_{\sigma\to\tau} r & \iff & \forall x, y.\, x \sim_\sigma y \Longrightarrow q \bullet x \sim_\tau r \bullet y. \end{array}$$

Next we consider the obvious interpretation $[\![\, - \,]\!]_s$ of types in $\mathbf{SSeqAlg}$:

$$[\![\, \overline{0} \,]\!]_s \;=\; N, \qquad [\![\, \sigma \to \tau \,]\!]_s \;=\; [\![\, \tau \,]\!]_s^{[\![\, \sigma \,]\!]_s},$$

where N is the CDS with a single cell which may be filled by any natural number. Define a partial equivalence relation $\approx_\sigma$ on the states of $[\![\, \sigma \,]\!]_s$ inductively as follows:

$$x \approx_{\overline{0}} y \iff x = y, \qquad a \approx_{\sigma\to\tau} b \iff \forall x, y \in D([\![\, \sigma \,]\!]_s).\, x \approx_\sigma y \Longrightarrow a{\cdot}x \approx_\tau b{\cdot}y.$$

Remark 5.1 The structures $(D([\![\, \sigma \,]\!]_s), \approx_\sigma)$ are exactly the structures $([\sigma]^{\mathbf{SEQ}}_*, \approx)$ defined in Section 6 of [17]. The interpretation $[\sigma]^{\mathbf{SEQ}}$ was there defined within the category of *sequential structures*, a larger category than $\mathbf{SeqAlg}$. However, we have already seen that $\mathbf{SSeqAlg}$ is a full sub-CCC of $\mathbf{SeqAlg}$, and it is shown in [8, Chapter 5] that $\mathbf{SeqAlg}$ is a full sub-CCC of the category of sequential structures. Thus, all the sequential structures $[\sigma]^{\mathbf{SEQ}}$ lie within the sub-CCC corresponding to $\mathbf{SSeqAlg}$.

One can view each structure $(D([\![\, \sigma \,]\!]_s), \approx_\sigma)$ as a modified modest set $N_\sigma = (Q_\sigma, \approx_\sigma)$: we take Q_σ to be the set of realizers for elements of $E([\![\, \sigma \,]\!]_s)$—these are in bijective correspondence with states of $[\![\, \sigma \,]\!]_s$—and just transport $\approx_\sigma$ along this bijection. By Proposition 4.13 it is clear that there are application morphisms $N_{\sigma\to\tau} \times N_\sigma \to N_\tau$ realized essentially by $\bullet$.

The definitions of M_σ and N_σ look very similar, and indeed they turn out to be isomorphic:

Proposition 5.2 *For each type σ, we have $M_\sigma \cong N_\sigma$ in* $\mathbf{MPER}(\mathcal{B})$*; moreover, these isomorphisms commute with the evident application morphisms $M_{\sigma\to\tau} \times M_\sigma \to M_\tau$ and $N_{\sigma\to\tau} \times N_\sigma \to N_\tau$.*

Proof By induction on σ. The isomorphism $M_{\overline{0}} \cong N_{\overline{0}}$ is easy. So suppose c_σ, d_σ realize an isomorphism $M_\sigma \to N_\sigma$ and its inverse respectively, and likewise for τ. Let $V : [\![\tau]\!]_s^{[\![\sigma]\!]_s} \to \mathcal{B}$ and $W : \mathcal{B} \to [\![\tau]\!]_s^{[\![\sigma]\!]_s}$ be the sequential algorithms given by Proposition 4.13, and suppose $v, w \in \mathcal{B}$ are the corresponding realizers for morphisms $E([\![\sigma \to \tau]\!]_s) \leftrightarrows B$. By Proposition 4.13 we have that if $r \in Q_{\sigma\to\tau}$ then $v \bullet r \in (Q_\sigma \Rightarrow Q_\tau)$. We therefore have the following diagram of subsets of $\mathcal{B}$ and realizable functions between them (we label each arrow with an appropriate realizer):

$$P_{\sigma\to\tau} \equiv (P_\sigma \Rightarrow P_\tau) \underset{\lambda^* xy.d_\tau(x(c_\sigma y))}{\overset{\lambda^* xy.c_\tau(x(d_\sigma y))}{\rightleftarrows}} (Q_\sigma \Rightarrow Q_\tau) \underset{v}{\overset{w}{\rightleftarrows}} Q_{\sigma\to\tau}$$

Hence we obtain realizers $c_{\sigma\to\tau}, d_{\sigma\to\tau}$ for the composite functions $P_{\sigma\to\tau} \leftrightarrows Q_{\sigma\to\tau}$. It is routine to verify (using the induction hypothesis) that $c_{\sigma\to\tau}$ and $d_{\sigma\to\tau}$ realize morphisms $M_{\sigma\to\tau} \to N_{\sigma\to\tau}$ and $N_{\sigma\to\tau} \to M_{\sigma\to\tau}$ respectively; that these morphism commute with application; and hence that they are mutually inverse. $\Box$

Now consider the type structure E, where E^σ is the subquotient of Q_σ by $\approx_\sigma$, and the application operations are induced by $\bullet$. This type structure is known as the *extensional collapse* of the sequential algorithms model. The following is now immediate from the above and Proposition 3.9:

Corollary 5.3 *The type structure E is isomorphic to the type structure R of SR functionals.* $\Box$

Remarks 5.4 (i) The effective analogue of this result also goes through: the extensional collapse E_{eff} of the evident interpretation $[\![-]\!]_{s,\mathit{eff}}$ in the effective sequential algorithms model $\mathbf{SSeqFun}_{\mathit{eff}}$ is isomorphic to the type structure of effective SR functionals. Note that the sets $P_{\mathit{eff},\sigma}$ are defined by working entirely within $\mathcal{B}_{\mathit{eff}}$, whereas the sets $Q_{\mathit{eff},\sigma}$ are defined via the interpretation of σ in the "full" sequential algorithms model. But this does not matter since it is easily shown that $P_{\mathit{eff},\sigma} = P_\sigma \cap \mathcal{B}_{\mathit{eff}}$ (cf. Remark 4.12(i)).

(ii) It is shown by Laird in [28] that the effective sequential algorithms model coincides exactly with the observational quotient of the language μPCF , an idealized functional language with control. From this and the above results, it follows directly that R_{eff} is the extensional collapse of the language μPCF. More precisely, R_{eff} is isomorphic to the subquotient of the type structure of closed μPCF terms by the logical relation induced by observational equivalence at ground type. Informally, this means that every effective SR functional can be written in μPCF.

One also obtains a similar result for the programming language PCF+catch studied by Cartwright, Curien and Felleisen in [11]. (The relationship between μPCF and PCF+catch is discussed in [28].) In [11] it is shown that PCF+catch has a fully abstract interpretation in the effective sequential algorithms model; a universality result for such an interpretation is proved by Kanneganti, Cartwright and Felleisen in [23].[3] It follows from this that R_{eff} is the extensional collapse of PCF+catch. We would expect that many other programming languages should give rise to R_{eff} in this way—see Section 12.3.

5.2 Sequential algorithms and hypercoherences

We now recall some definitions concerning hypercoherences, and state the main theorem of Ehrhard [17]. For more background on hypercoherences, see this paper or [15].

Definition 5.5 (Hypercoherences) *(i) A* hypercoherence X *is a countable set* $|X|$ *together with a set* Γ_X *of non-empty finite subsets of* $|X|$, *such that* $\{a\} \in \Gamma_X$ *for each* $a \in |X|$. *We call* $|X|$ *the* underlying set *and* Γ_X *the* atomic coherence *of* X.

(ii) The domain $D(X)$ *generated by a hypercoherence* X *is the set of all subsets* $x \subseteq |X|$ *such that for all non-empty finite* $u \subseteq x$ *we have* $u \in \Gamma_X$. *We will refer to the elements* $x \in D(X)$ *as* states *of* X.

(iii) The coherence $\mathcal{C}(X)$ *generated by* X *is the set of non-empty finite subsets* $A \subseteq D(X)$ *such that, for all non-empty finite* $u \subseteq |X|$, $u \lhd A$ *implies* $u \in \Gamma_X$, *where* $u \lhd A$ *means*

$$\forall a \in u.\, \exists x \in A.\, a \in x \quad \wedge \quad \forall x \in A.\, \exists a \in u.\, a \in x.$$

The intuition here is that $|X|$ is some set of "atoms of information", and Γ_X specifies which finite combinations of atoms are "consistent". The domain $D(X)$ can then be seen as the set of consistent "states" of X, where consistency is determined by looking at finite subsets. In fact $D(X)$ (ordered by inclusion) is always a *qualitative domain*. The coherent subsets A of $D(X)$ are, roughly speaking, those such that any finite "cross-section" u of A looks consistent. The coherent subsets are meant to generalize the notion of consistent pair of states in the theory of stable domains: they are those subsets for which we require morphisms to preserve the meets. This is reflected in the next definition:

Definition 5.6 (Strong stability) *Let* X, Y *be hypercoherences. A* strongly stable function $X \to Y$ *is a continuous function* $f : D(X) \to D(Y)$ *such that for any* $A \in \mathcal{C}(X)$ *we have* $f(A) \in \mathcal{C}(Y)$ *and* $f(\bigcap A) = \bigcap f(A)$. *We write* **HC** *for the category of hypercoherences and strongly stable functions.*

It is shown in [15, 16] that **HC** is a cartesian closed category, indeed a sub-CCC of the category of dI-domains with coherence. Given hypercoherences X, Y, the exponential $Z = Y^X$ may be defined explicitly as follows: $|Z|$ is the set of pairs (x, b) where $x \in D(X)$ is finite and $y \in |Y|$; and for non-empty finite $w \subseteq |Z|$ we have $w \in \Gamma(Z)$ iff

$$\pi_1(w) \in \mathcal{C}(X) \implies (\pi_2(w) \in \Gamma(Y) \wedge (\pi_2(w) \text{ a singleton} \implies \pi_1(w) \text{ a singleton})).$$

[3]Actually in [23] the proof is worked out for a variant of PCF+catch including *error values*, but the same technique works for the error-free version.

In addition, we have a hypercoherence N of natural numbers, defined by

$$|N| \;=\; \mathrm{N}, \qquad \Gamma_N = \{\{n\} \mid n \in \mathrm{N}\}.$$

The associated domain $D(N)$ is the familiar domain $N_\perp$. This means that we have an interpretation $[\![-]\!]_{HC}$ of the simple types in **HC**, giving rise to a type structure H in the sense of Definition 3.6 (take $H^\sigma = D([\![\sigma]\!]_{HC}) \cong \mathrm{Hom}(1, [\![\sigma]\!]_{HC})$). We call H the type structure of *strongly stable functionals*.

The following result appears as Theorem 5 of [17]:

Theorem 5.7 (Ehrhard) *The type structures E and H are isomorphic.* □

From this and Corollary 5.3 above, we may immediately recover the main result of [43]:

Corollary 5.8 (van Oosten) *The type structures H and R are isomorphic.* □

Remarks 5.9 (i) The proof of Theorem 5.7 is non-constructive: an appeal to König's Lemma is needed to show that every *infinite* element of H is represented by a sequential algorithm (see [17, Section 6]). The failure of the effective analogue of the theorem (see below) shows that the non-constructivity is essential.

(ii) If preferred, one could invoke van Oosten's result and deduce Ehrhard's theorem as a corollary. We give an alternative proof of Theorem 5.8, not dependent on either [17] or [43], in Section 7 below.

(iii) It is interesting to compare Ehrhard's result in in [17], which we have invoked above, with his earlier construction of the hypercoherence model as the extensional collapse of the *extensional sequential algorithms* model (see [16]). Roughly speaking, in [17] we first define an interpretation of all types without regard to extensionality, and then pick out a type structure by the extensional collapse construction, whereas in [16] we carve out the subset of extensionally well-behaved algorithms at each type level, and use only this in defining the interpretation of the next type level. Intuitively, this corresponds exactly to the difference between exponentials in *modified* and *standard* realizability models.

The connection between the construction of [17] and modified realizability is made precise above. One might hope to make a similar precise link between the construction of [16] and standard realizability, yielding yet another proof of van Oosten's result, and indeed we initially tried to do this. The precise connection seems elusive, but for an interesting reason. In both [16, 17] Ehrhard works in the category of sequential structures (cf. Remark 5.1), and it turns out that in [16] this extra generality is quite essential: even the sequential structure used to interpret the type $(\overline{0}^2 \to \overline{0}) \to \overline{0}$ falls outside the class of CDSs. We are thus unable to transfer the results of [16] to the realizability model.

The hypercoherence presentation of R shows a remarkable fact: the "computational" notion of sequential realizability can be captured exactly by a preservation property with a finitary character.

Many properties of the SR functionals that are hard to prove from the definition of R are relatively easy for H. One such property concerns the *effective* *presentability* of the finite elements (other examples appear in Section 5.4). Specifically, let H_0^σ be the poset of finite elements of $(H^\sigma, \subseteq)$. In view of the finitary nature of the construction of exponentials in **HC**, the following fact is clear (we omit the detailed proof):

Proposition 5.10 *There are bijections $\beta_\sigma : \mathrm{N} \to H_0^\sigma$ for each type σ, with respect to which*

- *the ordering and (pairwise) consistency relations on H_0^σ are decidable;*
- *the (binary) meet operations $H_0^\sigma \times H_0^\sigma \to H_0^\sigma$ are recursive;*
- *the application operations $H_0^{\sigma\to\tau} \times H_0^\sigma \to H_0^\tau$ are recursive.*

Moreover, codes for these operations may be obtained recursively from codes for σ and τ. $\square$

It follows that the evident *finitary* analogue R_{fin} of R (in which we take $2_\perp$ instead of $N_\perp$ as the ground type) is decidable. That is, the set R^σ_{fin} and the application functions $\cdot_{\sigma\tau}$ can be computed recursively from σ, τ (so that, for example, the size of R^σ_{fin} is recursive in σ. An important result of Loader [29] shows that the corresponding statement fails for PCF-sequentiality.

5.3 Effectivity in hypercoherences

Next we draw attention to a surprising anomaly. The effective analogue of Theorem 5.7 fails: that is, the extensional collapse of the effective sequential algorithms model does *not* coincide with the r.e. submodel of the hypercoherence model. We give an example to show that, even at first order types, there are r.e. strongly stable functions that are not represented by any effective sequential algorithm. This is somewhat analogous to the result due to M.B. Trakhtenbrot [56] that there exist r.e. elements of the Milner model that are not PCF-definable. However, the two situations are in fact quite distinct, so we cannot simply re-use known examples of the Trakhtenbrot phenomenon in our setting.

We should clarify what we mean by an r.e. element of the hypercoherence model. By an *enumerated hypercoherence* we shall mean a hypercoherence X together with a bijection between $|X|$ and a subset of N; the *r.e. states* of X are then those $x \in D(X)$ that correspond under this bijection to r.e. subsets of N. It is easy to give a standard enumeration for each of the hypercoherences $[\![\,\sigma\,]\!]_{HC}$: for $[\![\,\overline{0}\,]\!]_{HC} = N$ we may take the identity enumeration; and given enumerations for X, Y we may obtain an enumeration for $Z = Y^X$ using some standard effective codings for pairs and finite sets. We shall always consider each $[\![\,\sigma\,]\!]_{HC}$ as being equipped with this standard enumeration; it is worth remarking that the atomic coherence $\Gamma_{[\![\sigma]\!]_{HC}}$ then corresponds to a *decidable* set of finite subsets of N. The effective states of these hypercoherences are of course closed under application, and so they constitute a substructure H_{eff} of H.

For convenience, we will identify states of the CDS N, and states of the hypercoherence N, with elements of $\mathrm{N}_\perp$. In the following proposition we write $\sigma = \overline{0} \to \overline{0} \to \overline{0}$ and $\tau = \overline{0} \to \overline{0} \to \overline{0} \to \overline{0}$.

Proposition 5.11 *There exists an r.e. element $t \in D([\![\tau]\!]_{HC})$ such that there is no effective sequential algorithm $a \in [\![\tau]\!]_{s,\mathit{eff}}$ with $a \cdot x \cdot y \cdot z = txyz$ for all $x, y, z \in \mathrm{N}_\perp$.*

Proof Let X be the hypercoherence $[\![\tau]\!]_{HC}$. We may identify $|X|$ with the set of tuples (x, y, z, w) where $x, y, z \in \mathrm{N}_\perp$ and $w \in \mathrm{N}$. Let A, B be disjoint but recursively inseparable r.e. subsets of N: for example, take

$$A \ = \ \{e \mid \varphi_e(0) = 0\}, \qquad B \ = \ \{e \mid \varphi_e(0) = 1\},$$

where φ_e is the eth partial recursive function. Now let $t \subseteq |X|$ be the set

$$\{(x, 1, 1, 1) \mid x \in \mathrm{N}\} \ \cup \ \{(x, 0, \perp, 0) \mid x \in A\} \ \cup \ \{(x, \perp, 0, 0) \mid x \in B\}.$$

Then t is r.e. since both A, B are r.e. To see that $t \in D(X)$, first let $Y = [\![\sigma]\!]_{HC}$, and for each $x \in \mathrm{N}$ let $t_x = \{(y, z, w) \mid (x, y, z, w) \in t\}$ considered as a subset of $|Y|$. It is easy to check that $t_x \in D(Y)$ for each x (there are just three possibilities for t_x). But now since $X = Y^N$, we can see from the definition of exponentials in **HC** that if u is any non-empty finite subset of t then $u \in \Gamma_X$. Thus $t \in D(X)$.

Now suppose, for contradiction, that a is an effective sequential algorithm representing t. Then for $x \in \mathrm{N}$ we have that $a \cdot x$ is an effective sequential algorithm representing t_x, recursive in x. If M is the CDS $[\![\sigma]\!]_s$, we may identify C_M with the set of pairs (y, z) with $y, z \in \mathrm{N}_\perp$, and V_M with the set $\{?_y, ?_z\} \cup \{!w \mid w \in \mathrm{N}\}$, where $?_y, ?_z$ correspond to requests for the first and second arguments respectively. Now for each $x \in \mathrm{N}$ we have $a \cdot x \in D(M)$, and it is easy to see that $((1, 1), !1) \in a \cdot x$. Using the sequentiality of $a \cdot x$, we can deduce that the cell $(\perp, \perp)$ is filled with either $?_y$ or $?_z$ in $a \cdot x$. Furthermore, since $a \cdot x$ is recursive in x, there is a recursive function of x telling us which of these is the case. But such a function clearly separates A and B, giving us a contradiction. $\Box$

By Remark 5.4, the above proposition suffices to show that the type structure R_{eff} is not isomorphic to H_{eff}. We will see in Section 7 that R_{eff} is a proper substructure of H_{eff}. It appears that R_{eff} is the "right" effective analogue of the SR functionals, while H_{eff} seems to be something of a curiosity.

5.4 The stable order

The set of states of any hypercoherence are naturally ordered by inclusion. It is known e.g. from [15] that for a function space Y^X, this ordering coincides with Berry's *stable* order on functions. The stable order also plays an important role in the theory of sequential functions. Here we briefly consider how the stable order fits into the picture described in this section and the last.

We first recall some standard notions. A poset $(X, \leq)$ is called *stable* if whenever $x, y \in X$ have a common upper bound, they have a greatest lower bound $x \wedge y$. A function $f : X \to Y$ between such posets is *stable* if whenever $x, y \in X$ have an upper bound we have $f(x \wedge y) = f(x) \wedge f(y)$. It is well known that the states of a DCDS [resp. hypercoherence] form a stable poset, and moreover that sequential functions [resp. strongly stable functions] are stable.

If X, Y are stable posets, the *stable order* $\leq_s$ on stable functions $X \to Y$ is defined as follows:

$$f \leq_s g \iff \forall x, y \in X.\ x \leq y \Longrightarrow f(x) = f(y) \wedge g(x).$$

The category of stable posets and stable functions is cartesian closed: the exponential $[X \Rightarrow_s Y]$ is the set of stable functions $X \to Y$ ordered by $\leq_s$.

For DCDSs, the stable order admits alternative characterizations:

Proposition 5.12 *Suppose $f, g : M \to M'$ are sequential functions between DCDSs. Then the following are equivalent:*

(i) $f \leq_s g$.

(ii) There exist sequential algorithms a, b with $a_* = f$, $b_* = g$ such that $a \sqsubseteq b$.

(iii) There exist realizers $q \in \|Ef\|, r \in \|Eg\|$ such that $q \sqsubseteq r$.

PROOF (i) $\Rightarrow$ (ii): Given $f \leq_s g$, take any b such that $b_* = g$, and define

$$a \;=\; \{(xc', ?c) \in b \mid \mathit{true}\} \;\cup\; \{(xc', !v') \in b \mid (c', v') \in f(x)\}.$$

It is routine to verify that a is a sequential algorithm and that $a_* = f$.

(ii) $\Rightarrow$ (i): The evaluation morphism $D(M'^M) \times D(M) \to D(M')$ is stable, and so we obtain a stable function $D(M'^M) \to [D(M) \Rightarrow_s D(M')]$ mapping $a \mapsto a_*$. So if $a \sqsubseteq b$ then $a_* \leq_s b_*$.

(ii) $\Leftrightarrow$ (iii) follows easily from Proposition 4.8. $\square$

For hypercoherences, as we have already remarked, the stable order on morphisms $X \to Y$ coincides with the inclusion order on $D(Y^X)$. Furthermore, this ordering on the type structure H can alternatively be characterized in terms of E or R:

Proposition 5.13 *Suppose $x \in H^\sigma$, $x' \in E^\sigma$ and $x'' \in R^\sigma$ all correspond under the isomorphisms $H \cong E \cong R$, and similarly for y, y', y''. The following are equivalent:*

(i) $x \sqsubseteq y$ in the inclusion order.

(ii) There exist sequential algorithms $a \in x'$, $b \in y'$ such that $a \sqsubseteq b$.

(iii) There exist realizers $q \in \|x''\|$, $r \in \|y''\|$ such that $q \sqsubseteq r$.

PROOF (i) $\Rightarrow$ (ii): For *finite* elements x, y this is Lemma 6 of [17]. One can extend the result to infinite elements by an application of König's Lemma exactly as in [17, Theorem 5]. Let (r_n) be the standard increasing sequence of finite retractions on H^σ, and let (p_n) be the corresponding sequence of retractions on $D^\sigma \equiv D([\![\, \sigma \,]\!]_s)$, as described in [17]. Each of these retractions has a finite image consisting entirely of finite elements, and the lub of each sequence is the identity. Suppose $x \sqsubseteq y$, and for each n let $x_n = r_n(x)$, $y_n = r_n(y)$. Since these are finite elements, there exist $a_n \sqsubseteq b_n$ in the image of p_n that represent the corresponding elements $x'_n, y'_n \in E^\sigma$. In this case, we will say that $c_n = (a_n, b_n)$ *represents* $z_n = (x_n, y_n)$.

For each n there is a sequence $c_0 \sqsubseteq \ldots \sqsubseteq c_n$ in $D^\sigma \times D^\sigma$, where each c_i represents z_i: for instance, take (a_n, b_n) representing z_n and set $c_i = (p_i(a_n), p_i(b_n))$.

Moreover, each z_i is represented by only finitely many pairs, since the image of p_i is finite. So by König's Lemma, there exists an infinite sequence $c_0 \sqsubseteq c_1 \sqsubseteq \ldots$ where c_i represents z_i. By taking $c = \bigsqcup c_i$ in $D^\sigma \times D^\sigma$, we obtain a, b representing x', y' as required.

(ii) $\Rightarrow$ (i) is Lemma 7(i) of [17].

(ii) $\Leftrightarrow$ (iii): By Proposition 5.12, (ii) holds iff there are realizers $q' \sqsubseteq r'$ for the elements corresponding to x, y respectively in the modified modest set N_σ. So by Proposition 5.2, (ii) holds iff the corresponding property holds for M_σ. But this is clearly equivalent to (iii), by inspection of the proof of Proposition 3.9. $\square$

It follows immediately from Proposition 5.13 that the relation on each R^σ given by condition (iii) above is a complete partial order. What seems to be the analogous question for PCF-sequentiality is still open: for the games models of PCF (for example), it is not known whether the extensional collapse at each simple type is a CPO with the evident ordering.

6 A presheaf model

We now consider a characterization of the type structure R due to Colson and Ehrhard [13], inspired by the notion of reducibility from proof theory. As we shall see, it is a fairly short step from this to a description of R in terms of *presheaves* over the monoid of sequential endofunctions on $\mathrm{N}_\perp^\mathrm{N}$. For us, the interest of these results is twofold: firstly, they yield yet another mathematical characterization of R, quite different in flavour from those we have considered so far; and secondly, they reveal a sense in which sequential realizability may be seen as "canonically" extending the familiar notion of first-order sequentiality to higher types.

6.1 The Colson-Ehrhard characterization

We first summarize the crucial definitions and the main result from [13]. As before we write $\mathrm{N}_\perp^\mathrm{N}$ for the set of all partial functions $\mathrm{N} \rightharpoonup \mathrm{N}$, and we suppose we have a standard pairing function $\langle -, - \rangle : \mathrm{N}_\perp^\mathrm{N} \times \mathrm{N}_\perp^\mathrm{N} \to \mathrm{N}_\perp^\mathrm{N}$, with corresponding projections *fst*, *snd*. For each type σ we define a set L^σ, and a set L^σ_ω of functions from $\mathrm{N}_\perp^\mathrm{N}$ to L^σ, by simultaneous induction as follows:

- $L^{\overline{0}} = \mathrm{N}_\perp$.
- $L^{\overline{0}}_\omega$ is the set of sequential functions $\mathrm{N}_\perp^\mathrm{N} \to \mathrm{N}_\perp$ in the sense of Milner [37] and Vuillemin [57] (equivalently, the functions $\mathrm{N}_\perp^\mathrm{N} \to \mathrm{N}_\perp$ realizable by an element of $\mathcal{B}$).
- $L^{\sigma\to\tau}$ is the set of all functions $f : L^\sigma \to L^\tau$ such that for all $g \in L^\sigma_\omega$ we have $f \circ g \in L^\tau_\omega$.
- $L^{\sigma\to\tau}_\omega$ is the set of all functions $f : \mathrm{N}_\perp^\mathrm{N} \to L^{\sigma\to\tau}$ such that for all $g \in L^\sigma_\omega$ the function $\lambda r : \mathrm{N}_\perp^\mathrm{N}.\, f(\mathit{fst}\ r)(g(\mathit{snd}\ r))$ is in L^τ_ω.

