Computability 5 (2016) 127–146
DOI 10.3233/COM-150048
IOS Press

Effective operations of type 2 in PCAs

Eric Faber
DPMMS, Centre for Mathematical Sciences, University of Cambridge, Wilberforce Road, Cambridge CB3 0WB, UK
eef25@cam.ac.uk

Jaap van Oosten
Department of Mathematics, Utrecht University, P.O. Box 80.010, 3508TA Utrecht, The Netherlands
J.vanOosten@uu.nl
www.staff.science.uu.nl/~ooste110/

Abstract. We exhibit a way of "forcing a partial functional to be realizable as effective operation" for arbitrary partial combinatory algebras (pcas). This gives a method of defining new pcas from old ones for some fixed type 2 (partial) functional, where the new partial functions can be viewed as computable relative to that functional. It is shown that this generalizes a notion of computation relative to a functional as defined by Kleene for the natural numbers. The resulting notion of computation can be characterized by a universal property both in the category of pcas and in the lattice of local operators on the corresponding realizability topos.

Our result is useful in two ways. For one thing, we expect that the emphasis on forcing the applicative structure to satisfy certain properties yields new methods to tackle problems in complexity theory, for example in connection to higher computability. Second, the methods are defined for abstract notions of computation, which provides a connection with realizability toposes and categories of assemblies. We will see an instance of this in the effective topos using an example of a functional that forces every arithmetical set to be decidable and a local operator defined by A. Pitts in his thesis [The theory of triposes, PhD thesis, Cambridge University, 1981].

As an intermezzo we also prove the convenient result that the two definitions of a pca that are common in the literature are essentially the same.

Keywords: Partial combinatory algebra, computability theory, Turing degree, effective operation, computable functional, realizability topos, realizability

1. Introduction

Effective operations have originally been studied in computability theory, with as famous results the theorems of Myhill–Shepherdson and Rice–Shapiro. They have also been studied in relation to computable functionals (e.g. see [3]), although the two notions are not quite the same. For example, it is known that not every functional that can be realized as effective operation is also a computable functional ([3]). In realizability, partial combinatory algebras (pcas) are used as models for abstract Turing machines. The notion of effective operation can be extended to pcas in a straightforward way. In this paper, we discuss effective operations on arbitrary partial combinatory algebras, and present a way to force a functional to be realizable as effective operation: for a (partial) functional $F : A^A \to A$, where A is a pca, we construct a pca $A[F]$ for which F, restricted to the total functions on $A[F]$, is realizable as an effective operation. In Theorem 3.1, we characterize $A[F]$ by a universal property in the category of pcas.

In [8], Kleene introduced the notion of a *computable functional*, and with it the notion of computability relative to a functional. It turns out that for $A = \mathscr{K}_1$, Kleene's first model, the functions computable in $\mathscr{K}_1[F]$ are precisely the functions computable relative to F in Kleene's sense. This yields an alternative approach to computation in a type 2 functional that generalizes to arbitrary pcas.

The recursive application defined by $A[F]$ can also be characterized in the lattice of local operators on $\mathrm{RT}(A)$, the realizability topos on A. This is shown in Theorem 6.1. The theory of recursive hierarchies of functionals (e.g. see [4]) can therefore be applied to study subtoposes of realizability toposes, especially their computational complexity.

As an example, we look at a specific local operator J due to A. Pitts that is studied in [15]. It turns out that the functions "computable in J" are precisely the functions computable relative to a certain functional.

On the way we also prove that a strengthening of our definition of a pca, that we call strict pcas, is (up to applicative isomorphism) not a proper strengthening at all. This is posed as an open question in [2]. The result is

convenient, since in the literature there has not been wide consensus on the definition of pca; several authors have been using strict pcas by default. We will catch a glimpse of the usefulness of this result by exhibiting a pca that at first sight seems a long way from being strict (Proposition 6.3).

In the last section, we discuss the generalization of our results to higher types. Although Kleene's definitions of recursive functionals generalize to higher types, the connection with effective operations gets lost. A generalization of our results on effective operations seems to demand for a more complicated method of "forcing" than what we have described for a type two functional. Further investigations of this kind could thus be useful in computability theory and the study of realizability toposes.

2. Preliminaries

We start by reviewing the definition of a partial combinatory algebra and some basic constructions.