The sets L^σ constitute a type structure L, which can be seen as arising naturally from the set of sequential functions $\mathrm{N}_\perp^\mathrm{N} \to \mathrm{N}_\perp$.

The set $\mathrm{N}_\bot^{\mathrm{N}}$ obviously plays an important role in the above definition. We shall also consider the hypercoherence N^ω that intuitively corresponds to $\mathrm{N}_\bot^{\mathrm{N}}$. We can define N^ω directly as follows: $|N^\omega| = \mathrm{N} \times \mathrm{N}$; and for non-empty finite $w \subseteq |N^\omega|$ we have $w \in \Gamma(N^\omega)$ iff

$$\pi_1(w) \text{ a singleton } \Longrightarrow \pi_2(w) \text{ a singleton.}$$

One can check that $D(N^\omega) \cong \mathrm{N}_\bot^{\mathrm{N}}$, and that N^ω is indeed the ω-fold product of copies of N in **HC**.

We now import the main result of [13]. For the sake of completeness we give a summary of the proof.

Theorem 6.1 (Colson/Ehrhard) $L \cong H$.

PROOF (Outline.) The key lemma is the following: if X, Y are hypercoherences, a set-theoretic function $D(X) \to D(Y)$ is a strongly stable map $X \to Y$ provided for every strongly stable map $g : N^\omega \to X$, the function $f \circ g : N^\omega \to Y$ is strongly stable. This is proved as follows. Suppose f satisfies the proviso: we need to show that f is ω-continuous and preserves coherent sets and their meets. For continuity, one shows that every ω-chain in $D(X)$ is the image of a "generic" chain in $D(N^\omega)$ via a strongly stable map g. The strong stability of $f \circ g$ then implies that f preserves the lub of this chain. For coherent sets, one shows that every coherent set in X is the image of a coherent set in N^ω via a strongly stable map g. Again, the strong stability of $f \circ g$ then gives us that f preserves this coherent set and its meet.

One then shows that at all types σ there are bijections $L^\sigma \cong D(\llbracket \sigma \rrbracket_{HC})$ and $L^\sigma_\omega \cong D(N^\omega \Rightarrow \llbracket \sigma \rrbracket_{HC})$ that commute with the application maps. This is reasonably straightforward by simultaneous induction on σ, using the above lemma. $\Box$

6.2 A presheaf presentation

Next we relate L to a type structure arising from a certain category of presheaves. In the first instance, consider the two-object category $\mathcal{C}$ consisting of the sets 1 and $\mathrm{N}_\bot^{\mathrm{N}}$ together with all Milner-Vuillemin sequential functions between them. (Equivalently, $\mathcal{C}$ may be described as the full subcategory of **SeqFun** consisting of the objects 1 and $\mathcal{B}$.) For typographical convenience, we will write B for the object $\mathrm{N}_\bot^{\mathrm{N}}$ of $\mathcal{C}$.

The *presheaf category* over $\mathcal{C}$ is simply the category $[\mathcal{C}^{op}, \mathbf{Set}]$ of functors $\mathcal{C}^{op} \to \mathbf{Set}$ and natural transformations between them. Given $F, G : \mathcal{C}^{op} \to \mathbf{Set}$, we write $\mathrm{Nat}(F, G)$ for the set of natural transformations $F \to G$. It is well known that the presheaf category over any small category is cartesian closed. In the case of our category $\mathcal{C}$, if $F, G : \mathcal{C}^{op} \to \mathbf{Set}$ we may describe G^F explicitly as follows: $G^F(1) = \mathrm{Nat}(F, G)$; $G^F(B) = \mathrm{Nat}(h_B \times F, G)$, where $h_B = \mathrm{Hom}_{\mathcal{C}}(-, B)$; and $G^F(f)$ is induced by precomposition with $\mathrm{Hom}_{\mathcal{C}}(-, f) \times \mathrm{id}_F$.

We can obtain an interpretation of the simple types in $[\mathcal{C}^{op}, \mathbf{Set}]$ by fixing on an interpretation of the type $\overline{0}$. Let $M_{\overline{0}} : \mathcal{C}^{op} \to \mathbf{Set}$ be the functor defined as follows:

$M_{\overline{0}}(1) = L^{\overline{0}} = \mathrm{N}_\perp$, $M_{\overline{0}}(B) = L^{\overline{0}}_\omega$, and the action of $M_{\overline{0}}$ on morphisms is given by precomposition. At higher types we then define $M_{\sigma\to\tau}$ to be the exponential $M_\tau^{M_\sigma}$ in $[\mathcal{C}^{op}, \mathbf{Set}]$. The following proposition spells out the relationship between the functors M_σ and the sets $L^\sigma, L^\sigma_\omega$:

Proposition 6.2 *(i) For each type σ, there are bijections $\beta^\sigma : M_\sigma(1) \cong L^\sigma$ and $\beta^\sigma_\omega : M_\sigma(B) \cong L^\sigma_\omega$, with respect to which the action of M_σ on functors corresponds to precomposition.*

(ii) For each σ, τ, the function $M_{\sigma\to\tau}(1) \times M_\sigma(1) \to M_\tau(1)$, $(\alpha, x) \mapsto \alpha_1(x)$ corresponds under these bijections to the application function $L^{\sigma\to\tau} \times L^\sigma \to L^\tau$.

Proof By simultaneous induction on σ. For type $\overline{0}$, (i) holds by definition of $M_{\overline{0}}$. So suppose (i) holds for σ and τ. We first establish the bijection $\beta^{\sigma\to\tau} : M_{\sigma\to\tau}(1) \cong L^{\sigma\to\tau}$. Suppose $\alpha \in M_{\sigma\to\tau}(1)$, and let $f = \beta^\tau \circ \alpha_1 \circ \beta^{\sigma-1} : L^\sigma \to L^\tau$. By the naturality of α at each $x : 1 \to B$, we have that

$$\beta^\tau_\omega \circ \alpha_B \circ \beta^{\sigma-1}_\omega \;=\; f \circ - \;:\; L^\sigma_\omega \to L^\tau_\omega.$$

In particular, for any $g \in L^\sigma_\omega$ we have $f \circ g \in L^\tau_\omega$; thus $f \in L^{\sigma\to\tau}$. Conversely, suppose $f \in L^{\sigma\to\tau}$; then $f^- \equiv f \circ -$ is a map from L^σ_ω to L^τ_ω. Set

$$\alpha_1 \;=\; \beta^{\tau-1} \circ f \circ \beta^\sigma, \qquad \alpha_B \;=\; \beta^{\tau-1}_\omega \circ f^- \circ \beta^\sigma_\omega.$$

It is routine to check that this defines a natural transformation $\alpha \in M_{\sigma\to\tau}(1)$. Moreover, the above constructions are mutually inverse, since any $\alpha : M_\sigma \to M_\tau$ is uniquely determined by α_1, by naturality at each $x : 1 \to B$. We thus have the required bijection $\beta^{\sigma\to\tau}$.

We now establish the bijection $\beta^{\sigma\to\tau}_\omega : M_{\sigma\to\tau}(B) \cong L^{\sigma\to\tau}_\omega$. Suppose $\alpha \in M_{\sigma\to\tau}(B)$; then α is a natural transformation $h_B \times M_\sigma \to M_\tau$, so in particular we have $\alpha_1 : \mathrm{N}_\perp^{\mathrm{N}} \times M_\sigma(1) \to M_\tau(1)$ and $\alpha_B : \mathrm{Hom}(B,B) \times M_\sigma(B) \to M_\tau(B)$. Let $f : \mathrm{N}_\perp^{\mathrm{N}} \to L^{\tau L^\sigma}$ be the function obtained from α_1 by currying and composing with the given bijections. First note that for all $x \in \mathrm{N}_\perp^{\mathrm{N}}$ and all $g \in L^\sigma_\omega$, by the naturality of α at each $y : 1 \to B$ we have that $\alpha_B(k_x, g) = f(x) \circ g \in L^\tau_\omega$, where k_x is the constant morphism $B \to B$ corresponding to x. (Here, for clarity, we have omitted a few βs, as we shall generally do from now on.) Thus $f(x) \in L^{\sigma\to\tau}$ for each x. Next, we wish to show that $f \in L^{\sigma\to\tau}_\omega$. Given any $g \in L^\sigma_\omega$ and any $e : B \to B$, by naturality of α at each $y : 1 \to B$ we have that $\alpha_B(e, g) = (\lambda r.\, f(er)(gr)) \in L^\tau_\omega$. Now note that *fst*, *snd* are sequential functions and so correspond to morphisms $B \to B$. Note also that L^σ_ω is closed under precomposition with sequential functions $\mathrm{N}_\perp^{\mathrm{N}} \to \mathrm{N}_\perp^{\mathrm{N}}$ since this corresponds to the action of M_σ on morphisms $B \to B$. Hence, given any $g' \in L^\sigma_\omega$, setting $e = \mathit{fst}$ and $g = g' \circ \mathit{snd}$, we have that $(\lambda r.\, f(\mathit{fst}\ r)(g'(\mathit{snd}\ r))) \in L^\tau_\omega$; thus $f \in L^{\sigma\to\tau}_\omega$.

Conversely, suppose $f \in L^{\sigma\to\tau}_\omega$. Reversing the above argument, given any $g \in L^\sigma_\omega$ and any $e : B \to B$, there is a morphism $d : B \to B$ given by $r \mapsto \langle er, r\rangle$. Since $(\lambda r.\, f(\mathit{fst}\ r)(g(\mathit{snd}\ r))) \in L^\tau_\omega$ and L^τ_ω is closed under precomposition with d, we have that $(\lambda r.\, f(er)(gr)) \in L^\tau_\omega$. So define

$$\alpha_1(x, a) \;=\; f(x)(a), \qquad \alpha_B(e, g) \;=\; \lambda r.\, f(er)(gr).$$

It is routine to check that this defines a natural transformation $\alpha \in M_{\sigma\to\tau}(B)$.

Once again, the above constructions are mutually inverse since α_B is uniquely determined by α_1. This gives us our bijection $\beta_\omega^{\sigma\to\tau}$. The remaining clauses of the proposition are now immediate from the construction of $\beta^{\sigma\to\tau}$ and $\beta_\omega^{\sigma\to\tau}$. $\Box$

As usual, we can now obtain a type structure by taking global elements. For each σ let $P^\sigma = \mathrm{Hom}(1, M_\sigma)$, and consider the P^σ as equipped with the functions $P^{\sigma\to\tau} \times P^\sigma \to P^\tau$ induced by the evaluation morphisms in $[\mathcal{C}^{op}, \mathbf{Set}]$. Clearly we may identify $P^{\overline{0}}$ with $\mathrm{N}_\perp$, so this defines a type structure P. From the above proposition we easily deduce the following:

Corollary 6.3 *$P \cong R$.*

Proof The object 1 of $[\mathcal{C}^{op}, \mathbf{Set}]$ is just the constant functor $\{*\}$, which clearly coincides with the representable functor $h_1 = \mathrm{Hom}(-, 1)$. So by the Yoneda Lemma, functors $1 \to M_\sigma$ correspond naturally to elements of $M_\sigma(1) \cong L^\sigma$. Moreover, it is easy to see that the bijections $P^\sigma \cong L^\sigma$ respect the application operations, so $P \cong L$ as type structures. Combining this with Theorem 6.1 and Corollary 5.8, we have that $P \cong R$. $\Box$

The above characterization of the SR functionals using the category $\mathcal{C}$ was chosen to mesh well with the definition of L given in [13]. However, we can now dispense with the object 1 in $\mathcal{C}$ to obtain a more aesthetically pleasing description. Let $\mathcal{M}$ be the monoid of sequential endofunctions on $\mathrm{N}_\perp^{\mathrm{N}}$, as in Remark 4.9(ii), and let I be the full inclusion $\mathcal{M} \hookrightarrow \mathcal{C}$. It is easy to show the following:

Proposition 6.4 *The categories $[\mathcal{C}^{op}, \mathbf{Set}]$ and $[\mathcal{M}^{op}, \mathbf{Set}]$ are equivalent.*

Proof This follows trivially from the fact that 1 arises as an absolute coequalizer of two morphisms $B \rightrightarrows B$ within $\mathcal{C}$, and the Yoneda embedding exhibits $[\mathcal{M}^{op}, \mathbf{Set}]$ as the *free cocompletion* of $\mathcal{M}$ (see e.g. [36, Section 1.5]). $\Box$

Under this equivalence, the object $M_{\overline{0}}$ corresponds to the right $\mathcal{M}$-set X whose carrier is the set of sequential functions $\mathrm{N}_\perp^{\mathrm{N}} \to \mathrm{N}_\perp$, with right $\mathcal{M}$-action given by precomposition. The type structure generated from X in $[\mathcal{M}^{op}, \mathbf{Set}]$ thus gives a pleasing mathematical characterization of the SR functionals.

Remarks 6.5 (i) The object X bears very little relation to the natural number object in $[\mathcal{M}^{op}, \mathbf{Set}]$, which is a much more boring $\mathcal{M}$-set with a trivial $\mathcal{M}$-action.

(ii) In the light of the universal property mentioned above, $[\mathcal{M}^{op}, \mathbf{Set}]$ is in some sense a "canonical" extension of $\mathcal{M}$ to a cartesian closed category. The above result (and indeed the result of Colson and Ehrhard) therefore suggests that the SR functionals are in some way a canonical extension of the notion of infinitary first-order sequentiality (embodied by $\mathcal{M}$) to higher types.

(iii) The infinite arity of the functions embodied by $\mathcal{M}$ is quite essential here. If we work with presheaves over the category of *finitary* Milner-Vuillemin sequential functions, there is nothing to impose continuity at higher types.

(iv) There seems to be some analogy between $[\mathcal{M}^{op}, \mathbf{Set}]$ and its relation to the realizability model over $\mathcal{B}$, and Mulry's *recursive topos* [39] and its relation to the effective topos. However, we have not explored this connection.

7 A universal type

We now come to the main new result of the paper. We show that in R the type $\overline{2}$ is a *universal* type, in the sense that every type σ is a *retract* of $\overline{2}$. More specifically, for every type σ there are SR functionals $R^{\sigma} \leftrightarrows R^{\overline{2}}$ that compose to give the identity on $R^{\overline{2}}$; thus, $R^{\overline{2}}$ in some sense "contains" all higher types. The same also holds for R_{eff}. This gives us a very good grasp of these type structures, and numerous other results then follow fairly easily.

For ease of presentation, at this point in the paper we will shift the focus of our attention to the *call-by-value* SR functionals, rather than the call-by-name ones we have studied so far. Mathematically this does not make a big difference, and one can easily recover the call-by-name version of our results from the call-by-value version. In Section 7.1 we explain precisely the call-by-value interpretation of types that we are considering.

We then show in Sections 7.2 and 7.3 that the (call-by-value) type $\overline{3}$ is a retract of $\overline{2}$—it follows easily from this that *every* type is a retract of $\overline{2}$. The required morphism $I : \overline{3} \to \overline{2}$ is easy to construct, and the bulk of the work involves the morphism $H : \overline{2} \to \overline{3}$. We give the construction of H in Section 7.2, and the verification that $I \circ H = \mathrm{id}$ in Section 7.3, along with other facts.

Some applications of this result will be given in Sections 8 and 9.

7.1 Call-by-value types

So far we have been considering the *call-by-name* interpretation of types in $\mathbf{Mod}(\mathcal{B})$ as given by Definition 3.5 (we will write this interpretation here as $[\![-]\!]_N$, and the corresponding type structure as R_N). At this point in the paper we switch our attention to the *call-by-value* interpretation $[\![-]\!]_V$, defined essentially as follows:

$$[\![\overline{0}]\!]_V \cong N, \qquad [\![\sigma_1 \to \sigma_2]\!]_V \cong ([\![\sigma_2]\!]_{V\bot})^{[\![\sigma_1]\!]_V}.$$

By taking $R^{\sigma}_V = |[\![\sigma]\!]_V|$ we obtain a *call-by-value type structure*, that is, a family of sets R^{σ}_V together with *partial* application functions $\cdot : R^{\sigma \to \tau}_V \times R^{\sigma}_V \rightharpoonup R^{\tau}_V$. Similarly we define the effective call-by-value interpretation $[\![-]\!]_{V,\mathit{eff}}$ in $\mathbf{Mod}(\mathcal{B}_{\mathit{eff}})$, and the corresponding call-by-value type structure $R_{V,\mathit{eff}}$. (These turn out to be the appropriate objects for giving interpretations of call-by-value PCF—see Section 9.1 below.)

The reason for this shift is that the functional H lives most naturally in the call-by-value setting. We emphasize, however, that this is really just a matter of convenience. Indeed, the type structures R_N and R_V may each be recovered from the other, since each object $[\![\sigma]\!]_V$ can be defined as a retract of some $[\![\tau]\!]_N$, and *vice versa* (see e.g. [31, Chapter 6]). This means that the results we are interested in will transfer easily between the two settings. We can therefore regard R_N and R_V as being in some sense just different presentations of the same thing.

For the rest of this section, we will write $[\![-]\!]$ for $[\![-]\!]_V$, and R for R_V. Since most of the theorems and proofs in this section work identically in the full and effective settings, we will use undecorated notation when we could be referring to either case, and use the subscripts *full* and *eff* when we need to be specific. Thus, $\mathcal{B}$ may refer to either $\mathcal{B}_{full}$ or $\mathcal{B}_{eff}$.

The interpretation $[\![-]\!]$ (in both the full and effective cases) is given up to isomorphism by the above definition, but it will greatly simplify our task to work with judiciously chosen on-the-nose representations of these objects. Specifically, we take $[\![\overline{0}]\!]$ to be the object N as in Section 3.1, $[\![\overline{1}]\!] \cong N_\perp^N$ to be the object of realizers B given by Definition 3.4. For our representation of $[\![\overline{2}]\!]$, we need the following *ad hoc* definition:

Definition 7.1 (Semi-irredundant realizers) *We call an element* $r \in \mathcal{B}$ semi-irredundant *if, for all* $\alpha, \beta \in \mathrm{Seq(N)}$,

- *if* $r\langle\alpha\rangle$ *is defined then it is a question or an answer;*
- *if* $r\langle\alpha\rangle$ *is defined and* β *is a prefix of* α *then* $r\langle\beta\rangle$ *is defined;*
- *if* β *is a prefix of* α *and* $r\langle\alpha\rangle = r\langle\beta\rangle = ?n$ *then* $\alpha = \beta$.

Comparing this with Definition 2.9(i), any $|$-irredundant element is clearly semi-irredundant, but a semi-irredundant element may have inaccessible nodes underneath answer nodes. Note that if r is semi-irredundant then $irr_|(r) \sqsubseteq r$. (In this section, unlike in Section 4, we will use "irredundant" without qualification to mean $|$-irredundant.)

We now define $[\![\overline{2}]\!]$ as follows:

$$\begin{array}{rcl} |[\![\overline{2}]\!]| & = & \{F \mid F \text{ is a morphism } B \to N_\perp\} \\ \|F\|_{[\![\overline{2}]\!]} & = & \{r \mid r \text{ is semi-irredundant } \wedge \forall g \in B.\ r \mid g = F(g)\}. \end{array}$$

(Our reasons for choosing this semi-irredundant representation will become more apparent in Section 7.2 below.) It is easy to check that $[\![\overline{2}]\!] \cong N_\perp^B$. Note that *every* semi-irredundant realizer belongs to some $\|F\|_X$; in fact, we may define an equivalence relation $\sim$ on semi-irredundant realizers by setting $r \sim r'$ iff $r, r' \in \|F\|_X$ for some F.

For $[\![\overline{3}]\!]$ we take the following object:

$$\begin{array}{rcl} |[\![\overline{3}]\!]| & = & \{\Phi \mid \Phi \text{ is a morphism } [\![\overline{2}]\!] \to N_\perp\} \\ \|\Phi\|_{[\![\overline{3}]\!]} & = & \{r \mid \forall F \in [\![\overline{2}]\!].\ \forall t \in \|f\|.\ r \mid t = \Phi(F)\}. \end{array}$$

For all other types $\sigma \to \tau$ we may take $[\![\sigma \to \tau]\!] = [\![\tau]\!]_\perp^{[\![\sigma]\!]}$ as usual. This completes the definition of $[\![-]\!]$.

We remark briefly on the relationship between the full and effective interpretations of these types. Clearly $[\![\overline{0}]\!]_{full}$ and $[\![\overline{0}]\!]_{eff}$ are identical, and $[\![\overline{1}]\!]_{eff}$ can be identified with a subobject of $[\![\overline{1}]\!]_{full}$. The observation above that every semi-irredundant r realizes some $F \in [\![\overline{2}]\!]$ (valid in both the full and effective settings) yields the following easy facts: every $F \in [\![\overline{2}]\!]_{eff}$ extends uniquely to an element $\hat{F} \in [\![\overline{2}]\!]_{full}$, and every realizer for F is also a realizer for $\hat{F}$. The relationship between $[\![\overline{3}]\!]_{eff}$ and $[\![\overline{3}]\!]_{full}$ is far from obvious, but it will be established in Section 8.

The call-by-value types (apart from $\overline{0}$) also have counterparts in the hypercoherence model. Indeed, for $\sigma, \tau \neq \overline{0}$, we may define

$$\begin{array}{rcl}
[\![\, \overline{0} \to \overline{0} \,]\!]_{HC} & = & N^{\omega}, \\
[\![\, \sigma \to \overline{0} \,]\!]_{HC} & = & ([\![\, \sigma \,]\!]_{HC} \Rightarrow N), \\
[\![\, \sigma \to \tau \,]\!]_{HC} & = & ([\![\, \sigma \,]\!]_{HC} \Rightarrow [\![\, \tau \,]\!]_{HC\perp}),
\end{array}$$

where N^{ω} is as defined in Section 6. We will consider this interpretation in Section 8 below.

7.2 Construction of H

We now work towards the main result: in both $\mathbf{Mod}(\mathcal{B}_{\mathit{full}})$ and $\mathbf{Mod}(\mathcal{B}_{\mathit{eff}})$, the object $[\![\, \overline{3} \,]\!]$ is a retract of $[\![\, \overline{2} \,]\!]$. As we shall see, the construction of the retraction is directly inspired by the combinatory algebra $\mathcal{B}$.

The easy half of the retraction will be the morphism $I : \overline{3} \to \overline{2}$ defined as follows. Let $\mathit{bar} : B \times B \to N_{\perp}$ be the morphism defined by $(f, g) \mapsto f \mid g$, as in Section 3.1. Currying this, we obtain a morphism $B \to N_{\perp}^{B}$, and hence a morphism $b : [\![\, \overline{1} \,]\!] \to [\![\, \overline{2} \,]\!]$, mapping f to $f \mid -$. Now let $I : [\![\, \overline{3} \,]\!] \to [\![\, \overline{2} \,]\!]$ be the morphism induced by precomposition with b: that is,

$$I \;=\; - \circ b : \; ([\![\, \overline{2} \,]\!] \Rightarrow [\![\, \overline{0} \,]\!]_{\perp}) \;\longrightarrow\; ([\![\, \overline{1} \,]\!] \Rightarrow [\![\, \overline{0} \,]\!]_{\perp}).$$

Note that b is realized by the element $\mathbf{irr}_{|}$ of Section 2.3, and I itself is realized by $\imath = \lambda^{*}xy.\, x(\mathbf{irr}_{|}\, y)$.

The other half of the retraction will be a morphism $H : [\![\, \overline{2} \,]\!] \to [\![\, \overline{3} \,]\!]$ such that $H \circ I = \mathrm{id}_{[\![\overline{3}]\!]}$. Since the construction of H is rather involved, we begin with an informal explanation.

We think of the type $\overline{1}$ as the object B, and $\overline{2}$ as the object B equipped with the partial equivalence relation $\sim$ defined above. Any element of $\mathcal{B}$ will realize some morphism $B \to N_{\perp}$ (that is, $\overline{1} \to \overline{0}$), since each element of B has just one realizer; but an element r of $\mathcal{B}$ will realize a morphism $\overline{2} \to \overline{0}$ only if r acts extensionally on $\sim$-classes. The morphism I essentially takes a morphism $(B, \sim) \to N_{\perp}$ and treats it as a morphism $B \to N_{\perp}$, just by forgetting the equivalence relation.

The morphism H, on the other hand, takes a morphism $F : B \to N_{\perp}$, given by an arbitrary realizer r, and converts it into a morphism $\mathit{ext}(F) : B \to N_{\perp}$ that acts extensionally with respect to $\sim$, and can hence be treated as a morphism $(B, \sim) \to N_{\perp}$. In some sense, $\mathit{ext}(F)$ will be the closest approximation to F that is $\sim$-extensional.