Definition 2.1. A partial applicative structure on a set A is a partial map $A \times A \to A$ that we denote by

$$(a, b) \mapsto ab.$$

We write $ab \downarrow$ to say that ab is defined, $ab = c$ to say that ab is defined and has value c. In writing compositions of the application, we adopt the convention of *associating to the left*, so:

$$abcd = \big(\big((ab)c\big)d\big).$$

For $V = \{x_0, x_1, \ldots\}$ an enumerable set of variables, we define the set of *terms* by:

Definition 2.2. For all $a \in A$, $x \in V$

(1) a is a term;
(2) x is a term;
(3) For t, s terms, (ts) is a term.

A term t without free variables is called *closed* and might *denote* or not. This is defined inductively:

(1) a denotes and has value a;
(2) If t denotes and has value a, and s denotes and has value b, and $ab = c$, then (ts) denotes and has value c.

We write $t \downarrow$ if t denotes, and $t = a$ if t denotes and has value a. We write $t \lesssim s$ if $s \downarrow$ implies that $t \downarrow$, and in that case $s = t$. We write $t \simeq s$ if $t \lesssim s$ and $s \lesssim t$. As a warning, note that other authors sometimes define the relation "$\lesssim$" the other way around (this is comparable to the two conventions used to order forcing conditions in set theory). Here, we stick to the convention used in [14].

We denote a term t with free variables $x_1, \ldots, x_n$ by

$$t(x_1, \ldots, x_n).$$

For $a_1, \ldots, a_n \in A$, we can substitute $x_1, \ldots, x_n$ by $a_1, \ldots, a_n$, resulting in a closed term $t(a_1, \ldots, a_n)$.

Definition 2.3. A partial combinatory algebra is a partial applicative structure A, with elements $\mathsf{k}, \mathsf{s} \in A$ such that for all $a, b \in A$:

(1) $\mathsf{s}ab \downarrow$;
(2) $\mathsf{k}ab = a$;
(3) $\mathsf{s}abc \lesssim ac(bc)$.

We also say that the applicative structure on A is *combinatory complete*.

Example 2.1. There are several examples of pcas, [14] treats a lot of them. Here we just list a few with little explanation:

- The trivial pca, $A = \{*\}$ with $** = *$.
- *Kleene's first model* $\mathscr{K}_1$, with underlying set $\mathbb{N}$. Application is given by:

$$nm \simeq \phi_n(m),$$

where ϕ_n is the nth partial function for some model of computation and coding.

An introduction to partial combinatory algebras and its basic properties can be found in Chapter 1 of [14]. Here we will just quickly present some tools that we need later on, the proofs can all be found in [14]. For a pca A, an element $a \in A$ defines a partial function $A \to A$ by:

$$b \mapsto ab.$$

We say that a is an *index* for this partial function. By abuse of language, we sometimes say that a is a partial (computable) function.

An important result for pcas is the *recursion theorem*:

Theorem 2.1 (Recursion theorem)**.** *For every $f \in A$, there exists e such that for all $a \in A$:*

$$ea \lesssim fea.$$

Also important is the fact that a pca is *combinatory complete*. This means that for every term $t(x_1, \ldots, x_n)$ with free variables $x_1, \ldots, x_n$, there is an element $\langle x_1 \cdots x_n \rangle t \in A$ such that for all $a_1, \ldots, a_n \in A$:

$$\big(\langle x_1 \cdots x_n \rangle t\big) a_1 \cdots a_{n-1} \downarrow, \tag{1}$$

$$\big(\langle x_1 \cdots x_n \rangle t\big) a_1 \cdots a_{n-1} a_n \lesssim t(a_1, \ldots, a_n). \tag{2}$$

The notation $\langle x_1 \cdots x_n \rangle t$ should remind the reader of λ-abstraction in λ-calculus. We use a different notation because it is slightly different from λ-abstraction, but it is applied in the same way.

Besides k and s, we identify (a choice of) other canonical elements of A, among which:

- Boolean combinators T and F that satisfy for all $a, b \in A$:

$$\mathsf{T}ab = a, \tag{3}$$

$$\mathsf{F}ab = b. \tag{4}$$

  Indeed, one can take $\mathsf{T} = \mathsf{k}$.
- Using combinatory completeness and the Boolean combinators, we can define if/else statements: for t, s closed terms:

$$\big(\langle v \rangle v\big(\langle x \rangle t\big)\big(\langle x \rangle s\big)\mathsf{k}\big)b \lesssim \begin{cases} t & \text{if } b = \mathsf{T}, \\ s & \text{if } b = \mathsf{F}, \\ \text{unspecified} & \text{otherwise.} \end{cases}$$

  For legibility, we denote a term like $v(\langle x \rangle t)(\langle x \rangle s)\mathsf{k}$ by

$$\mathsf{if}\ v\ \mathsf{then}\ t\ \mathsf{else}\ s.$$

- The pairing combinator p together with projections p_0, p_1:

$$\mathsf{p}_0(\mathsf{p}ab) = a, \tag{5}$$

$$\mathsf{p}_1(\mathsf{p}ab) = b. \tag{6}$$

- The Curry numerals: each pca contains representables of the natural numbers, i.e. for any $n \in \mathbb{N}$ an element $\overline{n} \in A$. It is a basic result (for a proof, see [14]) that, for every k, there is for every k-ary partial recursive function $F : \mathbb{N}^k \to \mathbb{N}$ an index $f \in A$ such that for all $(n_1, \ldots, n_k) \in \operatorname{dom} F$:

$$f\overline{n_1} \cdots \overline{n_k} = \overline{F(n_1, \ldots, n_k)}.$$

When there is chance of confusion, we decorate (choices of) canonical elements by a subscript to denote the pca to which it belongs, e.g. k_A, s_A, T_A, etc.

We can also code tuples of elements in a computable way. That is, there is an index $t \in A$ such that for all $n \in \mathbb{N}$, $u_0, \ldots, u_n \in A$ we can code a tuple

$$[u_0, \ldots, u_n] := t\overline{n+1}u_0 \cdots u_n \in A$$

such that projection, concatenation of tuples, and a length function (that yields the length of a tuple as a curry numeral) are all computable. We denote concatenation by $*$:

$$[u_0, \ldots, u_i] * [u_{i+1}, \ldots, u_n] = [u_0, \ldots, u_n]$$

and for $u = [u_0, \ldots, u_n]$, we write $u^{<i+1}$ for $[u_0, \ldots, u_i]$. These basic operations can all be done computably and uniformly in the variables.

There is a notion of morphism between partial combinatory algebras due to John Longley (see [9]), called *applicative morphism*. In the following definition, we adopt the following notation: for a total relation $R : X \to Y$ between sets X and Y, we denote for every $x \in X$ by $R(x)$ the set:

$$\{y \in Y \mid xRy\}.$$

For A a pca, $a \in A$ and $P \subseteq A$ a subset, we write $aP\downarrow$ if for all $b \in P$, $ab\downarrow$, and in that case

$$aP := \{ab \mid b \in P\}.$$

For $P, Q \subseteq A$, we write $PQ\downarrow$ if for all $b \in P$, $b' \in Q$, $bb'\downarrow$, and in that case

$$PQ = \left\{bb' \;\middle|\; b \in P, b' \in Q\right\}.$$

Whenever we write compositions of these applications of an element and a subset, or a subset and a subset, we again associate to the left.

Definition 2.4 (Longley)**.** Let A, B be pcas. An applicative morphism $\gamma : A \to B$ is a total relation from A to B, together with an element $r \in B$ such that whenever $a, a' \in A$ and $aa'\downarrow$, we have:

$$r\gamma(a)\gamma\left(a'\right)\downarrow \quad \text{and} \quad r\gamma(a)\gamma\left(a'\right) \subseteq \gamma\left(aa'\right).$$

The element r is said to *realize* γ.

An applicative morphism γ is *decidable* if there is $d \in B$ such that

$$d\gamma(\mathsf{T}_A) = \{\mathsf{T}_B\}, \tag{7}$$

$$d\gamma(\mathsf{F}_A) = \{\mathsf{F}_B\}. \tag{8}$$

We call d a *decider* for γ.

The composition of two applicative morphisms is defined by composition of the relations, and one can verify that this is again an applicative morphism. There is a pre-order on applicative morphisms defined as follows: for $\gamma : A \to B, \delta : A \to B$, write $\gamma \preceq \delta$ when there is $t \in B$ such that for all $a \in A$:

$$t\gamma(a) \subseteq \delta(a).$$

We write $\gamma \simeq \delta$ if $\gamma \preceq \delta$ and $\delta \preceq \gamma$, in this case γ and δ are *equivalent*. One can show that this pre-order is preserved by composition on both sides. We thus obtain a pre-order enriched category PCA consisting of partial combinatory algebras as objects, and applicative morphisms as arrows. The identity arrow for a pca A is denoted by ι_A and given by the identity relation on A.