We define $\mathit{ext}(F)$ from F in something like the following way: for any $r \in \mathcal{B}$ we take $\mathit{ext}(F)(r) = n$ if $F(r') = n$ whenever $r' \sim r$; if there is no such n we take $\mathit{ext}(F)(r) = \perp$. (This is not quite right, but it gives the basic idea). Thus, $\mathit{ext}(F)$ "homogenizes" F with respect to $\sim$. However, it is not immediately clear that $\mathit{ext}(F)(r)$ can be computed by a sequential algorithm: it seems, on the face of it, we would need to test infinitely many realizers $r' \sim r$. The key to the proof is the observation that in some sense there are *essentially* only finitely many such realizers r', and so it suffices to test a finite set of realizers that are representative of all of them.

We need to show, of course, that this testing can be done within $\mathcal{B}$, so that we obtain a realizer for $ext(F)$. Finally, we show that the passage from F to $ext(F)$ can itself be performed within $\mathcal{B}$, uniformly in F, so that we obtain a realizer for the morphism H.

We approach the formal construction of H by first establishing some facts about the set of realizers for an arbitrary type 2 function F. For any $F \in [\![\overline{2}]\!]$, we define its *trace*, written $tr\ F$, to be the set of all pairs (g, Fg) where $g : \mathrm{N} \rightharpoonup \mathrm{N}$ is a finite partial function (given as a finite set of ordered pairs) which is minimal such that Fg is defined. (Of course, this is exactly the trace in the sense of the hypercoherence model, although we will not need this fact.) Clearly F is determined by $tr\ F$, since $Fh = n$ iff $(g, n) \in tr\ F$ for some $g \subseteq h$. Moreover, if p is any irredundant realizer for F, there is a bijective correspondence between elements of $tr\ F$ and nodes α such that $p\langle\alpha\rangle$ is an answer. Explicitly, $(g, Fg) \in tr\ F$ corresponds to α if $p\langle\alpha\rangle = !Fg$ and whenever β, m is a prefix of α and $p\langle\beta\rangle = ?n$ we have $g(n) = m$. (Note that in this situation the size of dom g is exactly the length of α).

We write $F \sqsubseteq F'$ if $tr\ F \subseteq tr\ F'$. We say F is *finite* if $tr\ F$ is finite. The following observations now provide the key to the whole construction:

Lemma 7.2 *Suppose $F \in [\![\overline{2}]\!]$ is finite. Then*

(i) F has only finitely many irredundant realizers, each with finite domain;

(ii) any realizer $p \in \|F\|$ extends exactly one of these irredundant realizers;

(iii) the set of minimal realizers can be effectively computed (as a finite list of finite graphs) from $tr\ F$.

PROOF (i) Clearly, if p is an irredundant realizer for F then dom p consists of all $\langle\beta\rangle$ such that β is a prefix of some α with $p\langle\alpha\rangle$ an answer. But by the above remarks, if F is finite then there are just finitely many such α; hence dom p is finite. Moreover, for each $(g, Fg) \in tr\ F$, p induces a total order $\preceq^g_p$ on dom g, where $n \preceq^g_p n'$ iff there exist proper prefixes β, β' of α with $p\langle\beta\rangle = n$, $p\langle\beta'\rangle = n'$ and β a prefix of β'. Clearly p may be recovered from $tr\ F$ together with the orders $\preceq^g_p$ as follows: if the elements of dom g in the order $\preceq^g_p$ are precisely $n_1, \ldots, n_l$ then we have

$$p\langle gn_1, \ldots, gn_i\rangle \;=\; ?n_{i+1} \;\text{ for } 0 \leq i < l; \qquad p\langle gn_1, \ldots, gn_l\rangle \;=\; !Fg.$$

But since there are only finitely many combinations of total orders $\preceq^g_p$ on the sets dom g, there can be only finitely many irredundant realizers p.

(ii) If p is any semi-irredundant realizer for F then $irr_{|}(p)$ is an irredundant realizer for F, and p extends $irr_{|}(p)$. Furthermore, if q is any irredundant realizer for F such that p extends q, then $irr_{|}(q) = q$ and $irr_{|}(p)$ extends $irr_{|}(q)$; hence clearly $q = irr_{|}(p)$.

(iii) Given $tr\ F$ (where for each element (g, Fg), g is given as a finite list of pairs), we may effectively compute the finite set of all possible families of orderings $(\preceq^g \mid (g, Fg) \in tr\ F)$ in which each $\preceq^g$ is a total order on dom g.

Moreover, for each such family $(\preceq^g)$, we may effectively determine whether the $\preceq^g$ all arise from some irredundant realizer p (this happens provided the above formulae for p in terms of the $\preceq^g$ do not give rise to clashes). If the $\preceq^g$ do all arise from some p, we may effectively compute the graph of p itself according to the above formulae. Thus we may effectively compute the finite set of (graphs of) irredundant realizers for F. $\Box$

The force of this proposition is that we may find a finite set of realizers for F that is representative of all of them. Part (ii) depends on the fact that our realizers for F are already semi-irredundant rather than arbitrary elements that track F. Note in passing that the set of irredundant realizers for a finite F is exactly the same in the full and effective settings.

The next lemma establishes a useful connection between the trace ordering on type 2 functions and the pointwise ordering on their realizers.

Lemma 7.3 *Suppose given $F, G \in [\![\, \overline{2} \,]\!]$ such that $F \sqsubseteq G$. Then for any irredundant realizer $q \in \|G\|$ there is an irredundant realizer $p \in \|F_0\|$ such $p \sqsubseteq q$. Moreover, if F is finite then p can be chosen to be finite (and hence effective).*

Proof Recall that the answer nodes of q correspond bijectively to the elements of *tr* G. Let p be the restriction of q to arguments $\langle \beta \rangle$ where β is a prefix of some answer node α corresponding to an element of *tr* F. Clearly $p \subseteq q$, and it is easy to see that p tracks F. Moreover, if F is finite then there are finitely many answer nodes corresponding to elements of *tr* F; hence p has finite domain. $\Box$

We are now ready to construct the morphism H. We proceed in several steps, beginning with the following definition:

Definition 7.4 *(i) Suppose $F, G_0 \in [\![\, \overline{2} \,]\!]$ with G_0 finite. We say that F* uniformly yields n on G_0 *if, for each irredundant realizer $p_0 \in \|G_0\|$, for all $q \sqsubseteq p_0$ in $\mathcal{B}$ we have $Fq = n$ iff $q = p_0$.*

(ii) We provisionally define a function $HF : |[\![\, \overline{2} \,]\!]| \to |[\![\, \overline{0} \,]\!]|$ as follows: For each $G \in [\![\, \overline{2} \,]\!]$, set $HF(G) = n$ iff there is some finite $G_0 \sqsubseteq G$ such that F uniformly yields n on G_0.

To ensure that part (ii) of the above definition is sound, we need to show that G_0, if it exists, is unique. This and other useful information is given by the following proposition.

Proposition 7.5 *Suppose given $F, G \in [\![\, \overline{2} \,]\!]$, p an irredundant realizer for G, r an irredundant realizer for F, and some finite $G_0 \sqsubseteq G$ such that F uniformly yields n on G_0. Then*

(i) $r \mid p = n$;

(ii) if p_0 is the smallest subfunction of p such that $r \mid p_0 = n$, then p_0 is an irredundant realizer for G_0;

(iii) if $G_1 \sqsubseteq G$ is finite and F uniformly yields n on G_1, then $G_1 = G_0$.

Proof (i) By Lemma 7.3 we may choose $p' \sqsubseteq p$ such that p' is an irredundant realizer for G_0. But then $Fp' = n$, so $r \mid p' = n$, whence $r \mid p = n$.

(ii) The above realizer $p' \sqsubseteq$ has the property that for all $q \sqsubseteq p'$ in $\mathcal{B}$ we have $Fq = n$ iff $q = p'$. So if p_0 is as given then $p_0 = p'$; thus p_0 is an irredundant realizer for G_0.

(iii) The realizer p_0 is defined without reference to G_0. So if G_1 is as given then p_0 realizes both G_0 and G_1, so $G_0 = G_1$. $\Box$

Proposition 7.6 *For each $F \in [\![\,\overline{2}\,]\!]$, the function HF defined above is realizable, i.e. is an element of $[\![\,\overline{3}\,]\!]$.*

Proof Let r be an irredundant realizer for F. Note that if F uniformly yields n on some G_0 and p_0 is an irredundant realizer for G_0, then the answer node of r that produces the result of the computation of $r \mid p_0$ corresponds to the element (γ, n) of $tr\ F$, where γ is the graph of p_0.

This suggests that we restrict r to a realizer r' defined as follows. If α is an answer node of r corresponding to an element $(\gamma, n) \in tr\ F$, where γ is the graph of some irredundant realizer p_0 for some finite $G_0 \in [\![\,\overline{2}\,]\!]$, then take $r'\langle\alpha\rangle$ to be $!n$ if F uniformly yields n on G_0, $\bot$ otherwise. If $a \in \mathrm{N}$ is not of the form $\langle\alpha\rangle$ for such a node α, then take $r'a = ra$. Clearly r' is irredundant, and from the above observation it is easy to see that r' tracks HF.

For the effective case, we also need to check that if r is effective then so is r'. Suppose r is effective and irredundant; then given a such that $ra = !n$ we must have $a = \langle\alpha\rangle$ for some α. Now given an answer node α of r we may effectively compute the values of $r\langle\beta\rangle$ for all prefixes β of α; from this we may effectively obtain the element (γ, n) corresponding to α. We may then effectively decide whether γ is the graph of an irredundant element p_0; if so, p_0 necessarily realizes some finite G_0, and we may effectively obtain $tr\ G_0$ from γ. By Lemma 7.2, from $tr\ G_0$ we may effectively compute the finite list of all irredundant realizers for G_0; and since r is effective, it is semi-decidable whether F uniformly yields n on G_0. Combining all this information we see that r' is effective. $\Box$

We have thus defined a set-theoretic function $H : |[\![\,\overline{2}\,]\!]| \to |[\![\,\overline{3}\,]\!]|$. Note that the definitions of H in the full and effective settings agree: given $F \in [\![\,\overline{2}\,]\!]_{\mathit{full}}$ such that F has an effective realizer and so restricts to an element $F_{\mathit{eff}} \in [\![\,\overline{2}\,]\!]_{\mathit{eff}}$, the element $H_{\mathit{eff}} F_{\mathit{eff}}$ coincides with the restriction of $H_{\mathit{full}} F$ to $[\![\,\overline{2}\,]\!]_{\mathit{eff}}$.

The last step in the construction is to show that the function H is itself realizable. In fact, exactly the same realizer will work in the full and effective settings. In the light of the above remarks, it suffices to show:

Proposition 7.7 *There is an element $h \in \mathcal{B}_{\mathit{eff}}$ such that, for all $F \in [\![\,\overline{2}\,]\!]_{\mathit{full}}$, if $r \in \|F\|$ then $h \bullet r \in \|HF\|$.*

Proof Essentially this amounts to showing that the construction of r' from r in the previous proof can be carried out within $\mathcal{B}_{\mathit{eff}}$ itself. However, since this construction works only on irredundant realizers and we require h to work on all semi-irredundant realizers, we had better define $h = \lambda^* x. h'(\mathbf{irr}_{|} x)$, where h' embodies the above construction.

A completely formal definition of h' would be unilluminating, so we content ourselves with an "anthropomorphic" description of the algorithm embodied by the ath decision tree in the forest represented by h', for an arbitrary $a \in \mathrm{N}$.

We first ask the question a (that is, $h'\langle a \rangle = ?a$). Suppose this receives the answer b. If either a is not of the form $\langle \alpha \rangle$ or b is not of the form $!n$, we simply return b as our final result (that is, $h'\langle a, b \rangle = !b$). Otherwise, we ask the question $\langle \beta \rangle$ for every prefix β of α in turn. From the answers received so far, we now "know" the element (γ, n) in the previous proof. If γ is not the graph of an irredundant realizer p_0, we simply give up and diverge. Otherwise, we now know p_0 and hence the trace of the functional G_0 it realizes. Let $p_1, \ldots, p_N$ be the list of all irredundant realizers of G_0, which we now know. We wish to check whether our argument r satisfies $r \mid p_i = n$ for each i. To do this, we first simulate the play of r against p_1, by asking questions to r, and ourselves providing the answers to the questions r asks using our knowledge of p_1. If in fact $r \mid p_1 = n'$, at some point we will receive the answer $!n'$. If $n' \neq n$, we again hang up and diverge. If $n' = n$, we then proceed to the simulation of r against p_2, and so on. Finally, if we complete the simulation of r against each of the p_i and have received the answer n in every case, we return n as our final result.

It should be clear that h' may be given effectively, and that if r is an irredundant realizer for F then $h' \bullet r$ is the realizer r' for HF described in the previous proof. It follows that h is also effective, and tracks H as required. $\Box$

This completes the construction of H.

7.3 Properties of H

We next show that H is indeed a one-sided inverse to I. For this, we need to establish some special properties of realizers for elements $\Phi \in [\![\, \overline{3} \,]\!]$. By definition, r is a realizer for some such Φ iff r acts *extensionally* on semi-irredundant realizers, that is, $r \mid p = r \mid q$ whenever $p \sim q$.

Proposition 7.8 *Suppose r realizes $\Phi \in [\![\, \overline{3} \,]\!]$, p realizes $F \in [\![\, \overline{2} \,]\!]$, and $\Phi F = n \in \mathrm{N}$. Let p_0 be the unique smallest finite subfunction of p such that $r \mid p_0 = n$. Then*

(i) dom p_0 *is* prefix-closed, *i.e. if $\langle \alpha \rangle \in$* dom p_0 *and β is a prefix of α then $\langle \beta \rangle \in$* dom p_0*;*

(ii) p_0 realizes some $F_0 \in [\![\, \overline{2} \,]\!]$ such that $\Phi F_0 = n$;

(iii) for all α, $p_0\langle \alpha \rangle$ is an answer iff α is a leaf in p_0 (i.e. α is not a proper prefix of any β where $\langle \beta \rangle \in$ dom p_0*).*

Proof (i) Extend p_0 to a function p_1 by setting $p_1\langle \beta \rangle = !0$ whenever $\beta \notin \mathrm{dom}\ p_0$ is a prefix of some $\alpha \in \mathrm{dom}\ p_0$. Then dom p_1 is finite and prefix-closed, and since p was semi-irredundant, so is p_1. Thus p_1 realizes some $F_1 \in [\![\, \overline{2} \,]\!]$. Furthermore, since $p_0 \sqsubseteq p_1$ we have $r \mid p_1 = n$, so $\Phi F_1 = n$. Now let $p_2 = irr_{|}(p_1)$; then p_2 also realizes F_1, and so $r \mid p_2 = n$ since r realizes Φ. Thus, we have $p_0, p_2 \sqsubseteq p_1$ with $r \mid p_0 = r \mid p_2 = n$; hence $p_0 \sqsubseteq p_2$ by minimality of p_0. But if there exists $\alpha \in \mathrm{dom}\ p_0$ with a prefix $\beta \notin \mathrm{dom}\ p_0$ then $\langle \alpha \rangle \notin \mathrm{dom}\ p_2$, a contradiction. So dom p_0 is already prefix-closed, and $p_1 = p_0$.

(ii) Since p is semi-irredundant, it now follows that p_0 is semi-irredundant, and so realizes some $F_0 \in [\![\overline{2}]\!]$ where $\Phi F_0 = n$.

(iii) By (i), $p_4 = irr_{|}(p_0)$ also realizes F_0, and so $r \mid p_4 = n$. Since $p_4 \sqsubseteq p_0$, by minimality of p_0 we have $p_4 = p_0$, so p_0 is irredundant. In particular, if $p_0\langle\alpha\rangle$ is an answer then α is a leaf in dom p_0. For the converse, suppose α is a leaf but $p_0\langle\alpha\rangle$ is a question. Let p_5 be obtained as a restriction of p_0 simply by removing the value at $\langle\alpha\rangle$. Clearly p_5 also realizes F_0, and so $r \mid p_5 = n$, contradicting the minimality of p_0. $\Box$

Part (iii) of the above proposition is not actually required below, but it provides some additional insight into the situation. The proof of part (i) shows why we need the notion of semi-irredundant realizer. If we assumed only that $r \mid p = n$ for all *irredundant* realizers p of F, we would not be able to show in the above proof that $r \mid p_2 = r \mid p_1$. On the other hand, if we took as $\|F\|$ the set of *all* p tracking F, we would lose the property that every realizer for a finite F extends one of the finitely many irredundant realizers (Lemma 7.2(ii)).

Theorem 7.9 *For the morphisms*

$$[\![\overline{3}]\!] \underset{H}{\overset{I}{\rightleftarrows}} [\![\overline{2}]\!]$$

defined above, we have $H \circ I = \mathrm{id}_{[\![\overline{3}]\!]}$.

PROOF Suppose $\Phi \in [\![\overline{3}]\!]$, and take any irredundant $r \in \|I\Phi\|$. It is easy to see that $r \in \|\Phi\|$: take any $G \in [\![\overline{2}]\!]$ and note that for any $g \in \|G\|$ we have

$$r \mid g \;=\; (I\Phi)(g) \;=\; \Phi(\lambda f.\, g \mid f) \;=\; \Phi G.$$

Let r' be the realizer for $H(I\Phi)$ obtained from r as in the proof of Proposition 7.6. To show that $H(I\Phi) = \Phi$ it will suffice to show that $r' = r$. So suppose α is an answer node of r corresponding to $(\gamma, n) \in tr\; I\Phi$, where γ is the graph of an irredundant realizer $p_0 \in \|G_0\|$ for some finite G_0. We wish to know that $r'\langle\alpha\rangle = r\langle\alpha\rangle$ (note that we automatically have $r'a = ra$ for all a not of this form). But $r\langle\alpha\rangle = !n$, and so to show that $r'\langle\alpha\rangle = !n$ it is enough to show that $I\Phi$ uniformly yields n on G_0.

Suppose q_0 is any irredundant realizer for G_0. Since r realizes Φ, we have $r \mid q_0 = \Phi G_0 = r \mid p_0 = n$. Now let q_1 be the least subfunction of q_0 such that $(I\Phi)q_1 = n$—that is, such that $r \mid q_1 = n$. Then by Proposition 7.8(ii), q_1 is an irredundant realizer for some $G_1 \sqsubseteq G_0$ such that $\Phi G_1 = n$. But now by Lemma 7.3 we may find a realizer $p_1 \in \|G_1\|$ such that $p_1 \sqsubseteq p_0$. Since r realizes Φ, we have $r \mid p_1 = n$. But p_0 is clearly minimal such that $r \mid p_0 = n$, and so $p_1 = p_0$. Hence $G_1 = G_0$, and since q_0 is a minimal realizer for G_0 we have $q_1 = q_0$. So q_0 is minimal such that $(I\Phi)q_0 = n$; thus $I\Phi$ uniformly yields n on G_0 as required. $\Box$

Remark 7.10 It is natural to ask whether H (or some other one-sided inverse to I) can be characterized by a universal property involving I—for instance, as an adjoint or Kan extension. At present, the answer would seem to be no. However, one *can* obtain a characterization of this kind if one decomposes I into two components.

Specifically, let Z be the modest set defined by

$$|Z| \;=\; \{F : B \to N_\perp \mid \forall g \in B.\, F(g) = F(irr_{|}(g))\}, \qquad \|F\|_Z \;=\; \|F\|_{[\![\overline{2}]\!]}.$$

Let $I_1 : Z \to [\![\,\overline{2}\,]\!]$ be the evident inclusion; it is easy to see that I_1 has a one-sided inverse H_1. Also I factors through I_1 via a morphism $I_0 : [\![\,\overline{3}\,]\!] \to Z$. Let H_0 be the restriction of H to Z; then we have $H_0 I_0 = \mathrm{id}$. The morphism $H_0 H_1$ is not exactly our morphism H, but at least it is a one-sided inverse to I. Moreover, one can characterize H_0 and H_1 abstractly as follows: H_0 is a right adjoint of I_0 with respect to the *stable* ordering, and H_1 is a right adjoint of I_1 with respect to the *pointwise* ordering.

It is perhaps surprising that no simpler characterization of a one-sided inverse to I is possible, if this is indeed the case.

It is now an easy step to show that *every* type is a retract of $\overline{2}$:

Theorem 7.11 (Universality of $\overline{2}$) *In* $\mathbf{Mod}(\mathcal{B})$*, every object* $[\![\,\sigma\,]\!]$ *is a retract of* $[\![\,\overline{2}\,]\!]$.

PROOF We first prove the result for all *pure* types $\overline{n}$. Clearly $[\![\,\overline{0}\,]\!], [\![\,\overline{1}\,]\!], [\![\,\overline{2}\,]\!]$ are all retracts of $[\![\,\overline{2}\,]\!]$. We have shown above that $[\![\,\overline{3}\,]\!] \lhd [\![\,\overline{2}\,]\!]$; and from a retraction $(i,j) : [\![\,\overline{n+1}\,]\!] \lhd [\![\,\overline{n}\,]\!]$ we obtain a retraction $(- \circ j, - \circ i) : [\![\,\overline{n+2}\,]\!] \lhd [\![\,\overline{n+1}\,]\!]$. By composing such retractions, we have $[\![\,\overline{n}\,]\!] \lhd [\![\,\overline{2}\,]\!]$ for all $n \geq 3$.

To extend the result to arbitrary types, we just need to know that every object $[\![\,\sigma\,]\!]$ is a retract of some $[\![\,\overline{n}\,]\!]$. This is a well known piece of folklore. For call-by-name types it is proved in [21, Section 8.1], and in fact the same proof also works for call-by-value types without any modification. $\Box$

Remark 7.12 If $(i,j) : [\![\,\sigma\,]\!] \lhd [\![\,\overline{2}\,]\!]$ is a retraction in $\mathbf{Mod}(\mathcal{B})$, it is immediate that i, j themselves live in the type structure R: essentially $i \in R^{\sigma \to \overline{2}}$ and $j \in R^{\overline{2} \to \sigma}$.

We thus have that for the call-by-value SR functionals, the type $\overline{2}$ is *universal* among simple types. Of course exactly the same result holds for the call-by-name SR functionals, and one way to see this is as follows. By the results of [31, Chapter 6], every object $[\![\,\sigma\,]\!]_N$ is a retract of some $[\![\,\tau\,]\!]_V$, and by the above results, this is in turn a retract of $[\![\,\overline{2}\,]\!]_V$, which is a retract of $[\![\,\overline{2}\,]\!]_N$.

It is interesting to compare our results with the situation for other known type structures. In both the full and effective Scott models, which correspond to a version of PCF with parallel operators, the type $\overline{1}$ is already universal. (This follows easily from the fact that every finite type is a definable retract of Plotkin's T^ω—see [34, Lemma 5.2].) However, in the full or effective games models, corresponding to pure PCF, there is *no* universal finite type.[4]

[4] I am grateful to Samson Abramsky and Martin Hyland for informing me of this fact.

Remark 7.13 We end this paragraph with some informal remarks on the computational complexity of H. Although we do not know what is the best way to measure complexity at higher types precisely, it is intuitively clear that the main factor contributing to the complexity of H is the number of irredundant realizers for G_0 that have to be tested in turn. In the worst case, this is factorial in the size of dom p_0 (call this n), and each irredundant realizer p_1 itself has size n. Moreover, n is essentially the number of queries made by F to the given realizer p for G. Thus, if we are given realizers r, p for F, G respectively, the time taken up by H in the computation of HFG is at most $t.t!$, where t is the time taken by F in the computation of Fp.

8 Applications of universality

One consequence of Theorem 7.11 is that, for many questions about the SR functionals, *it suffices to consider low types.* In this section we illustrate this principle by giving an alternative proof of van Oosten's theorem (Theorem 5.8). We also consider the category of retracts of $[\![\,\overline{2}\,]\!]$ as a category of domains, and indicate how one can interpret *recursive types* there. Further applications of the universality theorem will be given in the next section.

8.1 Hypercoherences revisited

We now use Theorem 7.11 to give an alternative proof that the full SR functionals coincide with the strongly stable functionals (cf. Section 5.2). Unlike the proofs in [17, 43], which both involve elaborate inductions up the type structure, our proof concentrates on the types $\overline{2}$ and $\overline{3}$; the extension to arbitrary types is then straightforward. Since the main interest of the proof is conceptual rather than technical, and result itself is not new, we concentrate on the main ideas, omitting some of the details.

The call-by-value interpretation $[\![\,\sigma\,]\!]_{HC}$ for types $\sigma \neq \overline{0}$ was given in Section 7.1. We define sets D^σ by

$$D^{\overline{0}} \;=\; \mathrm{N}, \qquad D^\sigma \;=\; D([\![\,\sigma\,]\!]_{HC}) \text{ for } \sigma \neq \overline{0}.$$

These come equipped with partial application operations $\cdot_{\sigma\tau} : D^{\sigma\to\tau} \times D^\sigma \rightharpoonup D^\tau$. Our goal is to establish bijections $\beta_\sigma : R^\sigma \cong D^\sigma$ that commute with application (note that R^σ here means $R^\sigma_{V,full}$). We will write $R^{\overline{n}}, D^{\overline{n}}, \beta_{\overline{n}}$ simply as R^n, D^n, β_n.

The bijections β_0 and β_1 are evident. For β_2, we proceed as follows. For each $k > 0$, let $[k]$ be the flat hypercoherence with atoms $0, \ldots, k-1$; then there are evident retractions $[k] \lhd N$ in **HC**, and the corresponding projections $N \to N$ form a chain whose limit is the identity. We also have retractions $N^k \lhd N^\omega$ corresponding to the projection from N^ω onto its first k factors. Using these, we can construct a chain of retractions $([k]^k \Rightarrow [k]) \lhd (N^\omega \Rightarrow N)$, with corresponding projections δ_k on $N^\omega \Rightarrow N$ where $\bigsqcup \rho_k = \mathrm{id}$. Note that each $\delta_k : D^2 \to D^2$ has a finite image consisting entirely of finite elements. We regard D^2 as equipped with the usual order on states of $[\![\,\overline{2}\,]\!]_{HC}$; the image of each δ_k is then a full sub-poset of D^2.