Note 2.1. As remarked in the Introduction, Definition 2.3 is weaker than the definition that is used in some other literature (including [14]). However, it will be a consequence of Theorem 5.1 below that the corresponding categories of pcas, with morphisms as above, are equivalent.

For a pca A, we define a set PR_A by:

$$\mathrm{PR}_A = \big\{f : A \to A \text{ partial} \mid (\exists a \in A)(\forall b \in A)ab \lesssim f(b)\big\}.$$

We also define a function $I_1^A : \mathrm{PR}_A \to \mathscr{P}(A)$ (here $\mathscr{P}(A)$ is the powerset of A) by:

$$I_1^A(f) = \big\{a \in A \mid (\forall b \in A)ab \lesssim f(b)\big\}.$$

For $\gamma : A \to B$ an applicative morphism, we define a set PR_γ by:

$$\mathrm{PR}_\gamma : \big\{f : A \to A \text{ partial} \mid (\exists b \in B)(\forall a \in A) f(a) \downarrow \Rightarrow b\gamma(a) \subseteq \gamma\big(f(a)\big)\big\}. \tag{9}$$

Lastly, we also define a function $I_1^\gamma : \mathrm{PR}_\gamma \to \mathscr{P}(B)$ by:

$$I_1^\gamma(f) = \big\{b \in B \mid (\forall a \in A) f(a) \downarrow \Rightarrow b\gamma(a) \subseteq \gamma\big(f(a)\big)\big\}.$$

Observe that $\mathrm{PR}_A = \mathrm{PR}_{\iota_A}$.

The following result by the second author (see [13] or [14]) tells us that we can adjoin any partial function $f : A \to A$ to a pca A, to obtain a pca $A[f]$ in which f has an index. This generalizes the notion of a Turing machine with oracle from ordinary computability theory to pcas.

Theorem 2.2 ([13])**.** *For A a pca and $f : A \to A$ a partial function, there exists a pca $A[f]$ defined on the set A such that the identity on A is a decidable applicative morphism $\iota_f : A \to A[f]$, and*:

(1) $f \in \mathrm{PR}_{A[f]} = \mathrm{PR}_{\iota_f}$.
(2) *Whenever $\gamma : A \to B$ is a decidable applicative morphism such that $f \in \mathrm{PR}_\gamma$, there exists a decidable $\gamma_f : A[f] \to B$ such that $\gamma_f \circ \iota_f = \gamma$. Moreover, if $\delta : A[f] \to B$ is a decidable applicative morphism such that $\delta \circ \iota_f \simeq \gamma$, then $\delta \simeq \gamma_f$.*

The proof of the above theorem is done by defining an applicative structure on A using *dialogues*:

Definition 2.5. Let $f : A \to A$ be a partial function, and $a, b \in A$. An *f-dialogue* between a and b is an element $u = [u_0, \ldots, u_n] \in A$ such that for all $i \leqslant n$:

$$a\big([b] * u^{<i}\big) = \mathsf{p}\mathsf{F}v_i,$$

where $f(v_i)$ is defined and $f(v_i) = u_i$.

An f-dialogue u between a and b *halts* if there exists $c \in A$ such that

$$a\big([b] * u\big) = \mathsf{p}\mathsf{T}c.$$

We write $a \cdot^f b \downarrow$ if there is a (necessarily unique) halting f-dialogue u between a and b, and $a \cdot^f b = c$ if $a \cdot^f b \downarrow$ and

$$a\big([b] * u\big) = \mathsf{p}\mathsf{T}c,$$

where u is a halting f-dialogue between a and b.

The applicative structure on $A[f]$ is then given by $(a, b) \mapsto a \cdot^f b$. For a full proof of combinatory completeness, see [14] or [13]. Note that these sources use the strict version of a pca (Definition 5.1), but the proofs work without any modification (alternatively, one can apply Theorem 5.1). The following corollary corresponds to Corollary 1.3 in [13]:

Corollary 2.1. *Let A be a pca.*

(1) *If $f \in \mathrm{PR}_A$, then $A[f]$ and A are isomorphic pcas.*
(2) *If $f, g : A \to A$ are partial functions, then $A[f][g]$ and $A[g][f]$ are isomorphic.*
(3) *If $\mathscr{K}_1^f$ is the pca of partial recursive application with an oracle for f, then $\mathscr{K}_1^f$ is isomorphic to $\mathscr{K}_1[f]$.*

3. Effective operations

We will now define the set of effective operations on pcas. For any pca A, let Tot_A be the subset of PR_A consisting of all total functions:

$$\mathrm{Tot}_A = \{f \in \mathrm{PR}_A \mid f \text{ is total}\}.$$

Similarly, define for any γ the subset $\mathrm{Tot}_\gamma \subseteq \mathrm{PR}_\gamma$ consisting of all total functions in PR_γ.