Likewise, in $\mathbf{Mod}(\mathcal{B})$ we have evident retracts $[k]_\perp \lhd N_\perp$ for each k, and so we can construct increasing sequence of retractions $\rho_k : R^2 \to R^2$ corresponding precisely to the δ_k. We regard R^2 as equipped with the trace ordering as in Section 7.2, so that the image of each ρ_k is a full sub-poset of R^2.

Proposition 8.1 *For each k there is an order-isomorphism* $\mathrm{Im}\ \delta_k \cong \mathrm{Im}\ \rho_k$. *Moreover, these isomorphisms commute with the retractions* $(\mathrm{Im}\ \delta_l) \lhd (\mathrm{Im}\ \delta_k)$ *and* $(\mathrm{Im}\ \rho_l) \lhd (\mathrm{Im}\ \rho_k)$ *for $l \leq k$, and with the application operations.*

PROOF We wish to show that the poset P_k of elements of $[k]_\perp^k \Rightarrow [k]_\perp$ in $\mathbf{Mod}(\mathcal{B})$ (with the trace ordering) is isomorphic to the poset Q_k of states of $[k]^k \Rightarrow [k]$ in $\mathbf{HC}$. But P_k is easily seen to be isomorphic to the poset S_k of Milner-Vuillemin sequential functions with the stable ordering (use the embedding $\mathbf{SSeqFun} \to \mathbf{Mod}(\mathcal{B})$ of Section 4.3). The bijection $S_k \cong Q_k$ is given by [9, Proposition 3], and this is an order-isomorphism since, for hypercoherences, the inclusion order on function spaces coincides with the stable order.

The isomorphisms $P_k \cong Q_k$ clearly respect application. They yield isomorphisms $\mathrm{Im}\ \rho_k \cong \mathrm{Im}\ \delta_k$, and the coherence conditions are easily verified. □

Let $D^2_{\mathit{fin}} = \bigcup \mathrm{Im}\ \delta_k$, $R^2_{\mathit{fin}} = \bigcup \mathrm{Im}\ \rho_k$. It is clear from the construction that D^2 is the chain-completion of the full sub-poset D^2_{fin}. We also know that any element $x \in R^2$ is a least upper bound $\bigsqcup \rho_k(x)$ of elements of R^2_{fin}. To establish the order-isomorphism $\beta_2 : R^2 \cong D^2$, it therefore suffices to check:

Proposition 8.2 *R^2 (with the trace ordering) is a CPO.*

PROOF It suffices to check that any chain $x_1 \sqsubseteq x_2 \sqsubseteq \ldots$ where $x_k \in \mathrm{Im}\ \rho_k$ has a least upper bound in R^2. This an easy application of König's Lemma, analogous to that in [17, Theorem 5] or our Proposition 5.13. □

In order to show that $\mathbf{Mod}(\mathcal{B})$ and $\mathbf{HC}$ contain the same morphisms at type $\overline{3}$, we characterize the hypercoherence structure of $[\![\,\overline{2}\,]\!]$ in terms of the realizability model. Recall that in $\mathbf{Mod}(\mathcal{B})$ we have a surjective morphism $b : [\![\,\overline{1}\,]\!] \to [\![\,\overline{2}\,]\!]$ mapping f to $f \mid -$, realized by $\mathbf{irr}_{|}$. The crucial step will be to show that all chains and coherent sets in $[\![\,\overline{2}\,]\!]$ arise, via b, from chains and coherent sets in $[\![\,\overline{1}\,]\!]$. This will allow us to reduce questions about type $\overline{2} \to \overline{0}$ to questions about $\overline{1} \to \overline{0}$. (The basic idea here is inspired by [13].)

Lemma 8.3 *(i) Under the bijections β_1 and β_2, the morphisms $R^1 \to R^2$ in* $\mathbf{Mod}(\mathcal{B})$ *correspond exactly to the strongly stable maps $D^1 \to D^2$ in* $\mathbf{HC}$. *In particular, the function b corresponds to a strongly stable map.*

(ii) For every increasing chain $x_0 \sqsubseteq x_1 \sqsubseteq \ldots$ in R^2, there is an increasing chain $y_0 \sqsubseteq y_1 \sqsubseteq \ldots$ in R^1 such that $b(y_i) = x_i$ for all i.

(iii) For every coherent set $U \subseteq R^2$ (that is, for every set corresponding to a coherent subset of D^2), there is a coherent set $V \subseteq R^1$ such that $b(V) = (U)$.

PROOF (i) It is easy to give a definition of a retraction $([\![\,\overline{1}\,]\!] \Rightarrow [\![\,\overline{2}\,]\!]) \lhd [\![\,\overline{2}\,]\!]$ that works uniformly for both $\mathbf{Mod}(\mathcal{B})$ and $\mathbf{HC}$. The corresponding idempotents are then identified by β_2, and the result follows easily.

(ii) By Proposition 8.2, take $x = \bigsqcup x_i$ in R^2. Let r be any irredundant realizer for x. By Lemma 7.3, we may take irredundant realizers r_i for the x_i such that $r_i \sqsubseteq r$ and (by inspection of the proof of Lemma 7.3) $r_i \sqsubseteq r_{i+1}$. Thus the r_i form an increasing chain in R^1, and $x_i = b(r_i)$ for each i.

(iii) An important lemma (see e.g. [13, Lemma 3.2]) says that in any hypercoherence X, every coherent set is the image of a coherent set in N^ω via a strongly stable map $N^\omega \to X$. So take $W \subseteq R^1$ coherent and $g : R^1 \to R^2$ strongly stable such that $g(W)$ corresponds to U. By (i), g may be regarded as a morphism $[\![\,\overline{1}\,]\!] \to [\![\,\overline{2}\,]\!]$ in $\mathbf{Mod}(\mathcal{B})$. Suppose $r \in \mathcal{B}$ tracks g; then g factors through b via a morphism $\tilde{g} : [\![\,\overline{1}\,]\!] \to [\![\,\overline{1}\,]\!]$ tracked by r. But now $\tilde{g} : R^1 \to R^1$ is a strongly stable map, so $V = \tilde{g}(W)$ is coherent in R^1, and $b(V) = U$. □

Proposition 8.4 *Under the bijection β_2, the morphisms $R^2 \to \mathrm{N}_\perp$ in $\mathbf{Mod}(\mathcal{B})$ correspond exactly to the strongly stable maps $D^2 \to \mathrm{N}_\perp$ in* **HC**.

Proof Given $\Phi : [\![\,\overline{2}\,]\!] \to N_\perp$ in $\mathbf{Mod}(\mathcal{B})$, we have $\Phi \circ b : [\![\,\overline{1}\,]\!] \to N_\perp$ and so $\Phi \circ b \in R^2$. Hence $\Phi \circ b$ is strongly stable. We wish to check that Φ is strongly stable, that is,

- Φ respects chains and their lubs;
- Φ respects coherent sets and their meets.

Suppose $x_0 \sqsubseteq x_1 \sqsubseteq \ldots$ is a chain in R^2 with lub x. By Lemma 8.3(ii), take $y_0 \sqsubseteq y_1 \subseteq \ldots$ in R^1 such that $b(y_i) = x_i$, and let $y = \bigsqcup y_i$, so that $b(y) = x$. Since $\Phi \circ b$ is strongly stable it respects the chain (y_i) and its limit; hence Φ respects the chain (x_i) and its limit.

Likewise, given $U \subseteq R^2$ coherent, by Lemma 8.3(ii) take $V \subseteq R^1$ coherent such that $b(V) = U$. Since b is strongly stable, we have $b(\bigwedge V) = \bigwedge U$. But now $\Phi \circ b$ respects the coherent set V and its meet; hence Φ respects the coherent set U and its meet. □

We thus obtain a bijection $\beta_3 : R^3 \cong D^3$ that respects application. We can now easily extend the isomorphism to all pure types:

Theorem 8.5 *For every n there is a canonical bijection $\beta_n : R^n \cong D^n$, and these bijections respect application.*

Proof We already know the result for $n = 0, 1, 2$. We will prove by induction (starting from $n = 2$) that there is a bijection $\beta_{n+1} : R^{n+1} \cong D^{n+1}$ that respects application, and moreover there are retractions $(i_n, j_n) : [\![\,\overline{n+1}\,]\!] \lhd [\![\,\overline{n}\,]\!]$ in both $\mathbf{Mod}(\mathcal{B})$ and **HC** which are identified under β_{n+1} and β_n.

For $n = 2$, we already have a bijection β_3 respecting application. Since we may define a retraction $([\![\,\overline{2}\,]\!] \Rightarrow [\![\,\overline{3}\,]\!]) \lhd [\![\,\overline{3}\,]\!]$ uniformly in both $\mathbf{Mod}(\mathcal{B})$ and **HC**, we have a bijection between morphisms $R^3 \to R^2$ in $\mathbf{Mod}(\mathcal{B})$ and strongly stable maps $D^3 \to D^2$ in **HC**. In particular, H is present as a strongly stable map in **HC**. Moreover, since the map $b : [\![\,\overline{1}\,]\!] \to [\![\,\overline{2}\,]\!]$ is present in both models, the map $I = - \circ b$ is present in both models. We therefore have retractions $(I, H) : [\![\, \overline{3} \,]\!] \lhd [\![\, \overline{2} \,]\!]$ in both $\mathbf{Mod}(\mathcal{B})$ and $\mathbf{HC}$ as required.

For the induction step, given i_n and j_n in both models as above, let $i_{n+1} = \lambda f : \overline{n}.\ f \circ j_n$ and $j_{n+1} = \lambda g : \overline{n+1}.\ g \circ i_n$ Clearly (i_{n+1}, j_{n+1}) is a retraction $[\![\, \overline{n+2} \,]\!] \lhd [\![\, \overline{n+1} \,]\!]$ in both models. The corresponding idempotent on $[\![\, n+1 \,]\!]$ is $\lambda f : \overline{n}. f \circ j_n \circ i_n$, and this has the same meaning in both models modulo the bijection β_{n+1}, since by hypothesis the latter respects application. By restricting this bijection to the image of the idempotent, we obtain a bijection $\beta_{n+2} : R^{n+2} \cong D^{n+2}$. Clearly this respects application, and the retractions (i_{n+1}, j_{n+1}) in the two models are identified under β_{n+2} and β_{n+1}. $\Box$

It is now routine to extend the isomorphism to arbitrary types σ by transporting the bijections β_n along appropriate retractions $[\![\, \sigma \,]\!] \lhd [\![\, \overline{n} \,]\!]$, defined (uniformly for both models) as in the proof of Theorem 7.11. Similarly, it is straightforward to obtain a isomorphism for the corresponding *call-by-name* type structures, since any call-by-name type may be obtained as a definable retract of a call-by-value type (cf. [31, Chapter 6]). We omit the rather uninteresting details.

8.2 The category of retracts of type $\overline{2}$

We now draw attention to another consequence of the universality theorem: the object $[\![\, \overline{2} \,]\!]$ can be regarded as a combinatory algebra, and its category of retracts is an attractive category of domains with much interesting structure. There is a close analogy with the situation described in [50] for Scott's $\mathcal{P}\omega$, whose category of retracts is the very beautiful category of countably-based continuous lattices. Again we give just a sketch of the main ideas, omitting tedious details. The results of this section hold for both $\mathcal{B}_{full}$ and $\mathcal{B}_{eff}$.

First recall that we already have operations on objects of $\mathbf{Mod}(\mathcal{B})$ corresponding to products, sums, exponentials and lifting. We may also define a *separated sum* operation $\oplus$ on objects by $X \oplus Y = (X + Y)_\perp$.

Let us now write B_2 for the object $[\![\, \overline{2} \,]\!]$ in $\mathbf{Mod}(\mathcal{B})$. We will be interested in objects which are retracts of B_2 in $\mathbf{Mod}(\mathcal{B})$; we write $\mathcal{K}$ for the full subcategory consisting of such objects. The crucial observations are the following:

Proposition 8.6 *(i) The objects $B_2 \times B_2, B_2^{B_2}, B_{2\perp}, B_2 \oplus B_2$ are retracts of B_2.*

(ii) The category $\mathcal{K}$ has products, exponentials, lifting and separated sums, all inherited from $\mathbf{Mod}(\mathcal{B})$.

(iii) Each object X of $\mathcal{K}$ comes with a lift algebra structure $X_\perp \to X$, and has a canonical fixed point operator $X^X \to X$.

Proof (i) It is routine to exhibit all these objects as retracts of $[\![\, \overline{3} \,]\!]$. So by Theorem 7.9 they are retracts of B_2.

(ii) Given $X \lhd B_2$ and $Y \lhd B_2$, we have induced retractions $X \times Y \lhd B_2 \times B_2$, $Y^X \lhd B_2^{B_2}$, $X_\perp \lhd B_{2\perp}$ and $X \oplus Y \lhd B_2 \oplus B_2$. By (i) all these objects are retracts of B_2, and the result follows since $\mathcal{K}$ is full in $\mathbf{Mod}(\mathcal{B})$.

(iii) The evident lift algebra structure on B_2 induces one on X via the retraction $X_\perp \lhd B_{2\perp}$. For fixed points, it is easy to see that there is a fixed point operator $Y : B_2^{B_2} \to B_2$ tracked by $\mathbf{y}$, since pre-irredundant realizers are closed under lubs of chains. (Otherwise, the existence of Y may be deduced from the connection with the hypercoherence model.) This transfers to a fixed point operator $X^X \to X$ via the retractions $X^X \lhd B_2^{B_2}$ and $X \lhd B_2$. $\Box$

The retraction $B_2^{B_2} \lhd B_2$ in particular means that the elements of B_2 form a combinatory algebra—indeed a λ-*model*, since $\mathbf{Mod}(\mathcal{B})$ is well-pointed (see [3]). We will write $\mathcal{B}_2$ for this combinatory algebra. Note that the monoid $\mathcal{M}_2$ of realizable endofunctions of $\mathcal{B}_2$ is precisely the monoid of endomorphisms of B_2 in $\mathbf{Mod}(\mathcal{B})$, and since all idempotents have splittings in $\mathbf{Mod}(\mathcal{B})$, the Karoubi envelope $K(\mathcal{M}_2)$ is equivalent to $\mathcal{K}$. It follows that there is a full embedding $E_2 : \mathcal{K} \to \mathbf{Mod}(\mathcal{B}_2)$, analogous to the embedding $E : \mathbf{SSeqFun} \to \mathbf{Mod}(\mathcal{B})$ of Section 4.3. The embedding E_2 preserves products, exponentials, lifting and separated sums, and has one significant advantage over the inclusion $\mathcal{K} \hookrightarrow \mathbf{Mod}(\mathcal{B})$: every object in the image of E_2 is *projective* (each element has just one realizer).

To summarize, $\mathcal{B}$ gave rise to the category of retracts $K(\mathcal{M})$, which included $\mathbf{SSeqFun}$, but $\mathcal{B}_2$ gives rise to the category of retracts $\mathcal{K}$. Since $\mathcal{K}$ is much larger than $K(\mathcal{M})$ and has better closure properties (it is cartesian closed), we conclude that $\mathcal{B}_2$ is in some sense a better model than $\mathcal{B}$. However, the construction of $\mathcal{B}_2$ is undoubtedly more complicated, and indeed proceeds via the construction of $\mathcal{B}$.

The situation described above for $\mathcal{B}_2$ is very similar to that for $\mathcal{P}\omega$: the category of retracts of $\mathcal{P}\omega$ is a good category of domains with products, exponentials, lifting and separated sums, and it embeds well in $\mathbf{Mod}(\mathcal{P}\omega)$ (see [31, Section 7.3]). Furthermore, it is shown in [50] that one can perform type operations on retracts of $\mathcal{P}\omega$ within $\mathcal{P}\omega$ itself, and hence construct recursive domains simply by taking fixed points in $\mathcal{P}\omega$. We can now see that one can do exactly the same thing for $\mathcal{B}_2$. We give only a very brief outline, since the details are formally identical to those in [50].

An object X in $\mathcal{K}$ can be *represented* (up to isomorphism) by any element $x \in \mathcal{B}_2$ that realizes an idempotent $\mathcal{B}_2 \to \mathcal{B}_2$ corresponding to some retraction $X \lhd B_2$. Note that x satisfies $xa = x(xa)$ for all $a \in \mathcal{B}_2$; we will call any such element $x \in \mathcal{B}_2$ an *idempotent*. Clearly every idempotent x represents some object X. If F is a k-ary operation on objects of $\mathcal{K}$, we say that F is *realizable* if there is an element $f \in \mathcal{B}_2$ such that whenever $x_1, \ldots, x_k$ are idempotents representing $X_1, \ldots, X_k$ respectively, the element $f x_1 \ldots x_k$ is an idempotent representing $F(X_1, \ldots, X_k)$. The following proposition lets us construct many examples of realizable operations:

Proposition 8.7 *(i) The product, exponential and separated sum operations are realizable binary operations on objects of $\mathcal{K}$, and the lift operation is a realizable unary operation.*

(ii) The identity operation on objects is realizable. Constant operations are realizable. Realizable operations are closed under composition.

Proof We consider exponentials as an example. Suppose $r \in \mathcal{B}_2$ represents the retraction $B_2^{B_2}$, and x, y represent retractions $X \lhd B_2$, $Y \lhd B_2$ respectively. Then $\lambda^* a. \lambda^* z.\, y((ra)(xz))$ (interpreted in $\mathcal{B}_2$) represents the induced retraction

$Y^X \lhd B_2$. Thus, $\lambda^* xyaz.\, y((ra)(xz))$ realizes the exponential operation. The other cases are similar.

(ii) is trivial. $\Box$

It is now easy to obtain recursive types as fixed points of realizable operations:

Theorem 8.8 *For any realizable unary operation F, there is an object X of $\mathcal{K}$ such that $X \cong F(X)$. In particular, $\mathcal{K}$ contains solutions to all recursive type equations $X \cong F(X)$ where F is a unary operation constructed using $\times, \Rightarrow, -_\perp, \oplus$ and fixed objects of $\mathcal{K}$.*

PROOF Suppose f realizes F. Define a sequence $r_0, r_1, \ldots$ in $\mathcal{B}_2$ by $r_0 = \perp$, $r_{n+1} = fr_n$. Clearly r_0 is an idempotent, and hence every r_n is an idempotent. Moreover, the action of f gives a morphism $B_2 \to B_2$ and so is monotone with respect to the trace ordering; thus $r_0 \sqsubseteq r_1 \sqsubseteq \ldots$. By Proposition 8.2, take $r = \bigsqcup r_n$ in B_2. Since each r_n is an idempotent, it is easy to see that r is an idempotent, and that $fr = r$. Take X an object represented by r; then $X \cong F(X)$. The rest of the theorem follows immediately by Proposition 8.7. $\Box$

Remark 8.9 The category **HC** also contains an object $[\![\overline{2}]\!]$ and has many of the closure properties mentioned above for $\mathcal{K}$. It would therefore appear that **HC** and $\mathcal{K}$ have a large full subcategory in common, but at present we do not know the precise extent of the overlap. Since not all idempotents on $[\![\overline{2}]\!]$ split in **HC**, there are objects of $\mathcal{K}$ with no counterpart in **HC**; we do not know whether the converse is true.

9 PCF and universal functionals

We now consider interpretations of call-by-value PCF in $\mathbf{Mod}(\mathcal{B})$ and $\mathbf{Mod}(\mathcal{B}_{\mathit{eff}})$. We will show that the functional H of Section 7 is *universal*, in the sense that every effective SR functional is definable relative to H in PCF. We then consider the extended language PCF+H, which gives us yet another handle on the SR functionals. Next we briefly touch on a notion of "degree" for SR functionals under the relation of relative PCF-definability. Finally, we prove a technical result: there is no universal SR functional with an essentially lower type than H.

9.1 Call-by-value PCF

We review the syntax and semantics of call-by-value PCF (henceforth called PCF). The *types* of PCF are just the simple types as given by Definition 3.5(i). For each type σ we have an infinite supply of variables $x_0^\sigma, x_1^\sigma, \ldots$. Exactly as in call-by-name PCF, the well-typed *terms* are built up from variables and the constants $0, 1, 2, \ldots : \overline{0}$, $\mathsf{succ}, \mathsf{pred} : \overline{1}$ and $\mathsf{cond} : \overline{0} \to \overline{0} \to \overline{0} \to \overline{0}$ by means of the following rules:

$$\frac{M : \tau}{(\lambda x_i^\sigma.M) : \sigma \to \tau} \qquad \frac{M : \sigma \to \tau \quad N : \sigma}{(MN) : \tau} \qquad \frac{M : \sigma}{(\mu x_i^\sigma.M) : \sigma}$$

A closed term is a *value* if it is either a numeral $0, 1, 2, \ldots$ or a λ-abstraction; we use V to range over values.

We give an operational semantics by defining a (big-step) *evaluation relation* $\Downarrow$ from closed terms to values, such that for each closed term M we have $M \Downarrow V$ for at most one V. The relation $\Downarrow$ is defined inductively by the following derivation rules. Here $\smile$ denotes truncated subtraction: $m \smile n = \max(m - n, 0)$.

$$V \Downarrow V \quad (V \text{ a value}) \qquad\qquad \frac{M[\mu x.M/x] \Downarrow V}{\mu x.M \Downarrow V}$$

$$\frac{M \Downarrow n}{\mathsf{succ}\, M \Downarrow n+1} \qquad\qquad \frac{M \Downarrow n}{\mathsf{pred}\, M \Downarrow n \smile 1}$$

$$\frac{M \Downarrow 0 \quad N \Downarrow V}{\mathsf{cond}\, MNP \Downarrow V} \qquad\qquad \frac{M \Downarrow n+1 \quad P \Downarrow V}{\mathsf{cond}\, MNP \Downarrow V}$$

$$\frac{M \Downarrow_N \lambda x.M' \qquad M'[N/x] \Downarrow_N V}{MN \Downarrow_N V}$$

We write $M \Uparrow$ to mean that there is no V such that $M \Downarrow V$.

Next we give the denotational semantics of this language in $\mathbf{Mod}(\mathcal{B})$ (where $\mathcal{B}$ is either $\mathcal{B}_{\mathit{full}}$ or $\mathcal{B}_{\mathit{eff}}$). The interpretation $[\![-]\!]$ of types is given essentially as in Section 7; however, here it is convenient to interpret function types using the standard exponentials and lifting rather than the special objects used there. We will also write $[\sigma]$ for $[\![\sigma]\!]_\perp$, and η for the inclusion $[\![\sigma]\!] \to [\sigma]$.

We now interpret a term $M : \tau$ whose free variables are included in $\Gamma = \langle x_1^{\sigma_1}, \ldots, x_n^{\sigma_n} \rangle$, by a morphism $[M]^\Gamma : [\sigma_1] \times \cdots \times [\sigma_n] \Rightarrow [\tau]$. The definition of $[M]^\Gamma$ is by induction on the structure of M. The clauses for variables, numerals, succ, pred and cond are evident. For the remaining clauses, some further notation is helpful. Let us write $\mathit{fix}_\sigma : [\sigma]^{[\sigma]} \to [\sigma]$ for the fixed point morphism given by Proposition 8.6. We also write α_σ for the evident lift algebra structure $[\![\sigma]\!]_\perp \to [\![\sigma]\!]$, and $\mathit{strict} : B_\perp^A \to B_\perp^{A_\perp}$ for the morphism that extends a map $A \to B_\perp$ to a strict map $A_\perp \to B_\perp$. We then have:

$$\begin{array}{rcl}
[\lambda x^\sigma.M]^\Gamma & = & \eta \circ [\sigma]^\eta \circ \mathit{curry}\,([M]^{\Gamma, x^\sigma}) \\
[MN]^\Gamma & = & \mathit{eval} \circ \langle \mathit{strict} \circ \alpha_{\sigma\to\tau} \circ [M]^\Gamma,\ [N]^\Gamma \rangle \\
[\mu x^\sigma.M]^\Gamma & = & \mathit{fix}_\sigma \circ \mathit{curry}\,([M]^{\Gamma, x^\sigma})
\end{array}$$

A closed term $M : \tau$ is interpreted as a morphism $1 \to [\tau]$, that is, an element of $[\tau]$. The following theorem establishes the agreement between the operational and denotational semantics.

Theorem 9.1 (Adequacy) *For any closed term $M : \overline{0}$, we have $[M] = n$ if $M \Downarrow n$, and $[M] = \perp$ if $M \Uparrow$.*

PROOF The usual technique for proving adequacy works (see [45]). In fact the result follows immediately from results in [31, Chapter 6], as we shall see in Section 10. □

We now investigate the expressive power of PCF for defining SR functionals. As before, we will write R for either R_{full} or R_{eff}. We say that an element $x \in R^{\sigma}$ is *PCF-definable* if there is a closed PCF term $M : \sigma$ such that $[M] = \eta(x)$. A function $f : R^{\sigma} \to R^{\tau}$ is PCF-definable if $\eta \circ f$ is PCF-definable as an element of $R^{\sigma \to \tau}$.

Proposition 9.2 *(i) The map $b : R^{\overline{1}} \to R^{\overline{2}}$ of Section 7.2 is PCF-definable.*
(ii) The map $I : R^{\overline{3}} \to R^{\overline{2}}$ is PCF-definable.

PROOF (i) Let $\mathsf{isans} : \overline{1}$, $\mathsf{strip} : \overline{1}$, $\mathsf{empty} : \overline{0}$ and $\mathsf{add} : \overline{0} \to \overline{0} \to \overline{0}$ be PCF terms such that for our chosen effective codings $[?, !]$ and $\langle \cdots \rangle$ we have

$$\begin{array}{rcl@{\qquad}rcl} [\![\,\mathsf{empty}\,]\!] & = & \langle \epsilon \rangle & [\![\,\mathsf{add}\,]\!](n)(\langle \alpha \rangle) & = & \langle \alpha, n \rangle \\ [\![\,\mathsf{isans}\,]\!](!n) & = & 0 & [\![\,\mathsf{isans}\,]\!](?n) & = & 1 \\ [\![\,\mathsf{strip}\,]\!](!n) & = & n & [\![\,\mathsf{strip}\,]\!](?n) & = & n. \end{array}$$

Now let $\rho \equiv \overline{1} \to \overline{1} \to \overline{0} \to \overline{0}$, and let play be the PCF term

$$\mu P^{\rho}.\, \lambda f^{\overline{1}}.\lambda g^{\overline{1}}.\lambda a^{\overline{0}}.\, \mathsf{cond}\ (\mathsf{isans}\,(f\ a))\ \ (\mathsf{strip}\,(f\ a))\ \ (P\ f\ g\,(\mathsf{add}\ a\ (g\,(\mathsf{strip}\,(f\ a))))).$$

Finally, let $\mathsf{b} = \lambda f^{\overline{1}}.\, \lambda g^{\overline{1}}.\, \mathsf{play}\ f\ g\ \mathsf{empty}$. We omit the tedious verification that b defines b.

(ii) Now let $\mathsf{I} = \lambda \Phi^{\overline{3}}.\, \lambda g^{\overline{1}}.\, \Phi(\mathsf{bar}\ g)$. Then I defines I. □

As a consequence of this we obtain an important property of the function H.

Theorem 9.3 (Universality of H) *Every element $f \in [\sigma]_{eff}$ is PCF-definable relative to H: that is, there is a closed PCF term $M : (\overline{2} \to \overline{3}) \to \sigma$ such that $f = [M](\eta(\eta \circ H))$.*

PROOF Clearly the term $\mu x^{\sigma}.x$ denotes $\perp \in [\sigma]_{eff}$, so it suffices to show that every $f \in [\![\,\sigma\,]\!]_{eff}$ is definable from H. Firstly, any element $r \in [\![\,\overline{1}\,]\!]_{eff}$ is definable in pure PCF as it is simply a partial recursive function. Secondly, given any element $F \in [\![\,\overline{2}\,]\!]_{eff}$, take $r \in B_{eff}$ a realizer for F and $M : \overline{1}$ denoting r; then $[\mathsf{b}\,M] = \eta(F)$. Thus every element of $[\![\,\overline{2}\,]\!]_{eff}$ is PCF-definable. Next, using Proposition 9.2(ii), both halves of the retraction $(I, H) : [\![\,\overline{3}\,]\!] \lhd [\![\,\overline{2}\,]\!]$ are PCF-definable relative to H, and it follows easily that for any σ, both halves of the induced retraction $(i, j) : [\![\,\sigma\,]\!] \lhd [\![\,\overline{2}\,]\!]$ are PCF-definable relative to H. Now given any $f \in [\![\,\sigma\,]\!]$, let $P : \overline{2}$ be a PCF term denoting $i(f)$, and let $N : \overline{2} \to (\overline{2} \to \sigma)$ be a PCF term such that $[N](\eta(\eta \circ H)) = j$. Then clearly $\lambda h.\, (Nh)P$ defines f relative to H. □

Remarks 9.4 (i) Clearly one can obtain an analogous result for the call-by-name setting by using a call-by-name analogue of H. As an easy corollary, one can derive Ehrhard's result that any *finite* SR functional is definable relative to some SR functional of type level 2. Specifically, let $H_0 \sqsubseteq H_1 \sqsubseteq \ldots$ be the sequence of finite approximations to H induced by the standard sequence of finite retractions $\rho_0 \sqsubseteq \rho_1 \sqsubseteq \ldots$ on type $\overline{2}$ (see Section 8.1). Any finite SR functional of any type is PCF-definable from H and hence from some H_i; but from the definition of ρ_i

it is clear that H_i is itself definable from an element of type level 2. (See also Remark 9.23(ii) below.)

(ii) It follows from the proof of the above theorem that, in the presence of H, one only needs the PCF fixed point operator μx^σ up to and including type $\overline{2}$ in order to define all effective SR functionals. (It is easy to see that the fixpoint operator for the type ρ in the proof of Proposition 9.2 may be defined from that for $\overline{2}$.) An analogous result for PCF with parallel features was pointed out in [45].

9.2 The language PCF+H

Let us now extend the language PCF by adding a new constant $\mathsf{H} : \overline{2} \to \overline{3}$. We can give a denotational semantics for the language PCF+H in $\mathbf{Mod}(\mathcal{B})$ simply by adding the clause $[\mathsf{H}]^\Gamma = \eta(\eta \circ H)$ to the definition of $[-]^\Gamma$. We can then rephrase Theorem 9.3 as follows: every effective SR functional is definable in PCF+H.

Does the language PCF+H have an operational semantics? At present we do not have a palatable way to give a semantics using structural operational rules. (One particular problem is to see how to deal correctly with terms involving recursive calls to H.) However, one can give an "operational semantics" in a rather abstract sense, by specifying a way in which terms of PCF+H may be "compiled" to elements of $\mathcal{B}$, which we can regard as a kind of abstract machine for which we already have an operational semantics. In particular, our denotational semantics of PCF+H suggests a way of mapping each term M to a particular choice of realizer for $[M]$. We can think of this mapping as a translation from PCF+H into the untyped setting of $\mathcal{B}$.

Let us indicate one way to do this explicitly. Let $\mathsf{K}, \mathsf{S}, \mathsf{Y}$ be new constant-symbols. Given any closed term M of PCF+H, we first translate M into a term $\overline{M}$ into a term of *combinatory* PCF+H, in which the terms are built up from the constants $0, 1, 2, \ldots, \mathsf{succ}, \mathsf{pred}, \mathsf{cond}, \mathsf{K}, \mathsf{S}, \mathsf{Y}, \mathsf{H}$ using application (subject to typing constraints). Specifically, $\overline{M}$ be obtained from M as follows:

- First replace each subterm $\mu x^\sigma.N$ by $\mathsf{Y}(\lambda x^\sigma.N)$.
- Then replace each subterm $\lambda x^\sigma.N$ by its standard Curry translation into combinatory logic, working from innermost subterms outwards.

It is easy to give a denotational semantics $[-]$ for (type-decorated) combinatory terms so that $[\overline{M}] = [M]$.

We may now change our perspective and regard $\overline{M}$ as an expression in an *untyped* language of combinators. Indeed, each of the above combinatory constants C has a denotation $\ll C \gg \in \mathcal{B}_{\mathit{eff}}$. For $C = 0, 1, 2, \ldots, \mathsf{succ}, \mathsf{pred}, \mathsf{cond}, \mathsf{H}$ we simply take $\ll C \gg$ to be some realizer for the corresponding element in $\mathbf{Mod}(\mathcal{B}_{\mathit{eff}})$; we also take $\ll \mathsf{K} \gg = \mathbf{k}$, $\ll PS \gg = \mathbf{s}$, $\ll \mathsf{Y} \gg = \mathbf{y}$. We can extend $\ll - \gg$ to arbitrary closed combinatory terms simply by taking $\ll MN \gg = \ll M \gg \bullet \ll N \gg$. Since we have an operational definition of the operation $\bullet$ (e.g. a simple adaptation of Proposition 9.2(i) gives a definition of $\bullet$ in PCF), we have a kind of "operational semantics" for this untyped combinatory language, and hence for PCF+H.

If M is a closed term of PCF+H, we define $\ll M \gg = \ll \overline{M} \gg$. We now define a big-step evaluation relation $\Downarrow$ for closed PCF+H terms of type $\overline{0}$ by

$$M \Downarrow n \;\text{ iff }\; \ll M \gg (0) \;=\; n.$$

We may also define a unary termination predicate $\Downarrow$ for terms of any type σ by

$$M \Downarrow \;\text{ iff }\; \ll M \gg (0) \downarrow .$$

The following facts are easily verified:

Theorem 9.5 (Adequacy for PCF+H) *In both the full and effective models,*
(i) For any closed PCF+H term $M : \overline{0}$ we have $M \Downarrow n$ iff $[M] = n$.
(ii) For any closed term $M : \sigma$ we have $M \Downarrow$ iff $[M] \in [\![\sigma]\!]$ (that is, $[M] \neq \bot$).

PROOF It is clear that for any combinatory term M, $\ll M \gg$ is a realizer for $[M]$ in both models. Hence the same holds for any PCF+H term M. But $r \in \mathcal{B}$ realizes $n \in [\overline{0}]$ iff $r(0) = n$; hence $[M] = n$ iff $\ll M \gg (0) = n$. For (ii), just recall that $[M] = [\![\sigma]\!]_\bot$, and from the standard description of lifting given in Section 10 we have that for any $x \in X_\bot$ and $r \in \|x\|_{X_\bot}$ we have $x \in X$ iff $r(0) \downarrow$. □

Remark 9.6 A rather more concrete way to specify an operational semantics for PCF+H might be to give a translation to a language such as PCF+catch [12] or μPCF [42], for which a structural operational semantics had already been given. Essentially this would amount to giving an implementation of H in the target language.

From the evaluation relation for PCF+H we may define a notion of observational equivalence for closed terms of the same type as usual: $M \approx N$ if for all contexts $C[-] : \overline{0}$ we have $C[M] \Downarrow n \Leftrightarrow C[N] \Downarrow n$. The following results are now straightforward:

Theorem 9.7 (Full abstraction for PCF+H) *For closed terms $M, N : \sigma$ the following are equivalent:*
(i) $M \approx N$;
(ii) $[M]_{\mathit{eff}} = [N]_{\mathit{eff}}$;
(iii) $[M]_{\mathit{full}} = [N]_{\mathit{full}}$.

PROOF (i) $\Rightarrow$ (ii): Suppose $\sigma = \sigma_1 \to \cdots \to \sigma_r \to \overline{0}$ $(r \geq 0)$ and $M \approx N$. Given any $x_1 \in [\![\sigma_1]\!]_{\mathit{eff}}, \ldots, x_r \in [\![\sigma_r]\!]_{\mathit{eff}}$, by universality we may take terms $P_1, \ldots, P_r$ such that $[P_i]_{\mathit{eff}} = x_i$; then $MP_1 \ldots P_r = NP_1 \ldots P_r$ and so $[M]_{\mathit{eff}} x_1 \ldots x_r = [N]_{\mathit{eff}} x_1 \ldots x_r$. Since $\mathbf{Mod}(\mathcal{B}_{\mathit{eff}})$ is well-pointed, this implies $[M]_{\mathit{eff}} = [N]_{\mathit{eff}}$.

(ii) $\Rightarrow$ (iii): For $\sigma = \overline{2}$ the result is easy: if $[M]_{\mathit{eff}} = [N]_{\mathit{eff}}$ then $MP \approx NP$ whenever P denotes a finite function; hence $[M]_{\mathit{full}}(f) = [N]_{\mathit{full}}(f)$ for all finite f, which clearly implies $[M]_{\mathit{full}} = [N]_{\mathit{full}}$.

Now consider an arbitrary type σ. Let $I_\sigma : \sigma \to \overline{2}$ and $J_\sigma : \overline{2} \to \sigma$ be terms of PCF+H that denote the two halves of the canonical retraction $[\![\sigma]\!] \lhd [\![2]\!]$

uniformly in both the full and effective models. Suppose $[M]_{\mathit{eff}} = [N]_{\mathit{eff}}$; then $[I_\sigma M]_{\mathit{eff}} = [I_\sigma N]_{\mathit{eff}}$, so $[I_\sigma M]_{\mathit{full}} = [I_\sigma N]_{\mathit{full}}$ by the above, and so $[J_\sigma(I_\sigma M)]_{\mathit{full}} = [J_\sigma(I_\sigma N)]_{\mathit{full}}$. There are now two cases. If $[M]_{\mathit{eff}} = \bot$ then $\neg M \Downarrow$ and so $[M]_{\mathit{full}} = \bot$, and similarly for N; hence $[M]_{\mathit{full}} = [N]_{\mathit{full}}$. Otherwise we have $[M]_{\mathit{full}} \in [\![\,\sigma\,]\!]$ and so $[J_\sigma(I_\sigma M)]_{\mathit{full}} = [M]_{\mathit{full}}$, and similarly for N, so again $[M]_{\mathit{full}} = [N]_{\mathit{full}}$.

(iii) $\Rightarrow$ (i) is easy by the adequacy of $[-]_{\mathit{full}}$. $\square$

One immediate consequence of full abstraction is the equational context lemma for PCF+H:

Proposition 9.8 (Context Lemma) *For closed PCF+*H *terms* $M, N : \sigma = \sigma_1 \to \cdots \to \sigma_r \to \overline{0}$*, we have* $M \approx N$ *iff for all closed* $P_1 : \sigma_1, \ldots, P_r : \sigma_r$*,* $MP_1 \ldots P_r = NP_1 \ldots P_r$*.* $\square$

Note however that the *inequational* version of the Context Lemma fails, because the observational ordering coincides with the stable order rather than the pointwise one.

Another corollary is that the type structure of effective SR functionals can be characterized as the term model for PCF+H. Specifically, suppose we have defined the syntax and operational semantics of PCF+H, and let $\approx$ be the notion of observational equivalence defined as above. For each type σ, let T^σ be the set of *values* (i.e. numerals or λ-abstractions) of PCF+H modulo $\approx$, and let $\cdot : T^{\sigma\to\tau} \times T^\sigma \rightharpoonup T^\tau$ be the operation induced by juxtaposition. Then it is immediate from Theorems 9.3 and 9.7 that $T \cong R_{V,\mathit{eff}}$. This characterization gives a genuinely new insight into the effective SR functionals; in particular we may now deduce the following pleasing recursion-theoretic property, which is not obvious from any of the previous characterizations:

Theorem 9.9 *The type structure* $R_{V,\mathit{eff}}$ *is* effective *in the following strong sense: there are* total *enumerations* ϵ^σ *of the set* $R^\sigma_{V,\mathit{eff}}$ *(that is, surjective total functions* $\mathrm{N} \to R^\sigma_{V,\mathit{eff}}$*) and partial recursive functions* $\varphi_{\sigma\tau} : \mathrm{N} \times \mathrm{N} \rightharpoonup \mathrm{N}$ *such that for all* σ, τ, m, n *we have*

$$\begin{array}{lcl} \varphi_{\sigma\tau}(m,n)\downarrow & \Longrightarrow & \epsilon^{\sigma\to\tau}(m)\cdot\epsilon^\sigma(n) \;=\; \epsilon^\tau(\varphi_{\sigma\tau}(m,n)), \\ \varphi_{\sigma\tau}(m,n)\uparrow & \Longrightarrow & \epsilon^{\sigma\to\tau}(m)\cdot\epsilon^\sigma(n)\uparrow . \end{array}$$

Proof For each type σ let ϵ^σ be induced by an effective surjective coding $\lceil - \rceil^\sigma$ of the (decidable) set of PCF+H values of type σ, via the bijection $T^\sigma \cong R^\sigma_{v,\mathit{eff}}$. We wish to construct $\varphi_{\sigma\tau}$ such that, for all values $M : \sigma \to \tau, N : \sigma$,

- if $MN \Downarrow$ then $\varphi_{\sigma\tau}(\lceil M\rceil^{\sigma\to\tau}, \lceil N\rceil^\sigma) = \lceil P\rceil^\tau$ for some value $P : \tau$ such that $MN \approx P$;
- if $\neg MN \Downarrow$ then $\varphi_{\sigma\tau}(\lceil M\rceil^{\sigma\to\tau}, \lceil N\rceil^\sigma)\uparrow$.

There are two cases. If $\tau = \overline{0}$ then let $\varphi_{\sigma\tau}$ be the function which given $\lceil M\rceil^{\sigma\to\tau}$ and $\lceil N\rceil^\sigma$ returns $\lceil n\rceil^{\overline{0}}$ if $\ll MN \gg (0) = n$, and diverges if $\ll MN \gg (0)\uparrow$. Clearly $\varphi_{\sigma\tau}$ is partial recursive and has the required properties. If $\tau = \tau_1 \to \tau_2$, let $\varphi_{\sigma\tau}$ be the function which given $\lceil M\rceil^{\sigma\to\tau}, \lceil N\rceil^\sigma$ returns $\lceil \lambda x^{\tau_1}.MNx\rceil^\tau$ if $\ll MN \gg (0)\downarrow$, and diverges otherwise. Again this clearly has the required properties. $\square$

We can also now read off the following fact, whose proof (essentially) we have been deferring since Section 3.2:

Theorem 9.10 *$R_{V,\mathit{eff}}$ is a* substructure *of R_V, i.e., there are inclusions $R^\sigma_{V,\mathit{eff}} \hookrightarrow R^\sigma_V$ that commute with the application operations.*

Proof Immediate from the fact that $R_{V,\mathit{eff}} \cong T$ and the values of PCF+H admit a fully abstract interpretation in R_V. $\Box$

Remarks 9.11 (i) The situation of the above theorem is typical for notions of higher-type *partial* functional: the computable objects obtained hereditarily as those that act on computable arguments can just as well be seen as acting on arbitrary continuous arguments. (For instance, the effective analogue of the Scott-continuous functionals yields a substructure of the full Scott-continuous type structure.) The situation is radically different for (hereditarily) *total* notions of computability: not all effective total type 2 operations on the total recursive functions can be extended to an effective total operation on arbitrary total functions (see e.g. [46]).

(ii) All the above results of course go through for the call-by-name SR functionals, using the call-by-name analogue of H.

(iii) Our general impression is that the *existence* of a universal SR functional is interesting, but the *particular* functional H is not. For both conceptual and practical reasons (cf. Remark 7.13), it would be very interesting if one could find a simpler functional with the same universality property. We show below that there can be no universal functional of an essentially simpler type than H, but it is open whether there is a universal functional of lower computational complexity than H.

9.3 Degrees of expressivity

Next we consider some effective SR functionals other than H. There is clearly a sense in which some SR functionals are more expressive than others, and indeed there is a structure of *degrees* of expressivity, analogous to the degrees of parallelism in the Scott model [49]. These ideas are made precise as follows. We will write PCF_N for the standard call-by-name version of PCF (as in [45]), and PCF_V for the call-by-value version described above. For simplicity, we will restrict our attention to degrees of *effective* SR functionals, and so for the rest of this section we will write R_N, R_V in place of $R_{N,\mathit{eff}}$, $R_{V,\mathit{eff}}$.

Definition 9.12 *(i) Let $|R_X| = \bigsqcup R^\sigma_X$, where X is either N or V. Given $f, g \in |R_X|$, we write $f \preceq g$ if there is an element $m \in R^{\tau\to\sigma}_X$, definable in* PCF_X*, such that $m \cdot g = f$.*
We write $f \sim g$ if both $f \preceq g$ and $g \preceq f$.

(ii) A degree *is an equivalence class for $\sim$; we write $d_X(f)$ for the degree containing f. The partial order on degrees induced by $\preceq$ is also written as $\preceq$.*

Our first observation is that degrees in R_N can be correlated with those in R_V, so that we really have just one universe of degrees. For each type σ we have types

$\hat{\sigma}, \tilde{\sigma}$ and standard retractions $(\delta^\sigma, \epsilon^\sigma) : [\![\sigma]\!]_N \lhd [\![\hat{\sigma}]\!]_V$ and $(\zeta^\sigma, \xi^\sigma) : [\![\sigma]\!]_V \lhd [\![\tilde{\sigma}]\!]_N$ in $\mathbf{Mod}(\mathcal{B}_{\mathit{eff}})$, as in [31, Chapter 6]. Roughly speaking, these retractions have the property that all possible composites of type $[\![\rho]\!]_N \to [\![\rho']\!]_N$ are definable in PCF_N, and similarly for V (see [31] for details).

Given $f \in R_N^\sigma$ and $g \in R_V^\tau$, we say $f \preceq g$ if $\delta^\sigma(f) \preceq g$, or equivalently if $f \preceq \zeta^\tau(g)$. Likewise, $g \preceq f$ if $\zeta^\tau(g) \preceq f$, or equivalently if $f \preceq \delta^\sigma(g)$. We say $f \sim g \sim f$ if both $f \preceq g$ and $g \preceq f$. With these definitions, $\preceq$ becomes a preorder on the whole of $|R| = |R_N| \sqcup |R_V|$, with $\sim$ the induced equivalence relation. For $f \in |R_X|$, we write $d(f)$ for the $\sim$-equivalence class of f in $|R|$. Clearly the poset of degrees in $|R|$ is isomorphic to the poset of degrees on $|R_X|$, where X is either N or V. Henceforth we will freely mix call-by-name and call-by-value functionals in our discussion, using whichever seems most convenient.

The smallest degree (with respect to $\preceq$) is the degree $d(0)$ of all PCF-definable functionals; the largest degree is $d(H)$. The structure of the poset of degrees is probably extremely complicated, and in this paper we will do no more than consider a few examples.

An example of a degree strictly between these is the degree of the functional $F \in R_N^{\overline{2}}$ defined in Remark 3.7(ii) (see also the Introduction). The fact that $d(F)$ is strictly above $d(0)$ is immediate from the pointwise non-monotonicity of F; the fact that $d(F)$ is strictly below $d(H)$ (i.e. that H is not definable from F) will be shown below.

Another SR functional that seems particularly interesting from the point of view of programming applications is the *modulus* functional $M : R_V^{\overline{2} \to \overline{1} \to \overline{0}}$ specified as follows:

- If $G \cdot f \in \mathrm{N}$ then $M \cdot G \cdot f = \lceil \text{graph } f_0 \rceil$, where f_0 is the unique smallest finite subfunction of f such that $G \cdot f_0 = n$, and $\lceil \text{graph } f_0 \rceil$ is the normalized graph of f_0 (e.g. a sorted non-repetitive list of ordered pairs), coded as a natural number.
- If $G \cdot f = \bot$ then $M \cdot G \cdot f = \bot$.

It is easy to check that M is indeed an effective call-by-value SR functional of the specified type. The following observation is due to Alex Simpson:

Proposition 9.13 *$d(M) = d(F)$.*

Proof Let $F' \in R_V^{\overline{3}}$ be the functional

$$\lambda G^{\overline{2}}.\, \mathsf{cond}\ (MG(\lambda x^{\overline{0}}.0) = \lceil \emptyset \rceil)\ 0\ 1$$

(we mix syntax and semantics in a semi-formal way). Intuitively F' is a call-by-value analogue of F, and indeed it is easy to see that $d(F') = d(F)$. Thus $d(F) \preceq d(M)$. For the converse, we may recover M from F' as follows. Let $\mathit{conv} : \overline{1} \to \overline{0} \to \overline{0}$ be the function $\lambda p^{\overline{1}}.\, \lambda x^{\overline{0}}.\, \mathsf{cond}\ (p\, 0)\, x\, x$. Let $\mathit{insert} : \overline{0} \to \overline{0} \to \overline{0} \to \overline{0}$ be a function such that

$$\mathit{insert}\ x\ y\ \lceil \gamma \rceil \;=\; \lceil \gamma \cup (x \mapsto y) \rceil$$

for all natural numbers x, y and graphs γ. Next, let $M' : \overline{2} \to \overline{1} \to \overline{0} \to \overline{0} \to \overline{0}$ be the function

$$
\begin{array}{l}
\lambda G^{\overline{2}}.\lambda f^{\overline{1}}.\mu m^{\overline{0}\to\overline{0}\to\overline{0}}.\lambda n^{\overline{0}}.\lambda g^{\overline{0}}. \\
\quad \mathsf{cond}\ F(\lambda p^{\overline{1}}.\, G(\lambda x^{\overline{0}}.\, \mathsf{cond}\ (x < n)\ (f\,x)(conv\,(p\,0)(f\,x)))) \\
\qquad g \\
\qquad \mathsf{cond}\ F(\lambda p^{\overline{1}}.\, G(\lambda x^{\overline{0}}.\, \mathsf{cond}\ (x \neq n)\ (f\,x)(conv\,(p\,0)(f\,x)))) \\
\qquad\quad (m\ (\mathsf{succ}\ n)\ g) \\
\qquad\quad (m\ (\mathsf{succ}\ n)\ (insert\,n\,(f\,n)\,g).
\end{array}
$$

Finally, take $M = \lambda Gf.\, M'\, G\, f\, 0\, \lceil\emptyset\rceil$. We leave it to the reader to check that M is indeed the modulus functional. □

More examples of degrees will appear in the next paragraph. A few further examples, and some discussion of their potential programming applications, are given in an electronically available Standard ML source file [33].

9.4 Types and universal functionals

We now ask what types a universal SR functional may have. Recall that H has call-by-value type $\overline{2} \to \overline{3}$. Since it is easy to construct a retraction $\overline{2} \to \overline{3} \lhd \overline{3}$, we certainly have a universal functional of call-by-value type $\overline{3}$. It is natural to ask whether there is a universal functional of any "lower" type, in either call-by-name or call-by-value PCF. We only need consider call-by-name types of the form

$$(\overline{0}^{r_1} \to \overline{0}) \to \cdots \to (\overline{0}^{r_n} \to \overline{0}) \to \overline{0},$$

since all other types below $\overline{3}$ are even lower than these. But any such type is a PCF-definable retract of some call-by-name type $\rho_r \equiv (\overline{0}^r \to \overline{0}) \to \overline{0}$, so it suffices to consider types ρ_r.

We will show, however, that there can be no such universal functional, and indeed the expressive power of the set of functionals of type ρ_r increases strictly with r. To this end we will exhibit, for each r, a denotational model of PCF that contains all the SR functionals of type ρ_r and below, but lacks certain functionals of type ρ_{r+1}.

Recall from [15] that **HC** is a full sub-CCC of the category of *dI-domains with coherence*, defined as follows.

Definition 9.14 *(i) A* dI-domain *is an ω-algebraic bounded-complete dcpo X such that*

- *for all $x, y, z \in X$ where y, z are bounded, $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$;*
- *every compact element has only finitely many lower bounds.*

We write **dI** *for the category of dI-domains and stable continuous functions.*

(ii) A coherence *on a dI-domain X is a family $\mathcal{C}$ of finite non-empty subsets of X, such that*

- *for all $x \in X$, $\{x\} \in \mathcal{C}$;*

- *if $A \in \mathcal{C}$ and $B \subseteq_{fin} X$ is below A in the Egli-Milner order (that is, $\forall x \in A. \exists y \in B. x \leq y$ and $\forall y \in B. \exists x \in A. x \leq y$), then $B \in \mathcal{C}$.*
- *if $D_1, \ldots, D_n \subseteq X$ are directed and for all $x_1 \in D_1, \ldots, x_n \in D_n$ we have $\{x_1, \ldots, x_n\} \in \mathcal{C}$, then $\{\bigsqcup D_1, \ldots, \bigsqcup D_n\} \in \mathcal{C}$.*

A dI-domain with coherence *is a dI-domain X equipped with a coherence $\mathcal{C}(X)$; elements of $\mathcal{C}(X)$ are called* coherent *sets. A* strongly stable *function between dI-domains with coherence is a continuous function that respects coherent sets and their meets. We write* **dIC** *for the category of dI-domains with coherence and strongly stable functions.*

(iii) If $\mathcal{C}$ is a coherence on X, a subset $\mathcal{B} \subseteq \mathcal{C}$ is a basis *for $\mathcal{C}$ if $\mathcal{C}$ is the coherence generated by $\mathcal{B}$ (that is, the smallest coherence on X containing $\mathcal{B}$).*

In a dI-domain with coherence, any *bounded* finite non-empty set is coherent. It follows that all strongly stable functions are stable; thus **dIC** is a (non-full) subcategory of **dI**.

The embedding of **HC** in **dIC** can be immediately read off from Definitions 5.5 and 5.6. We will write $N_\perp$ for the object of **dIC** corresponding to N in **HC**, and $[\![-]\!]$ for the interpretation of PCF types in **dIC**. Explicit definitions of products and exponentials in **dIC** can be found e.g. in [10].

We now define the models we want as full subcategories of **dIC**. We suppose r is any positive integer.

Definition 9.15 *(i) Let X be a dI-domain with coherence. A set $A \in \mathcal{C}(X)$ is* basic r-coherent *if there exist a coherent set $B \in \mathcal{C}(N_\perp^r)$ and a strongly stable function $f : N_\perp^r \to X$ such that $A = f(B)$. We say X is a* dI-domain with r-coherence *if the basic r-coherent sets form a basis for $\mathcal{C}(X)$. We write* $\mathbf{dIC}_r$ *for the category of dI-domains with r-coherence and strongly stable functions.*

It turns out that both $\mathbf{dIC}_1$ and $\mathbf{dIC}_2$ are equivalent to **dI**, though we shall not need this below. We will show that $\mathbf{dIC}_r$ is a CCC, but (crucially) the inclusion $\mathbf{dIC}_r \hookrightarrow \mathbf{dIC}$ does not preserve the CC structure. The following simple observations will be useful:

Lemma 9.16 *Suppose X, Y are dI-domains, $\mathcal{B}$ a basis for $\mathcal{C}(X)$, and $f : X \to Y$ a stable continuous map. If $f(A) \in \mathcal{C}(Y)$ for all $A \in \mathcal{B}$, then $f(A) \in \mathcal{C}(Y)$ for all $A \in \mathcal{C}(X)$.*

Proof Let $\mathcal{D} = \{A \in \mathcal{C}(X) \mid f(A) \in \mathcal{C}(Y)\}$. Then $\mathcal{B} \subseteq \mathcal{D}$, so to show that $\mathcal{D} = \mathcal{C}(X)$ it suffices to check that $\mathcal{D}$ is a coherence. That $\mathcal{D}$ contains all singletons is trivial. If $A \in \mathcal{D}, B \in \mathcal{C}(X)$ and $B \sqsubseteq A$ in the Egli-Milner order, then $f(B) \sqsubseteq f(A) \in \mathcal{C}(Y)$ and so $f(B) \in \mathcal{C}(Y)$. The closure of $\mathcal{D}$ under directed lubs follows easily from the continuity of f. $\square$

Proposition 9.17 *The category* $\mathbf{dIC}_r$ *has finite products.*

PROOF The terminal object presents no problem. Given objects X, Y of $\mathbf{dIC}_r$, let $X \times Y$ be the usual cartesian product of dI-domains, and let $\mathcal{C}(X \times Y)$ be the coherence generated by the basis

$$\mathcal{B} \;=\; \{\langle f, g\rangle(A) \mid f : N_\perp^r \to X, g : N_\perp^r \to Y \text{ strongly stable}, A \in \mathcal{C}(N_\perp^r)\}.$$

Clearly all the sets in $\mathcal{B}$ are then basic r-coherent, so $X \times Y$ is a dI-domain with r-coherence. The projection maps obviously respect all the coherent sets in $\mathcal{B}$, so by Lemma 9.16 they respect arbitrary coherent sets. Since they also preserve all existing meets, they are strongly stable.

Now suppose we are given an object Z of $\mathbf{dIC}_r$ and strongly stable maps $f : Z \to X$, $g : Z \to Y$. We wish to check that the stable and continuous function $\langle f, g\rangle : Z \to X \times Y$ is strongly stable. But if $A \in \mathcal{C}(Z)$ is a basic r-coherent set induced by $h : N_\perp^r \to Z$ then $\langle f, g\rangle(A)$ is a basic r-coherent set induced by $\langle fh, gh\rangle$. Thus $\langle f, g\rangle$ respects arbitrary coherent sets by Lemma 9.16. Moreover, it is easy to check that for any $A \in \mathcal{C}(Z)$ we have $\langle f, g\rangle(\bigwedge A) = \bigwedge\langle f, g\rangle(A)$. □

Proposition 9.18 *The category* $\mathbf{dIC}_r$ *has exponentials.*

PROOF Given objects X, Y of $\mathbf{dIC}_r$, let Y^X be the set of strongly stable functions $X \to Y$, endowed with the stable ordering; then Y^X is a dI-domain by [10, Proposition 14]. Now let $\mathcal{C}(Y^X)$ be the coherence generated by the basis

$$\mathcal{B} \;=\; \{(\mathit{curry}\; f)(A) \mid f : N_\perp^r \times X \to Y \text{ strongly stable}, A \in \mathcal{C}(N_\perp^r)\},$$

where $N_\perp^r \times X$ is as given by the previous proposition. Again, all the sets in $\mathcal{B}$ are then basic r-coherent, so Y^X is a dI-domain with r-coherence.

To see that the evaluation map $\epsilon : Y^X \times X \to Y$ is strongly stable, we compare our object Y^X with the exponential $W = Y^X$ calculated in $\mathbf{dIC}$ (see [10]). We see that Y^X and W have the same underlying dI-domain, and $\mathcal{B} \subseteq \mathcal{C}(W)$ so $\mathcal{C}(Y^X) \subseteq \mathcal{C}(W)$. Since the evaluation map $W \times X \to Y$ is known to be strongly stable, it follows that ϵ is strongly stable.

Now suppose we are given an object Z of $\mathbf{dIC}_r$ and a strongly stable map $f : Z \times X \to Y$. We wish to check that the stable and continuous function $\mathit{curry}\; f : Z \to Y^X$ is strongly stable. But if $A \in \mathcal{C}(Z)$ is a basic r-coherent set induced by $h : N_\perp^r \to Z$, then $(\mathit{curry}\; f)(A)$ is a basic r-coherent set induced by $\mathit{curry}(f \circ (h \times \mathrm{id}))$. Thus $\mathit{curry}\; f$ respects arbitrary coherent sets by Lemma 9.16. Moreover, for any $A \in \mathcal{C}(Z)$ we have $(\mathit{curry}\; f)(\bigwedge A) = \bigwedge(\mathit{curry}\; f)(A)$ in W, hence also in Y^X. □

Thus $\mathbf{dIC}_r$ is a cartesian closed category. Moreover, it is easy to see that $\mathbf{dIC}_r$ contains the dI-domain $N_\perp$ with the usual coherence, and that any morphism $f : X \to X$ in $\mathbf{dIC}_r$ has least fixed point $\bigsqcup f^i(\perp)$. It follows in the usual way that $\mathbf{dIC}_r$ is a model of PCF—that is, there is an adequate interpretation $[\![-]\!]_r$ of call-by-name PCF in $\mathbf{dIC}_r$, where $[\![\overline{0}]\!]_r = N_\perp$ and $[\![\sigma \to \tau]\!]_r = [\![\tau]\!]_r^{[\![\sigma]\!]_r}$ (calculated in $\mathbf{dIC}_r$).

Let us write $[N_\perp^k]$ for the product of k copies of $N_\perp$ computed in $\mathbf{dIC}$, and $[N_\perp^k]_r$ for the product computed in $\mathbf{dIC}_r$. The following proposition gives a measure of the similarity between the interpretations $[\![-]\!]_r$ and $[\![-]\!]$.

Proposition 9.19 *For any $1 \leq k \leq r$,*

(i) the objects $[N_\perp^k]_r$ and $[N_\perp^k]$ coincide;

(ii) the objects $[\![\, \overline{0}^k \to \overline{0} \,]\!]_r$ and $[\![\, \overline{0}^k \to \overline{0} \,]\!]$ have the same underlying dI-domain, and $\mathcal{C}([\![\, \overline{0}^k \to \overline{0} \,]\!]_r) \subseteq \mathcal{C}([\![\, \overline{0}^k \to \overline{0} \,]\!])$;

(iii) every element of $[\![\, \rho_k \,]\!]$ is also an element of $[\![\, \rho_k \,]\!]_r$.

Proof Part (i) follows from the easy observation that in $[N_\perp^k]$ every coherent set is basic r-coherent. Part (ii) follows from this by comparing exponentials in $\mathbf{dIC}_r$ and $\mathbf{dIC}$ (as in the proof of Proposition 9.18). It is immediate from this that every strongly stable function $[\![\, \overline{0}^k \to \overline{0} \,]\!] \to N_\perp$ is also a strongly stable function $[\![\, \overline{0}^k \to \overline{0} \,]\!]_r \to N_\perp$, which gives (iii). $\Box$

We now come to the main point of the exercise: we exhibit an element F_r of $[\![\, \rho_{r+1} \,]\!]$ that is not an element of $[\![\, \rho_{r+1} \,]\!]_r$. For $r \geq 1$ and $1 \leq i \leq r$, let $\epsilon_{ri} \in N_\perp^r$ be the tuple $(n-i+1, \ldots, n-1, \perp, 1, \ldots, n-i)$. Let $E_r = \{\epsilon_{r1}, \ldots, \epsilon_{rr}\}$; we call E_r the *rth Berry set*. Recall from [16] that E_r is coherent in $[N_\perp^r]$, but no proper subsets of E_r of size greater than 1 are coherent.

Lemma 9.20 *Suppose $k \geq 2$, and $A \subseteq N_\perp^k$ is such that $E_k \subseteq A$ and $E_k \sqsubseteq A$ in the Egli-Milner order. Then A is not basic r-coherent for any $r < k$.*

Proof Suppose for a contradiction this is false for some k and A. Take r minimal such that E_k is basic r-coherent; then clearly $r > 1$. Take $B \in \mathcal{C}(N_\perp^r)$ and $f : N_\perp^r \to N_\perp^k$ strongly stable such that $f(B) = A$. Let $f_i = \pi_i \circ f : N_\perp^r \to N_\perp$ for $1 \leq i \leq k$. Since none of the f_i are constant functions, each has some sequentiality index $1 \leq j_i \leq r$. So there must be two of the f_i, say f_{i_1} and f_{i_2}, with the same sequentiality index j. Now consider an arbitrary element $x = (x_1, \ldots, x_r) \in N_\perp^r$. Then $f_{i_1}(x) \in N$ if $f(x) \neq \epsilon_{ki_1}$, and similarly for i_2; so in any case, we have $x_j \in N$. So $\perp \notin \pi_j(B)$, and since $\pi_j(B) \in \mathcal{C}(N_\perp)$, this means $\pi_j(B)$ is a singleton. So by deleting the jth factor in $N_\perp^r$ we can exhibit A as basic $(r-1)$-coherent, contradicting the minimality of r. $\Box$

Let us write $\theta_{ri} : N_\perp^r \to N_\perp$ for the least monotone function such that $\theta_{ri}(\epsilon_{ri}) = 0$. Now take $\phi_r = \theta_{r1}$ and ψ_r the pointwise least upper bound of $\theta_{r2}, \ldots, \theta_{rr}$. It is easy to see that both ϕ_r and ψ_r are PCF-definable, and so are morphisms in $\mathbf{dIC}$ and in any $\mathbf{dIC}_{r'}$.

Proposition 9.21 *Suppose $r \geq 2$.*

(i) The set E_{r+1} is coherent in $[N_\perp^{r+1}]_{r+1}$ but not in $[N_\perp^{r+1}]_r$.

(ii) The elements $\theta_{(r+1)1}, \ldots, \theta_{(r+1)(r+1)}$ have an upper bound in $[\![\, \overline{0}^{r+1} \to \overline{0} \,]\!]_r$ but not in $[\![\, \overline{0}^{r+1} \to \overline{0} \,]\!]_{r+1}$.

(iii) In $[\![\, \rho_{r+1} \,]\!]$ and $[\![\, \rho_{r+1} \,]\!]_{r+1}$, but not in $[\![\, \rho_{r+1} \,]\!]_r$, there is a function T_{r+1} such that $T_{r+1}(\phi_{r+1}) = 0$ and $T_{r+1}(\psi_{r+1}) = 1$.

Proof (i) The first claim is immediate since E_{r+1} is coherent in $[N_\perp^{r+1}]$, which coincides with $[N_\perp^{r+1}]_{r+1}$. To prove that E_{r+1} is not coherent in $[N_\perp^{r+1}]_r$, first note that every directed set in $N_\perp^{r+1}$ already contains its lub; thus the first two closure conditions in Definition 9.14(ii) suffice to generate the coherence $\mathcal{C}([N_\perp^{r+1}]_r)$ from the basic r-coherent sets. So if E_{r+1} is coherent in $[N_\perp^{r+1}]_r$ then we have $E_{r+1} \sqsubseteq A$ in the Egli-Milner order, where A is either a singleton or basic r-coherent. Since $r \geq 2$, clearly A cannot be a singleton. But if A is basic r-coherent, by the above lemma we cannot have $E_{r+1} \subseteq A$. So take i such that $\epsilon_{(r+1)i} \notin A$; then clearly $\perp \notin \pi_i(A)$. But since $\pi_i(A)$ is coherent and is not a singleton, we have a contradiction.

(ii) Let θ be any upper bound for the $\theta_{(r+1)i}$ in the stable order; then $\theta(E_{r+1}) = \{1, \ldots, r+1\}$ is not coherent, so θ is not an element of $[\![\, \overline{0}^{r+1} \to \overline{0} \,]\!]_{r+1}$. Now let θ_{r+1} be the (pointwise) least upper bound of the $\theta_{(r+1)i}$; we claim that $\theta_{r+1} : [N_\perp^{r+1}]_r \to N_\perp$ is strongly stable. Suppose $A \in \mathcal{C}([N_\perp^{r+1}]_r)$. If $\perp \in \theta_{r+1}(A)$ then $\theta_{r+1}(A)$ is automatically coherent and $\theta_{r+1}(\bigwedge A) = \bigwedge \theta_{r+1}(A)$. Otherwise, we must have $\forall x \in A. \exists y \in E_{r+1}.\, y \leq x$. Let $E' = \{y \in E_{r+1} \mid \exists x \in A.\, y \leq x\}$; then $E' \sqsubseteq A$ and so $E' \in \mathcal{C}([N_\perp^{r+1}]_r)$. But $E_{r+1} \notin \mathcal{C}([N_\perp^{r+1}]_r)$ and no non-singleton proper subset of E_{r+1} is even in $\mathcal{C}([N_\perp^{r+1}])$, so E' is a singleton. It follows that $\theta_{r+1}(A)$ is a singleton (and hence coherent), and trivially $\theta_{r+1}(\bigwedge A) = \bigwedge \theta_{r+1}(A)$.

(iii) In $[\![\, \rho_{r+1} \,]\!]$ one may define T_{r+1} as follows:

$$T_{r+1}(f) \;=\; \begin{cases} 0 & \text{if } \phi_{r+1} \leq_s f, \\ 1 & \text{if } \psi_{r+1} \leq_s f, \\ \perp & \text{otherwise} \end{cases}$$

This function is well-defined by (ii). To see that it is strongly stable, suppose $A \in \mathcal{C}([\![\, \overline{0}^{r+1} \to \overline{0} \,]\!])$. If $T_{r+1}(A)$ is not coherent then we must have $T_{r+1}(A) = \{0, 1\}$, so $\{\theta_{(r+1)1}, \theta_{(r+1)2}\} \sqsubseteq \{\phi, \psi\} \sqsubseteq A$, so $\{\theta_{(r+1)1}, \theta_{(r+1)2}\}$ is coherent, a contradiction. Hence $T_{r+1}(A)$ is coherent, and a simple inspection of cases shows that $T_{r+1}(\bigwedge A) = \bigwedge T_{r+1}(A)$. Thus $T_{r+1} \in [\![\, \rho_{r+1} \,]\!]$, and it follows from Proposition 9.19(iii) that $T_{r+1} \in [\![\, \rho_{r+1} \,]\!]_{r+1}$.

However, no such function T_{r+1} can exist in $[\![\, \rho_{r+1} \,]\!]_r$, since $\phi_{r+1}, \psi_{r+1} \leq_s \theta_{r+1}$ but $0, 1$ have no upper bound in $N_\perp$. $\square$

Note that θ_3 is the *Gustave function*, the standard example (due to Berry) of a stable but not sequential function.

We may now read off the following result:

Theorem 9.22 *There is no SR functional G of any type ρ_r such that all effective SR functionals, or even all finite SR functionals, are PCF-definable relative to G.*

PROOF We show that the SR functional T_{r+1} (which is finite and hence effective) is not PCF-definable from G. Suppose $M : \rho_r \to \rho_{r+1}$ were a PCF term such that $[\![\, M \,]\!](G) = T_{r+1}$ in **dIC**. For each i let N_i be a PCF term denoting $\theta_{(r+1)i}$; then $[\![\, M \,]\!](G)([\![\, N_i \,]\!]) = i$ for each i. But the evident logical relation between the PCF types in **dIC** and **dIC**$_r$ relates G to G, $[\![\, M \,]\!]$ to $[\![\, M \,]\!]_r$, and $[\![\, N_i \,]\!]$ to $[\![\, N_i \,]\!]_r$ (using the logical relations lemma). It follows that $[\![\, M \,]\!]_r(G) \in [\![\, \rho_{r+1} \,]\!]_r$ fulfils the criteria for T_{r+1}, a contradiction. $\square$

Remarks 9.23 (i) In fact the model **dIC**$_r$ shows that T_{r+1} is not PCF-definable even from the set of all SR functionals of type ρ_r or below. This answers negatively a question implicitly posed at the very end of [17].

(ii) A simple corollary of the above theorem is that there is no *finite* SR functional G of any type whatever such that all the finite SR functionals are PCF-definable from G. For if such a G existed, it would be definable from some finite approximation H_i to H arising from the standard sequence of finite retractions; but each such H_i has the same degree as some functional of type ρ_r.

(iii) It is an easy exercise to show that each T_{r+1} is PCF-definable from T_{r+2}. We therefore have a strictly ascending chain of degrees $d(T_2) \prec d(T_3) \prec \ldots$. Note that T_2 is easily definable from the modulus functional, so $d(T_2) = d(F)$ by Proposition 9.13. This shows that $d(F)$ is strictly below $d(H)$.

10 Synthetic domain theory

In this section we revisit our models $\mathbf{Mod}(\mathcal{B}_{full})$ and $\mathbf{Mod}(\mathcal{B}_{eff})$ from the standpoint of *synthetic domain theory* (see e.g. [19, 35, 48, 52]). We will see that at least two existing versions of synthetic domain theory work out very well in these models; we also point out some ways in which our models are rather different in flavour from most of those previously considered.

10.1 Well-complete and replete objects

Consider again the lift functor $L = -_\perp$ on $\mathbf{Mod}(\mathcal{B})$ (where $\mathcal{B}$ is $\mathcal{B}_{full}$ or $\mathcal{B}_{eff}$). As observed in Section 3.1, it is easy to make this into a monad, i.e. to give natural transformations $\eta : \mathrm{id} \to L$ and $\mu : L^2 \to L$ satisfying the monad laws. Let Σ be the object $1_\perp$, and let $\top$ be the morphism $\eta_1 : 1 \to \Sigma$. Pullbacks of $\top$ along morphisms $X \to \Sigma$ are called *Σ-monos*. It is easy to see that $(\Sigma, \top)$ is a *dominance* on $\mathbf{Mod}(\mathcal{B})$—that is, any Σ-mono $Y \rightarrowtail X$ is the pullback of $\top$ along a unique *classifying map* $X \to \Sigma$, and Σ-monos are closed under composition. (For the latter claim, we use the fact that the "generic" composite of Σ-monos $\top_\perp \circ \top$ is classified by μ_1.) Indeed, in the terminology of [35], $(\Sigma, \top)$ arises from a *divergence* on $\mathcal{B}$, namely the singleton set $\{\perp\}$.

For general reasons, the functor $\perp$ has both an initial algebra ω and a final coalgebra $\overline{\omega}$ (see e.g. [35].) In the case of $\mathbf{Mod}(\mathcal{B})$, the following concrete presentations of ω and $\overline{\omega}$ are especially convenient. For each $n \in \mathrm{N}$ define $q_n \in \mathcal{B}$ by $q_n(m) = 0$ if $m < n$, $q_n(m) = \perp$ if $m \geq n$; and take $q_\infty = \lambda m.0$. The objects $\omega, \overline{\omega}$ may then be defined by

$$\begin{array}{rcl} |\omega| &=& \mathrm{N}, \quad \|n\|_\omega \;=\; \{q_n\}, \\ |\overline{\omega}| &=& \mathrm{N}, \quad \|n\|_{\overline{\omega}} \;=\; \{q_n\}, \quad \|\infty\| \;=\; \{q_\infty\}. \end{array}$$

The structure map $\omega_\perp \to \omega$ for the initial lift algebra is the isomorphism that just interchanges the points 0 and 1, and likewise for the final coalgebra. Moreover, the unique algebra map $\iota : \omega \to \overline{\omega}$ is the evident inclusion. In all versions synthetic domain theory, the morphism ι (or something very similar) plays a fundamental role in formulating an appropriate notion of "chain-completeness" for predomains.

We will briefly consider two possible such categories of predomains in $\mathbf{Mod}(\mathcal{B})$: the *well-complete* objects introduced by Longley and Simpson [35], and the *replete* objects considered by Hyland [19] and Taylor [55]. We will show that both these notions yield good categories of predomains. First recall the following definition from [35]:

Definition 10.1 (Well-completeness) *An object X is* complete *if $X^\iota : X^{\overline{\omega}} \to X^\omega$ is an isomorphism. An object X is* well-complete *if $X_\perp$ is complete.*

Actually the notions of complete and well-complete object coincide in $\mathbf{Mod}(\mathcal{B})$ (as they do in many other natural models), though we will not use this fact. The following proposition shows that *strong completeness axiom* of [35] holds in $\mathbf{Mod}(\mathcal{B})$:

Proposition 10.2 *The object N in $\mathbf{Mod}(\mathcal{B})$ is well-complete.*

PROOF Suppose c is any morphism $\omega \to N_\perp$, realized by r. Then each $r \bullet q_n$ realizes some element of N, and $r \bullet q_\infty = \bigsqcup r \bullet q_n$. But the set of realizers for elements of $N_\perp$ is (trivially) closed under lubs, and so $r \bullet q_\infty$ also realizes some element of N. Thus r realizes a (necessarily unique) morphism $\overline{\omega} \to N_\perp$ extending c. So $N_\perp^\iota : N_\perp^{\overline{\omega}} \to N_\perp^\omega$ is an isomorphism whose inverse is realized by $\lambda^* r.r$. $\square$

This shows that we are in the situation described for general realizability models in [31, 35], and indeed that all the axioms appearing in [48] or in [52] are true in $\mathbf{Mod}(\mathcal{B})$. This means that all the machinery of well-complete objects is available to us. In particular:

Theorem 10.3 *(i) The well-complete objects in $\mathbf{Mod}(\mathcal{B})$ are closed under finite limits, dependent products (including exponentials), finite sums, lifting and retracts, and contain the natural number object.*

(ii) For any well-complete object X equipped with a (necessarily unique) lift monad algebra structure $\alpha : X_\perp \to X$, every morphism $X \to X$ has a fixed point, and indeed there is a morphism $y_X : X^X \to X$ that computes it. Furthermore, for any $f : X \to X$ and any $t : X \to \Sigma$, we have $t(y_X f) = \top$ iff $t(f^n(\alpha\perp)) = \top$ for some n.

By the results of [31, Chapter 6], we also have adequate interpretations of call-by-name and call-by-value PCF. Moreover, the well-complete objects form a small complete category (i.e. an impredicative universe), so we can also interpret much more powerful type theories (see e.g. [18]).

Next we turn our attention to the *replete* objects in $\mathbf{Mod}(\mathcal{B})$. There are many equivalent definitions of repleteness (see e.g. [47]); here we recall a characterization due to Taylor [19].

Definition 10.4 (Repleteness) *A morphism $e : X \to Z$ is called* Σ-epi *if $\Sigma^e : \Sigma^Z \to \Sigma^X$ is mono. A morphism $f : X \rightarrowtail Y$ is called an* extremal mono *if, for any mono m and Σ-epi e with $f = m \circ e$, e is an isomorphism. An object X is* replete *if the canonical map $\epsilon_X : X \to \Sigma^{\Sigma^X}$ is an extremal mono.*

In the situation of [35] it is always the case that every replete object is well-complete; in fact, one can regard the well-complete objects as the largest good category of predomains, and the replete objects as the smallest.

Proposition 10.5 *The object N in* $\mathbf{Mod}(\mathcal{B})$ *is replete.*

Proof Consider the canonical map $\epsilon : N \to \Sigma^{\Sigma^N}$. We first show that this is a *regular mono*—that is, there is a realizer p such that $a \in \|\epsilon n\|$ implies $p \bullet a \in \|n\|$. For convenience we work with the following presentations of N, Σ and Σ^N: $\|n\|_N = \{r_n\}$ where $r_n(0) = n, r_n(m+1) = \bot$; $\|\top\|_\Sigma = \{r_0\}$ and $\|\bot\|_\Sigma = \{\lambda m.\bot\}$; and for $t : |N| \to |\Sigma|$ we have $\|t\|_{\Sigma^N} = \{r_t\}$, where $r_t n = 0$ if $tn = \top$, $r_t n = \bot$ if $tn = \bot$. In addition, we consider as realizers for $f \in |\Sigma^{\Sigma^N}|$ all $\bullet$-*irredundant* realizers for f considered as a morphism $\Sigma^N \to \Sigma$ (with respect to the above presentations).

Relative to these presentations, any realizer a for any $\epsilon n : \Sigma^N \to \Sigma$ clearly satisfies $a\langle 0\rangle =?n$; From this, it is easy to construct the required realizer p that extracts n from a. So ϵ is a regular mono; in particular, it is maximal among subobjects of Σ^{Σ^N} whose points are exactly the elements ϵn.

To see that ϵ is an extremal mono, note first that if a is a $\bullet$-irredundant realizer for any $f : \Sigma^N \to \Sigma$, then $f = \epsilon n$ iff $a\langle 0\rangle =?n$ and $a\langle 0,0\rangle =!0$. Using this, it is easy to construct a morphism $C : \Sigma^{\Sigma^N} \to \Sigma$ such that Cf is $\top$ if $f = \epsilon n$ for some n, and $Cf = \bot$ otherwise. Now suppose ϵ_N factors as $N \overset{e}{\to} Z \overset{m}{\to} \Sigma^{\Sigma^N}$ where m is mono and e is Σ-epi. Clearly the maps $C \circ m$ and $\lambda z.\top : Z \to \Sigma$ both induce the constant map $\lambda n.\top : N \to \Sigma$, and so $C \circ m = \lambda z.\top$ since e is Σ-epi. Thus the image of m contains only the points ϵn, and so e is an isomorphism by the above. $\Box$

For general reasons (see [48]) we can now deduce the analogue of Theorem 10.3 for the replete objects. So the replete objects also suffice for interpreting PCF and indeed much richer languages. We do not know whether there are objects of $\mathbf{Mod}(\mathcal{B})$ that are well-complete but not replete.

Remark 10.6 Clearly the category $\mathcal{K}$ of Section 8.2 lies within the category of replete objects; hence so does the image of $E : \mathbf{SSeqFun} \to \mathbf{Mod}(\mathcal{B})$. Likewise, we expect that some good subcategory of $\mathbf{HC}$ embeds well into the replete objects.

10.2 The Σ-order

We now consider another basic notion from synthetic domain theory: the Σ-*order* (or *intrinsic order*) on objects. We shall see that in $\mathbf{Mod}(\mathcal{B})$ this coincides exactly with the stable order (at least for finite types). This comes as no surprise after the results of Section 5, but it means that the model $\mathbf{Mod}(\mathcal{B})$ has a completely different flavour from any of the models considered e.g. in [35], in which the Σ-order (seemingly) coincides with the pointwise order.

For any modest set X, the *(external)* Σ-*preorder* $\preceq_X$ on $|X|$ is defined by

$$x \preceq_X y \iff \forall t : X \to \Sigma.\ (tx = \top \implies ty = \top).$$

Clearly all morphisms between modest sets preserve the Σ-preorder. It is easy to see that in $\mathbf{Mod}(\mathcal{B})$ the Σ-preorder on $N_\perp$ coincides with the usual partial order; and hence that for all finite types over $N_\perp$ the Σ-preorder is a partial order. Since Σ is a retract of $N_\perp$, the definition of the Σ-order on the finite types R^σ can be formulated as follows:

$$x \preceq_\sigma y \iff \forall t \in R^{\sigma \to \overline{0}}.\ (t \cdot x = 0 \implies t \cdot y = 0).$$

This emphasizes that the Σ-order is intrinsic to the type structure R and not dependent on any particular model for R such as the category $\mathbf{Mod}(\mathcal{B})$.

Proposition 10.7 *For $x, y \in R^\sigma_{full}$, the following are equivalent:*
(i) $x \preceq_\sigma y$.
(ii) $x \sqsubseteq_\sigma y$ in the sense of Proposition 5.13.
(iii) x, y are path-related*, i.e. there exists $p \in R^{\overline{0} \to \sigma}$ with $p \cdot \perp = x$ and $p \cdot \perp = y$.*

Proof For (i) $\Rightarrow$ (ii), it seems easiest to work with the characterization of $\sqsubseteq_\sigma$ given by the hypercoherence model (see Proposition 5.13). For each atom a in the underlying set of the hypercoherence $X = [\![\, \sigma \,]\!]_{HC}$, let $t_a : X \to N$ be the strongly stable function whose trace is $\{(\{a\}, 0)\}$; then for $z \in D(X)$ we have $t_a(z) = 0$ if $a \in z$ and $t_a(z) = \perp$ otherwise. So if $x \preceq_\sigma y$ then (regarding x, y as states of X), for each a we have

$$a \in x \implies t_a(x) = 0 \implies t_a(y) = 0 \implies a \in y.$$

Thus $x \subseteq y$, and so $x \sqsubseteq_\sigma y$ by Proposition 5.13.

(ii) $\Rightarrow$ (iii): Suppose $x \sqsubseteq_\sigma y$, i.e. there exist realizers $a \in \|x\|$, $b \in \|y\|$ with $a \sqsubseteq b$. It is trivial to construct a realizer r such that $r \bullet (\lambda n.\perp) = a$ and $r \bullet r_n = b$ for any n (where $r_n(0) = n$ and $r_n(m+1) = \perp$ as above). Clearly r realizes a morphism p with the required properties.

(iii) $\Rightarrow$ (i): Given $p \in R^{\overline{0} \to \Sigma}$ as above and $t \in R^{\sigma \to \overline{0}}$, if $t \cdot x = t \cdot (p \cdot \perp) = 0$ then $t \cdot y = t \cdot (p \cdot \perp) = 0$ since the map $\mathrm{N}_\perp \to \mathrm{N}_\perp$ given by $t \circ p$ is monotone. $\square$

Note that the implication (ii) $\Rightarrow$ (iii) fails in the effective case: if $x, y \in R^{\overline{1}}$ correspond to two partial recursive functions such that $x \subseteq y$ but dom y − dom x is not r.e., there is no effective realizer for an element p with the required properties.

It follows from the above that the Σ-order on Σ^Σ is given by $\lambda x.\perp \preceq \lambda x.x$, $\lambda x.\perp \preceq \lambda x.\top$. Indeed, it is amusing to note that the functional F described in the Introduction is actually an isomorphism $\Sigma^\Sigma \cong 2_\perp$ in $\mathbf{Mod}(\mathcal{B})$. Likewise, one can easily show that $\Sigma^\omega \cong \Sigma^{\overline{\omega}} \cong N_\perp$ in $\mathbf{Mod}(\mathcal{B})$.

Remark 10.8 The model $\mathbf{Mod}(\mathcal{B}_2)$ also fits well with the theory of well-complete objects: the divergence $\{\perp\}$ gives rise to a dominance in $\mathbf{Mod}(\mathcal{B}_2)$ satisfying the Strong Completeness Axiom. As mentioned in Section 8.2, $\mathbf{Mod}(\mathcal{B}_2)$ also has the potential advantage that most of the objects of interest are projective. It would be interesting to know whether the modified realizability model $\mathbf{mRT}(\mathcal{B})$ of Section 3.3 and the presheaf model $[\mathcal{M}^{op}, \mathbf{Set}]$ in Section 6 also fit smoothly into some version of synthetic domain theory.

11 Notions of higher-type computability

We end the main body of the paper by comparing the SR functionals with other known type structures. The line of thought we describe here is part of a general investigation of notions of computability at higher types, which will be developed more fully in a forthcoming survey paper [30].

Our motivating philosophy is as follows: since it is not clear *a priori* what should be meant by the "computable" functionals of higher types, we take a step back and consider the space of all possible notions of higher-type computability. More precisely, we define a general concept of "class of computable functionals", in such a way that any reasonable notion of higher-type computability can be expected to provide an example of such a class. Within this abstract framework, we can then collect particular notions of higher-type computability, and ask how they are related.

11.1 General definitions

Here we shall restrict attention to notions of *(hereditarily) partial* computable functional. In general, a class of partial higher-type functionals is embodied by a *partial type structure*, as in Definition 3.6(i) (see also Section 7.1). For definiteness, we will concentrate here on the call-by-name notion of type structure, though (as usual) this choice does not matter much. We repeat the definition here for convenience.

Definition 11.1 *A* partial type structure (PTS) T *consists of*

- *a set* T^σ *for each type* σ*, where* $T^{\overline{0}} = \mathrm{N}_\perp$*,*
- *for each* σ, τ *a total "application" function* $\cdot_{\sigma\tau} : T^{\sigma\to\tau} \times T^\sigma \to T^\tau$*.*

We say T *is* extensional *(or is an* EPTS*) if, for all types* σ, τ *and all* $f, g \in T^{\sigma\to\tau}$*,*

$$(\forall x \in T^\sigma.\, f \cdot x = g \cdot x) \implies f = g.$$

It seems clear that any reasonable notion of computable partial functional will give rise to a PTS T in this sense, since we expect computable functionals to be closed under application. We would also expect T to be extensional, since the elements of $T^{\sigma\to\tau}$ are just functions.[5] Moreover, if T consists of computable partial functions, we expect it to be *effective* in some way. We can build a notion of effectivity onto the above definition as follows.

Definition 11.2 *An* effective structure *on a PTS* T *assigns to each element* f *of each* T^σ *a non-empty set* $\|f\|_\sigma \subseteq \mathrm{N}$ *of* realizers *for* f*, such that:*

- *for each* σ, τ *there is a partial recursive function* $\varphi_{\sigma\tau} : \mathrm{N} \times \mathrm{N} \rightharpoonup \mathrm{N}$ *such that, for all* $f \in T^{\sigma\to\tau}, x \in T^\sigma, m, n \in \mathrm{N}$ *we have*

$$m \in \|f\|_{\sigma\to\tau} \ \wedge\ n \in \|x\|_\sigma \implies \varphi(m,n) \in \|f \cdot x\|_\tau;$$

[5] This is a slight oversimplification: if the domain of definition of the functions in $T^{\sigma\to\tau}$ were larger than T^σ, one could imagine distinct elements $f, g \in T^{\sigma\to\tau}$ that had the same restriction to σ. We will discuss the issue of extensionality in more detail in [30].

- *there is a partial recursive function* $\psi : \mathrm{N} \rightharpoonup \mathrm{N}$ *such that for all* $n \in \mathrm{N}$ *we have*

$$\psi(n)\downarrow \implies n \in \|\psi(n)\|_{\overline{0}}, \qquad \psi(n)\uparrow \implies n \in \|\perp\|_{\overline{0}}.$$

A effective PTS *is a PTS that admits at least one effective structure.*

The reader may recognize that a PTS with effective structure is nothing other than a certain kind of PTS within the realizability model on the partial combinatory algebra K_1. The above definition is designed to be as liberal as possible so as to admit any conceivable notion of computable functional—for instance, there is no requirement that the set $\bigcup_{x \in T^\sigma} \|x\|_\sigma$ of realizers for objects of type σ be recursively enumerable. Indeed, it seems reasonable to propose the following informal hypothesis:

Thesis 11.3 *Any reasonable notion of computable hereditarily partial functional at higher types (henceforth "notion of computability") is embodied in an effective extensional PTS.*

(Of course, there are also many silly effective EPTSs that do not embody a reasonable notion of computability.) The above claim rests on Church's Thesis and on the intuitive idea that "effective" objects of any kind are ultimately representable by natural numbers. It also depends on the supposition, true in all known cases of interest, that any reasonable class of functionals (for example, a class given as a call-by-value type structure) can be embodied in an "equivalent" call-by-name type structure. In the absence of any evidence to the contrary, we adopt the above thesis as a working hypothesis.

Given two EPTSs, it is natural to ask when one of them in some sense "contains" the other. Again, we seek the most liberal notion of containment that seems reasonable—for instance, the obvious definition of homomorphic inclusion, and even that of logical relation, seem too strong for our purposes. We propose the following notion:

Definition 11.4 *Let T, U be EPTSs. A* simulation $s : T \to U$ *consists of a total relation* $s^\sigma : T^\sigma \twoheadrightarrow U^\sigma$ *for each type* σ*, such that*

- $s^{\overline{0}}$ *is the identity relation on* $\mathrm{N}_\perp$*;*
- *for any* $f \in T^{\sigma\to\tau}, g \in U^{\sigma\to\tau}, x \in T^\sigma, y \in U^\sigma$ *we have*

$$s^{\sigma\to\tau}(f, g) \;\wedge\; s^\sigma(x, y) \implies s^\tau(f \cdot x, g \cdot y).$$

We write $T \le U$ *if there exists a simulation* $s : T \to U$.

Thesis 11.5 *If U embodies a notion of computability that includes or subsumes the notion embodied by T in any reasonable sense, then* $T \le U$.

The intuition is that for any element x of T, there is at least one element y of U that represents or "simulates" x. Note that $T \le U$ iff T arises as a homomorphic *subquotient* of U (where homomorphisms are required to be the identity on type $\overline{0}$).

It is clear that EPTSs and simulations between them form a category. It is also easy to see that the only simulation $T \to T$ is the identity; it follows that the relation $\le$ is a partial order on EPTSs. We will write $\mathcal{P}$ for the poset of EPTSs ordered by $\le$, and $\mathcal{P}_{\mathit{eff}}$ for the full sub-poset of effective EPTSs. (Note that we not impose any effectivity conditions on simulations between effective EPTSs.)

11.2 Three notions of computability

We now consider three important examples of effective EPTSs. These correspond to the three good candidates for a natural notion of computable partial functional at higher types that we are currently aware of, and each of them admits several different characterizations.

Examples 11.6 (i) The *effective continuous functionals*—that is, those present as effective elements of finite types in the standard Scott model—constitute an effective EPTS. By a universality result due to Plotkin [45] and Sazonov [49], all such functionals are definable in the programming language PCF + parallel-or + exists (we call this language PCF^{++}). It follows that this EPTS can be characterized syntactically as the term model for PCF^{++} (that is, the type structure of closed PCF^{++} terms modulo observational equivalence). It is also the EPTS arising from the familiar category of modest sets over K_1 (see e.g. [31, Chapter 7]). There are many other mathematical characterizations of this type structure—these will be surveyed in [30].

(ii) The *effective PCF-sequential functionals* constitute an effective EPTS. Syntactically this can be characterized as the term model for pure PCF. Semantic characterizations of this type structure are provided by the effective versions of the various game models for PCF [2, 21, 40]. There are also characterizations via realizability models (see [32] for a survey).

(iii) The *effective SR functionals*, as described in this paper, clearly constitute an effective EPTS. Of the many characterizations we have discussed, the syntactic characterization as the term model for PCF+H is most relevant to our present concerns.

Thesis 11.7 *The above three type structures all represent reasonable notions of computability.*

For convenience we name each of these type structures after the corresponding programming language, referring to them as $T(\mathrm{PCF}^{++})$, $T(\mathrm{PCF})$ and $T(\mathrm{PCF}+\mathsf{H})$ respectively. In each case, an effective structure on the PTS can be obtained from the corresponding language via a Gödel-numbering of closed terms (note that evaluation of terms of type $\overline{0}$ is effective, and the application operation can be simply juxtaposition of terms.) In fact, for the arguments below we only need to posit that $T(\mathrm{PCF}^{++})$ and $T(\mathrm{PCF}+\mathsf{H})$ represent reasonable notions of computability.

It is easy to see that the syntactic inclusions PCF $\hookrightarrow$ PCF^{++} and PCF $\hookrightarrow$ PCF+H induce simulations between the corresponding term models, so that we have $T(\mathrm{PCF}) \leq T(\mathrm{PCF}^{++})$ and $T(\mathrm{PCF}) \leq T(\mathrm{PCF}{+}\mathsf{H})$. (Note that the total relations in question will not all be functions, since PCF terms that are observationally equivalent in PCF need not be so in PCF^{++} or PCF+H.)

Of course, many other interesting EPTSs are known. Some of these, such as the effective part of Berry's *stable* model [4], appear to be mathematically natural, but no reasonable computational interpretation for the corresponding functionals is known. Others arise from degrees of parallelism [resp. the degrees defined in Section 9.3], and consequently lie between $T(\mathrm{PCF})$ and $T(\mathrm{PCF}^{++})$ [resp. $T(\mathrm{PCF}{+}\mathsf{H})$]. Whilst these EPTSs do have a computational interpretation, none of them has yet emerged as particularly *canonical* in the sense that it admits other mathematical characterizations. It is for these reasons that we claim that our three EPTSs represent the only natural notions of partial higher-type computability known to date.

The following facts are easily established:

Proposition 11.8 *(i)* $T(\mathrm{PCF}^{++}) \not\leq T(\mathrm{PCF}{+}\mathsf{H})$.
(ii) $T(\mathrm{PCF}{+}\mathsf{H}) \not\leq T(\mathrm{PCF}^{++})$.

Proof (i) Suppose we had a simulation $s : T(\mathrm{PCF}^{++}) \to T(\mathrm{PCF}{+}\mathsf{H})$. Let $p \in T(\mathrm{PCF}^{++})^{\overline{0}\to\overline{0}\to\overline{0}}$ be some parallel operation—for instance, suppose that $pxy = \bot$ iff $x = y = \bot$. Take $q \in T(\mathrm{PCF}{+}\mathsf{H})^{\overline{0}\to\overline{0}\to\overline{0}}$ such that $s(p,q)$. Since $s^{\overline{0}} = \mathrm{id}$, it is clear that $q = p$ as a function $\mathrm{N}_\bot \times \mathrm{N}_\bot \to \mathrm{N}_\bot$. But this is a contradiction, since p is not Milner-Vuillemin sequential and hence not SR.

(ii) Conversely, suppose we had a simulation $t : T(\mathrm{PCF}{+}\mathsf{H}) \to T(\mathrm{PCF}^{++})$. Let $F \in T(\mathrm{PCF}{+}\mathsf{H})^{\overline{2}}$ be the functional defined in Remark 3.7(ii), and take $G \in T(\mathrm{PCF}^{++})^{\overline{2}}$ such that $t(F,G)$. Then clearly G also satisfies the specification given in Remark 3.7(ii). But this is impossible, since every $G \in T(\mathrm{PCF}^{++})^{\overline{2}}$ is monotone with respect to the pointwise order. $\Box$

The relationships between our three type structures in $\mathcal{P}_{\mathit{eff}}$ may therefore be depicted as follows:

$$\begin{tikzcd}
T(\mathrm{PCF}^{++}) \arrow[dr, dash] & & T(\mathrm{PCF}{+}\mathsf{H}) \arrow[dl, dash] \\
& T(\mathrm{PCF}) &
\end{tikzcd}$$

It is already interesting to note that there are two seemingly natural but incomparable notions of computability. One might now ask whether there is some more generous notion of computability that subsumes both of them. We shall see that the answer is no: the notions of computability corresponding to PCF^{++} and PCF+H are essentially incompatible.

We will write T for the type structure $T(\mathrm{PCF}^{++})$, We assume some familiarity with the characterization of T as the EPTS arising from $\mathbf{Mod}(K_1)$ (see [31]), and we write $\| - \|^T$ for the effective structure on T given by $\mathbf{Mod}(K_1)$.

Proposition 11.9 *For each type σ there is a total recursive function $\epsilon_\sigma : \mathrm{N} \to \mathrm{N}$ such that*

- *for any $n \in \mathrm{N}$, $\epsilon_\sigma(n) \in \|x\|^T_\sigma$ for some $x \in T^\sigma$;*
- *if $n \in \|x\|^T_\sigma$ then also $\epsilon_\sigma(n) \in \|x\|^T_\sigma$.*

Proof It is straightforward to construct such a function for the type $\overline{1}$. The result for arbitrary types then follows from the fact that every type T^σ is a PCF^{++}-definable retract of $T^{\overline{1}}$ (see [34, Lemma 5.2]). □

Lemma 11.10 *The type structure T is a maximal element of $\mathcal{P}_{\mathit{eff}}$.*

Proof Suppose we have an EPTS U with effective structure $\| - \|^U$, and a simulation $s : T \to U$. We can regard each structure $(U^\sigma, \| - \|^U_\sigma)$, as well as each $(T^\sigma, \| - \|^T_\sigma)$, as an object in $\mathbf{Mod}(K_1)$. We will show that there are isomorphisms $T^\sigma \cong U^\sigma$ in $\mathbf{Mod}(K_1)$ that respect application.

For $\overline{0}$ the bijection $T^{\overline{0}} \cong U^{\overline{0}}$ is immediate, and the fact that this is an isomorphism in $\mathbf{Mod}(K_1)$ follows from the existence of the functions ψ of Definition 11.4. For type $\sigma \to \tau$, given any $f \in U^{\sigma\to\tau}$ and any $n \in \|f\|^U$ we may recursively obtain a Kleene index tracking $f \cdot - : U^\sigma \to U^\tau$. By the recursive equivalence of T, U at σ and τ we may thence recursively obtain a Kleene index tracking $f \cdot - : T^\sigma \to T^\tau$, that is, an element of $\|f\|^T_{\sigma\to\tau}$. We thus have a morphism $U^{\sigma\to\tau} \to T^{\sigma\to\tau}$ in $\mathbf{Mod}(K_1)$ that respects application. Conversely, for every $f \in T^{\sigma\to\tau}$ there exists $f' \in U^{\sigma\to\tau}$ such that $s(f, f')$, and so we certainly have a function $f \mapsto f'$ respecting application. To see that this is a morphism $T^{\sigma\to\tau} \to U^{\sigma\to\tau}$, we use the function $\epsilon_{\sigma\to\tau}$ of the above proposition. By extending $\epsilon_{\sigma\to\tau}$ to a strict function we can clearly regard $\epsilon_{\sigma\to\tau}$ as an element of $T^{\overline{0}\to(\sigma\to\tau)}$. So take $e \in U^{\overline{0}\to(\sigma\to\tau)}$ such that $s(\epsilon_{\sigma\to\tau}, e)$; then from an element of $\|e\|^U$ one may construct a Kleene index for the morphism $N \to U^{\sigma\to\tau}$ corresponding to e, and this index tracks the required morphism $T^{\sigma\to\tau} \to U^{\sigma\to\tau}$. □

Theorem 11.11 *The type structures $T(\mathrm{PCF}^{++})$ and $T(\mathrm{PCF}{+}\mathsf{H})$ have no upper bound in $\mathcal{P}_{\mathit{eff}}$. Hence $\mathcal{P}_{\mathit{eff}}$ contains no top element.*

Proof Immediate from the above lemma and Proposition 11.8(ii). □

From this and the above theses, it follows that there can be no reasonable notion of computability that subsumes all other reasonable notions—that is, there is no ultimate class of computable functionals containing "all" computable functionals of higher type. This observation seems to hold some philosophical interest in its own right, and is closely related to a problem posed by Kleene in [24, §1.2] (also quoted in [21]):

> I aim to generate a class of functions ... which shall coincide with all the partial functions which are "computable" or "effectively decidable", so that Church's 1936 will apply with the higher types included.

The above results suggest that Kleene's problem, understood in a certain way, has no solution. If one is seeking a notion of computability that is inclusive as possible, one inevitably has to choose to exclude certain kinds of functional in order to include others. This fact may appear paradoxical at first sight—for instance, one might wonder whether there was not some class of functionals computable in the language PCF^{++}+H. The answer is that any attempt to design an operational semantics for such a language results in a system that is either non-effective or non-deterministic (depending on exactly how it is attempted): the question of whether a function needs to look at its argument in order to return a result cannot be answered effectively when we are dealing with infinitely many parallel threads of computation. Another way to say this is that the question of whether f looks at its argument is only sensible for intensional representations of f of a certain restricted kind.

Whilst our argument depends on the thesis that $T(\mathrm{PCF}^{++})$ and $T(\mathrm{PCF}+\mathsf{H})$ both represent reasonable notions of computability, we have not needed to assume that the type structures we have considered represent the *only* reasonable such notions. Indeed, there appears to be no evidence for this at present, beyond the fact that no other compelling notions of computability are currently known. Thus, there remains the intriguing possibility that there are other natural notions awaiting discovery, represented by elements of $\mathcal{P}_{\mathit{eff}}$ but embodying some computational principle quite different from those above.

We conclude with some scattered observations about the poset $\mathcal{P}_{\mathit{eff}}$. First, it is natural to ask whether $T(\mathrm{PCF}+\mathsf{H})$ is also a maximal element. In fact it is not: the effective elements of the hypercoherence model clearly constitute an effective EPTS, and it follows from the results of Section 5.3 that this is strictly larger than $T(\mathrm{PCF}+\mathsf{H})$. However, it would appear that this larger type structure is a distraction since it does not embody an interesting computational notion—certainly it is not "effectively sequential" in any way. We would conjecture that, in some informal sense, $T(\mathrm{PCF}+\mathsf{H})$ represents a maximal notion of effectively sequential functional, and perhaps even the unique maximal notion.

Secondly, one might ask whether $T(\mathrm{PCF})$ is a greatest lower bound in $\mathcal{P}_{\mathit{eff}}$ for $T(\mathrm{PCF}^{++})$ and $T(\mathrm{PCF}+\mathsf{H})$. In general, one can construct a greatest lower bound of two EPTSs T, U (not necessarily the unique one) by forming the product $T \times U$ and then taking the subquotient by the unary logical relation induced by the diagonal relation at ground type. Clearly if T, U are effective then so is the resulting type structure. However, if this is construction is applied to $T(\mathrm{PCF}^{++})$ and $T(\mathrm{PCF}+\mathsf{H})$, the resulting type structure V is strictly greater than $T(\mathrm{PCF})$. This can be seen from the fact that both $T(\mathrm{PCF}^{++})$ and $T(\mathrm{PCF}+\mathsf{H})$, and hence also V, contain a function $C : (\overline{0}^2 \to \overline{0}) \to \overline{0}$ corresponding to Curien's Third Counterexample (see [14, p.269]), which is not PCF-definable:

$$C\,f \;=\; \begin{cases} 0 & \text{if } f\,0\,\bot = 0 \;\text{ or }\; f\,\bot\,0 = 0 \;\text{ or }\; f\,\bot\,1 = f\,1\,0 = 0, \\ \bot & \text{otherwise.} \end{cases}$$

The idea that this counterexample survives even when the Scott model and the strongly stable model are combined is already implicit in [10]. Operationally, the function C can be implemented using either parallel operations such as $\mathsf{parallel\text{-}or}$ or SR functions such as H, though it cannot be implemented in pure PCF.

12 Conclusions and further directions

12.1 Review of results

We have presented a number of contrasting but equivalent characterizations of the type structures R, R_{eff}. For convenience we summarize them here (glossing over the difference between the call-by-name and call-by-value variants).

1. The type structure given by standard realizability over the combinatory algebra $\mathcal{B}$ or $\mathcal{B}_{\mathit{eff}}$ (Definition 3.6).
2. The type structure given by modified realizability over $\mathcal{B}$ or $\mathcal{B}_{\mathit{eff}}$ (Proposition 3.9).
3. The extensional collapse of the [effective] sequential algorithms model (Corollary 5.3).
4. (R_{eff} only) The extensional collapse of the programming languages μPCF and PCF+catch (Remark 5.4).
5. (R only) The type structure in the strongly stable model **HC** (Corollary 5.8; Section 8.1).
6. The type structure in the presheaf category $[\mathcal{M}^{op}, \mathbf{Set}]$ (Section 6.2).
7. (R_{eff} only) The closed term model of the language PCF+H (Section 9.2).

A few of these characterizations (e.g. 1,6) seem sufficiently simple and appealing that even by themselves they suggest that the SR functionals are a mathematically natural object of study. More importantly, however, the fact that so many constructions of such diverse characters all give rise to the same type structure creates a strong impression that this type structure is a highly canonical mathematical object.

The above list provides a good example of how different mathematical characterizations can illuminate different facets of the same object. For instance, the characterizations 1–4 all have an intensional or operational flavour, and they show how the SR functionals arise from a natural and very general notion of sequential process. These constructions are all basically "extensional collapses" of some kind—that is, at each type level they pick out the set of elements that "just happen" to behave extensionally—thus, they do not immediately give much of a grasp on what these elements are. (Note, however, that characterization 1 leads to a proof of the universality of type $\overline{2}$, and we do not know whether this result can be obtained easily by any other means.) By contrast, the strongly stable model, while it does not directly reveal the computational aspect of the SR functionals, does give a "finitary" characterization of them in terms of a preservation property, and hence yields effective information (such as the decidability of equality for finite elements) that is not evident from any of the other characterizations. Likewise, the language PCF+H, while seemingly artificial in itself, provides a pleasing "constructive" handle on R_{eff} which is not given by any of the other descriptions—we can recursively enumerate all the effective SR functionals and only them. The functional H can also be used to establish the relationship between the full and effective type structures (Theorem 9.10).

We have also shown that the SR functionals enjoy some pleasing properties not shared by the PCF-sequential functionals: for instance, the decidable presentation of the finite elements (Proposition 5.10) and the existence of a universal type (Theorem 7.11). These results indicate that the class of SR functionals is in some sense of a lower "logical complexity" than the class of PCF-sequential functionals.

Though our main focus has been on the type structures R and R_{eff}, we believe our results also indicate that van Oosten's combinatory algebra $\mathcal{B}$ is an attractive object worthy of study in its own right. It provides a simple mathematical setting in which sequential algorithms have a natural home, and it has directly inspired our construction of a retraction $\overline{3} \lhd \overline{2}$. Moreover, the corresponding notion of realizability coincides with that embodied by Abramsky's combinatory algebra $\mathcal{A}$ (see Remark 2.8). In addition, we suspect that the combinatory algebra $\mathcal{B}_2$, discussed in Section 8.2, will also turn out to be a natural and important object.

12.2 The meaning of "sequentiality"

Having discussed the mathematical status of the SR functionals, it is worth considering their computational aspects, and in particular, examining more closely the claim that the SR functionals are "sequential" in some reasonable sense. As is pointed out in the Introduction to [2], there is a tension inherent in the the phrase "sequential functional", since sequentiality refers primarily to a computational process rather than a mere function. Clearly, by specifying an abstract type structure such as R_{eff}, or even a functional such as H, we do not commit ourselves to any particular kind of operational implementation. Indeed, many different operational realizations of R_{eff} are possible (see e.g. Section 12.3 below). The claim that the SR functionals are sequential, therefore, can only mean (in our view) that at least some reasonable realizations of them have a "sequential" character.

The question, then, is what sequentiality should mean for computational processes. The problem here is that there is no agreed general definition of "sequentiality" for processes of higher type. We can therefore argue in two ways: we can demonstrate the relationship with other things that have been called "sequential", or we can appeal to an informal understanding of what the word means. Regarding the first of these, we have shown that the SR functionals can all be computed by sequential algorithms, and that they can be implemented in languages such as PCF+catch and μPCF which are widely referred to as sequential.

Regarding the informal concept of sequentiality, it is instructive to compare a language such as PCF+catch with pure PCF. Intuitively, in both languages we can only "do one thing at a time": slightly more precisely, we cannot have two subcomputations that are directly triggered by the same function call in progress at the same time. This contrasts with PCF+parallel-or, even when the latter is implemented using deterministic reduction rules, for example:

$$\text{if } M \to M' \quad \text{then } \mathsf{parallel\text{-}or}\, MN \to \mathsf{parallel\text{-}or}\, NM'.$$

In PCF, however, there is an additional requirement: each subcomputation must be "completed" before another subcomputation, directly triggered by the same function call, can be begun. (In game-theoretic terms, this can be enforced by a *well-bracketing* condition.) This requirement is not satisfied by PCF+catch, since the catch operator allows subcomputations to be aborted before they are completed. Our view is that sequentiality informally means "doing only one thing at a time", which is not the same thing as "completing one thing before starting another". On this informal understanding, the languages PCF and PCF+catch are both sequential, while PCF+parallel-or is not. Insofar as PCF+catch embodies one reasonable realization of the SR functionals, we therefore feel justified in referring to the SR functionals as sequential.

The above discussion suggests that, from at least one point of view, the difference between PCF-sequentiality and sequential realizability lies in whether or not computations are required to be well-bracketed. The same message is reinforced by Abramsky's combinatory algebra $\mathcal{A}$ and its well-bracketed subalgebra $\mathcal{A}_{wb}$ (see Remark 2.8), which give realizability models of the SR and PCF-sequential functionals respectively.

We are anxious to point out, however, that we do not see the notions of sequential realizability and PCF-sequentiality as being in competition in any way. In our view, they both represent natural and compelling notions of sequentiality at higher types, and we are not inclined to argue over which is "the more canonical". It is true, as we have seen, that in some ways the SR functionals have better *mathematical* properties than the PCF ones, but we think these should be distinguished from questions about the *conceptual* status of the two notions.

12.3 Game-theoretic models

Perhaps the biggest omission from our theoretical investigation of the SR functionals concerns their relationship to *game models* for sequential computation (see e.g. [2, 21]). Here we will briefly describe the situation as we see it, and suggest some questions for further investigation.

Abramsky has proposed an "intensional hierarchy" of computational phenomena, such as state and control features, which can be captured semantically via various relaxations of the constraints on strategies in the basic game model for PCF. Roughly speaking, for deterministic sequential languages there appear to be two main axes of interest along which such relaxations can be made: we may weaken either the *well-bracketing* condition on strategies (leading to models for languages with control features—see [27]), or the *innocence* condition (leading to models for languages with state—see e.g. [1]). These conditions can be relaxed to various degrees, and also in combination (this seems to be necessary to provide good models for ML-style exceptions, for example). In general, one tries to correlate particular programming language features with particular (combinations of) conditions on strategies, by means of full abstraction or universality results.

The recent Ph.D. thesis of Laird [28] makes explicit some of the connections between this work in game semantics and the results of the present paper. He considers a class of *innocent unbracketed strategies*, and shows that its observational quotient coincides exactly with the sequential algorithms model. It follows immediately from the results of our Section 5.1 that the extensional collapse of the innocent unbracketed games model yields the SR functionals, and similarly for the effective analogue. Laird also shows that every effective innocent unbracketed strategy is definable in μPCF. It follows from this that R_{eff} is the extensional collapse of μPCF (see our Remark 5.4), or indeed of any other language that defines the same class of strategies.

We believe that similar results will hold for many other interesting conditions on strategies (and for the corresponding programming languages). For instance, we have recently shown that our functional H (and hence all SR functionals) can be implemented using ML-style references of ground type. Consequently, it would appear that the SR functionals arise as the extensional collapse of a certain class of non-innocent strategies (or of a corresponding programming language with state). Likewise, H can be implemented using ML-style exceptions; we would expect this to lead to another result of the same kind.

Indeed, we would conjecture that more than this is true. It seems likely that, in some sense, *any* reasonable class of strategies that is "sufficiently unconstrained" along either axis (i.e. sufficiently larger than the basic class of PCF strategies) will yield the type structure R as its extensional collapse. This would suggest that a large class of sequential programming languages, including languages with continuations, exceptions and references of various kinds and in various combinations, would have as their extensional collapse the type structure R_{eff}. A precise result to this effect would constitute further evidence for the ubiquity of the SR functionals.

However, we are aware of at least one example of a sequential programming language of the above kind for which such a result does not hold. Our implementations of SR functionals using ML-style exceptions can fail to work if their arguments are allowed to contain *wildcard* exception handlers;[6] thus, the extensional collapse of a certain extension of PCF with exceptions and wildcard handlers does *not* yield the effective SR functionals. However, this need not seriously challenge the kind of general result suggested above, since it appears that wildcard handlers do not fit comfortably into the general framework under consideration. (There is already some consensus that features such as wildcard handlers—a kind of *control delimiter*—are troublesome from a semantic point of view; see e.g. [11]).

Perhaps the most important open question in this area is whether there is (a sensible description of) a class of strategies whose observational quotient is exactly the type structure R. For all the extensional collapse constructions mentioned above, we have to throw away non-extensional junk in order to obtain just the SR functionals. It would be very interesting indeed if one could find a constraint on strategies (preferably of a "local" nature) that gave rise to the SR functionals without any such junk, in the same way that the fully constrained games model

[6] I am indebted to Nick Benton and Andrew Kennedy for this observation.

gives rise to the PCF type structure. Such a constraint on strategies would be likely to suggest a programming language for the SR functionals using some novel kind of language feature, rather than one obtained just by adding a universal functional to PCF.

12.4 A programming language for SR functionals

We end by discussing the possibility of a practical programming language based on the SR functionals.

The existence of universal SR functionals means that, at least in principle, one could design a programming language that incorporates the full power of the effective SR functionals into its purely "functional" fragment. For example, a functional such as H could be built into the language, and programmers could use it and believe that they were doing pure functional programming. If this proved impractical, one could at least build in a selection of weaker functionals, such as the modulus functional M of Section 9.3, giving at least some of the power of the SR functionals.

One can imagine, in principle, several reasons why these possibilities might be interesting. Firstly, they would *increase the power of the purely functional fragment* of the programming language. In existing functional languages such as Standard ML, the pure functional fragment is based essentially on pure PCF. However, there are natural examples of "functional" programs in the SR sense which would be (variously) impossible, inefficient or just inelegant in pure PCF. For such programs, one would gain the transparency and ease of reasoning offered by pure functional programming, as well as increased scope for compiler optimization and perhaps garbage collection.

Secondly, an SR-based functional programming language would allow us to *design better and simpler program logics* than can be easily done for PCF-based languages. We have already pointed out that the class of SR functionals is in many ways mathematically simpler than the class of PCF-sequential functionals; this means that operational notions such as observational equivalence and definability are theoretically more tractable in the SR case. If one adopts the philosophy (advocated e.g. in [34]) that a good program logic should have a clear operational content of a kind easily grasped by programmers, it follows that it should be easier to design and axiomatize a good program logic (with reasonable completeness properties) for an SR-based programming language.

Thirdly, besides the hope that one could *design* a better program logic, it seems likely that one could also *perform proofs about particular programs* more easily within such a logic. In view of the lower "complexity" of the SR functionals, we would expect larger fragments of the logic to be decidable (e.g. equality for finite elements), and this would give greater scope for automation in machine-assisted proofs.

It is worth noting that all of the above points would in principle apply to the parallel language PCF^{++}. In criticism of the above arguments, one might point out that despite the overwhelming mathematical naturalness and simplicity of the type structure $T(\mathrm{PCF}^{++})$, no one has so far been seriously tempted to use PCF^{++} as the basis of a practical programming language. However, we feel there are some grounds for being more optimistic in the case of the SR functionals. Firstly, there is a consensus that parallel operators are "painful to implement and encourage hideously inefficient programming" [12], whereas the implementation of SR functionals is relatively straightforward owing to their sequential nature. (This is attested by the fact that many SR functionals can be implemented very easily in Standard ML.) Secondly, there is the question of the interaction with side-effects (such as exceptions or references) in an impure functional language. In a parallel language with side-effects, we would be obliged to sacrifice the deterministic character of the language (or else to specify a particular interleaving protocol in gruesome detail). In an SR-based language with side-effects, the deterministic character would be easily retained (again, this can be seen by reflecting on the fact that SR functionals can be implemented in ML).

The main weakness in our proposal, at present, is that the universal functional H has an unacceptably high computational complexity (as pointed out in Remark 7.13, it involves a factorial-size search), and we do not currently know of a universal functional that is essentially any better. We might respond to this in two ways: we could hope to discover a better universal functional than H (or perhaps some other way of designing a language for the SR functionals, as suggested at the end of Section 12.3); or else we could sacrifice some expressive power and content ourselves with a selection of simpler functionals (such as M) that were computationally feasible. Regarding the first possibility, we suspect that any universal functional will have at least an exponential complexity in the worst case, but we are hopeful that there may be universal functionals that are feasible *in practice*, since the offending "worst cases" would not arise in natural examples of programs. The second possibility—that of restricting ourselves to a proper subset of the SR functionals—would be aesthetically less satisfying, and we might lose some of the advantages of a clean and simple program logic that still had a clear operational meaning; however, we feel that it might ultimately turn out to be the most practical way forward.

Setting aside the issue of designing completely new programming languages, some of the advantages of an SR-based programming style can already be enjoyed simply by implementing a few useful SR functionals in a language such as Standard ML. This allows one to experiment easily with these functionals, and to discover the kinds of task to which such a programming style is particularly well-suited. We have recently begun an investigation these practical issues. It would appear so far that the construction of general-purpose *search algorithms* and *exact real-number computation* both provide promising application areas for SR-based programming, though there may be many other possible applications, such as the static analysis of programs. It would be fair to admit that our examples to date do not provide clinching proof of the usefulness of SR-based programming, but they are certainly intriguing enough to encourage further investigation.

Some examples of SR functionals implementations in ML, and some tentative programming applications, may be found in the Standard ML source file [33], available electronically from the author's web page.

References

[1] S. Abramsky, K. Honda, and G. McCusker. A fully abstract game semantics for general references. In *Proc. 13th Annual Symposium on Logic in Computer Science*. IEEE, 1998.

[2] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. Accepted for publication, 1996.

[3] H.P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* North-Holland, revised edition, 1984.

[4] G. Berry. Stable models of typed lambda-calculi. In *Proc. 5th International Colloquium on Automata, Languages and Programming, Lecture Notes in Computer Science 62*, pages 72–89. Springer, 1978.

[5] G. Berry and P.-L. Curien. Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20(3):265–321, 1982.

[6] G. Berry, P.-L. Curien, and J.-J. Lévy. Full abstraction for sequential languages: the state of the art. In M. Nivat and J. Reynolds, editors, *Algebraic Semantics*, pages 89–132. Cambridge University Press, 1986.

[7] L. Birkedal, A. Carboni, G. Rosolini, and D.S. Scott. Type theory via exact categories. In *Proc. 13th Annual Symposium on Logic in Computer Science*. IEEE, 1998.

[8] A. Bucciarelli. *Sequential models of PCF: some contributions to the domain-theoretic approach to full abstraction.* PhD thesis, Dipartimento di Informatica, Università di Pisa, 1993.

[9] A. Bucciarelli and T. Ehrhard. Sequentiality and strong stability. In *Proc. 6th Annual Symposium on Logic in Computer Science*, pages 138–145. IEEE, 1991.

[10] A. Bucciarelli and T. Ehrhard. Sequentiality in an extensional framework. *Information and Computation*, 110:265–296, 1994.

[11] R. Cartwright, P.-L. Curien, and M. Felleisen. Fully abstract semantics for observably sequential languages. *Information and Computation*, 111(2):297–401, 1994.

[12] R. Cartwright and M. Felleisen. Observable sequentiality and full abstraction. In *Proc. 19th POPL*, pages 328–342. ACM Press, 1992.

[13] L. Colson and T. Ehrhard. On strong stability and higher-order sequentiality. In *Proc. 9th Annual Symposium on Logic in Computer Science*, pages 103–108. IEEE, 1994.

[14] P.-L. Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming.* Birkhäuser, second edition, 1993.

[15] T. Ehrhard. Hypercoherences: a strongly stable model of linear logic. *Mathematical Structures in Computer Science*, 3:365–385, 1993.

[16] T. Ehrhard. Projecting sequential algorithms on strongly stable functions. *Annals of Pure and Applied Logic*, 77:201–244, 1996.

[17] T. Ehrhard. A relative PCF-definability result for strongly stable functions and some corollaries. To appear in *Information and Computation*, 1997.

[18] J.M.E. Hyland. A small complete category. *Annals of Pure and Applied Logic*, 40:135–165, 1988.

[19] J.M.E. Hyland. First steps in synthetic domain theory. In *Category Theory, Proceedings, Como, Lecture Notes in Mathematics 1488*, pages 131–156. Springer, 1990.

[20] J.M.E. Hyland and C.-H. L. Ong. Modified realizability toposes and strong normalization proofs. In J.F. Groote and M. Bezem, editors, *Typed Lambda Calculi and Applications, Lecture Notes in Computer Science 664*, pages 179–194. Springer, 1993.

[21] J.M.E. Hyland and C.-H. L. Ong. On full abstraction for PCF: I, II and III. Submitted for publication, 1996.

[22] G. Kahn and G.D. Plotkin. Concrete domains. *Theoretical Computer Science*, pages 187–277, 1993. First appeared in French as INRIA-LABORIA technical report, 1978.

[23] R. Kanneganti, R. Cartwright, and M. Felleisen. SPCF: its model, calculus, and computational power. In *Proc. REX Workshop on Semantics and Concurrency, Lecture Notes in Computer Science 666*, pages 318–347. Springer, 1993.

[24] S.C. Kleene. Recursive functionals and quantifiers of finite types revisited I. In J.E. Fenstad, R.O. Gandy, and G.E. Sacks, editors, *Generalized Recursion Theory II*, pages 185–222, 1978.

[25] S.C. Kleene and R.E. Vesley. *The Foundations of Intuitionistic Mathematics.* North-Holland, 1965.

[26] G. Kreisel. Interpretation of analysis by means of functionals of finite type. In A. Heyting, editor, *Constructivity in Mathematics*, pages 101–128. North-Holland, 1959.

[27] J. Laird. Full abstraction for functional languages with control. In *Proc. 12th Annual Symposium on Logic in Computer Science*, pages 58–67. IEEE, 1997.

[28] J. Laird. *A Semantic Analysis of Control.* PhD thesis, University of Edinburgh, 1998. Submitted for examination.

[29] R. Loader. Finitary PCF is not decidable. To appear, 1996.

[30] J.R. Longley. Notions of computability at higher types. In preparation.

[31] J.R. Longley. *Realizability Toposes and Language Semantics.* PhD thesis, University of Edinburgh, 1995. Available as ECS-LFCS-95-332.

[32] J.R. Longley. Realizability models for sequential computation. In preparation, 1998.

[33] J.R. Longley. When is a functional program not a functional program?: a walkthrough introduction to the sequentially realizable functionals. Standard ML source file, available from `http://www.dcs.ed.ac.uk/home/jrl/`, 1998.

[34] J.R. Longley and G.D. Plotkin. Logical full abstraction and PCF. In J. Ginzburg et al., editor, *Tbilisi Symposium on Language, Logic and Computation*, pages 333–352. SiLLI/CSLI, 1997.

[35] J.R. Longley and A.K. Simpson. A uniform approach to domain theory in realizability models. *Mathematical Structures in Computer Science*, 7:469–505, 1997.

[36] S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic.* Springer, 1992.

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

[38] R. Milner, M. Tofte, and R. Harper. *The Definition of Standard ML.* MIT Press, 1990.

[39] P.S. Mulry. Generalized Banach-Mazur functionals in the topos of recursive sets. *Journal of Pure and Applied Algebra*, 26:71–83, 1982.

[40] H. Nickau. Hereditarily sequential functionals. In *Proc. 3rd Symposium on Logical Foundations of Computer Science, Lecture Notes in Computer Science 813*, pages 253–264. Springer, 1994.

[41] P.W. O'Hearn and J.G. Riecke. Kripke logical relations and PCF. *Information and Computation*, 120(1):107–116, 1995.

[42] C.-H.L. Ong and C.A. Stewart. A Curry-Howard foundation for functional computation with control. In *Proc. Symposium on Principles of Programming Languages*, pages 215–227. ACM Press, 1997.

[43] J. van Oosten. A combinatory algebra for sequential functionals of finite type. Technical Report 996, University of Utrecht, 1997. To appear in Proc. Logic Colloquium, Leeds.

[44] J. van Oosten. The modified realizability topos. *Journal of Pure and Applied Algebra*, 116:273–289, 1997.

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

[46] G.D. Plotkin. Full abstraction, totality and PCF. Accepted for publication, 1997.

[47] B. Reus. *Program Verification in Synthetic Domain Theory.* PhD thesis, University of Munich, 1995.

[48] B. Reus and T. Streicher. General synthetic domain theory—a logical approach. In *Category Theory in Computer Science, Lecture Notes in Computer Science 1290*, pages 293–313. Springer, 1997.

[49] V.Yu. Sazonov. Degrees of parallelism in computations. In *Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 45*, pages 517–523. Springer, 1976.

[50] D.S. Scott. Data types as lattices. *SIAM Journal of Computing*, 5(3):522–587, 1976.

[51] D.S. Scott. A type-theoretical alternative to ISWIM, CUCH, OWHY. *Theoretical Computer Science*, 121:411–440, 1993. First written in 1969 and widely circulated in unpublished form since then.

[52] A.K. Simpson. Computational adequacy in an elementary topos. Presented at CSL '98; submitted for publication, 1998.

[53] G.L. Steele and G.J. Sussman. The revised report on Scheme, a dialect of Lisp. Technical Report Memo 452, MIT AI Lab, 1978.

[54] T. Streicher. Investigations into intensional type theory. Habilitationsschrift, München, 1993.

[55] P. Taylor. The fixed point property in synthetic domain theory. In *Proc. 6th Annual Symposium on Logic in Computer Science*, pages 152–160. IEEE, 1991.

[56] M.B. Trakhtenbrot. On representation of sequential and parallel functions. In *Proc. 4th Symposium on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 32*, pages 411–417. Springer, 1975.

[57] J. Vuillemin. *Syntaxe, Sémantique et Axiomatique d'un Langage de Programmation Simple.* PhD thesis, Université Paris VII, 1974.