Definition 3.1 (Effective operation)**.** For a pca A, we define the set E_A of *effective operations* in A by:

$$E_A = \big\{F : D \to A \mid D \subseteq \mathrm{Tot}_A \text{ and } (\exists a \in A)(\forall f \in \mathrm{Tot}_A) \tag{10}$$

$$f \in D \Longrightarrow \big(aI_1(f) \downarrow \text{ and } aI_1^A(f) = \big\{F(f)\big\}\big)\big\}. \tag{11}$$

Define a function $I_2^A : E_A \to \mathscr{A}$ by:

$$I_2^A(F) = \big\{a \mid (\forall f \in \mathrm{dom}\, F)\big(\forall b \in I_1^A(f)\big)\big(ab = F(f)\big)\big\}.$$

Similarly, for $\gamma : A \to B$ an applicative morphism we define the set E_γ of *effective operations relative to γ* by:

$$E_\gamma = \big\{F : D \to A \mid D \subseteq \mathrm{Tot}_\gamma \text{ and } (\exists b \in B)(\forall f \in \mathrm{Tot}_\gamma) \tag{12}$$

$$f \in D \Longrightarrow \big(bI_1^\gamma(f) \downarrow \text{ and } bI_1^\gamma(f) \subseteq \gamma\big(F(f)\big)\big)\big\}. \tag{13}$$

Example 3.1. As an example, we look at the following functional $E : A^A \to A$, where A is an arbitrary pca,

$$E(\alpha) = \begin{cases} \mathsf{T} & \text{if } (\exists a)\alpha(a) = \mathsf{T}, \\ \mathsf{F} & \text{otherwise.} \end{cases}$$

For $A = \mathscr{K}_1$, it is easily seen that the above (restricted to the total computable functions) is not an effective operation (otherwise the halting problem is decidable). However, the construction below shows that there are a lot of pcas in which E, when restricted, is an effective operation.

The statement of the following theorem is very similar to Theorem 2.2.

Theorem 3.1. *Let A be a pca, and $F : A^A \to A$ a (partial) functional. Then there exists a pca $A[F]$ defined on the set A such that the identity relation on A is a decidable applicative morphism $\iota_F : A \to A[F]$ and:*

(1) $F \restriction \mathrm{Tot}_{A[F]} \in E_{A[F]} = E_{\iota_F}$;
(2) *Whenever $\gamma : A \to B$ is a decidable applicative morphism such that $F \restriction \mathrm{Tot}_\gamma \in E_\gamma$, there exists a unique (up to equivalence) decidable applicative morphism $\gamma_F : A[F] \to B$ such that $\gamma_F \circ \iota_F = \gamma$. Moreover, if $\delta : A[F] \to B$ is a decidable applicative morphism such that $\delta \circ \iota_F \simeq \gamma$, then $\delta \simeq \gamma_F$.*

Remark 3.1. Before embarking on the proof, we would like to clarify an issue that might at this point have confused the reader. It may seem that we are treating a partial functional $F : A^A \to A$ and an effective operation on the same level (up to restriction), which is why we say that our construction "forces F to be realizable as effective operation". However, since for any pca A, $|\mathrm{Tot}_A| = |A| < |A^A|$, F may be defined on a lot more functions than $A[F]$ could ever contain. So F will always contain more information than the effective operation we construct from it.

However, $\mathrm{Tot}_{A[F]}$ will usually contain functions that were not in Tot_A, so we cannot restrict F beforehand. Indeed, $A[F \restriction \mathrm{Tot}_A]$ is quite different from $A[F]$! Take for example the functional E above. In this case, $\mathscr{K}_1[E \restriction \mathrm{Tot}_{\mathscr{K}_1}] = \mathscr{K}_1[\emptyset']$ (here $\emptyset'$ is the first *jump* of the empty set), whereas we will see (Corollary 4.1) that $\mathscr{K}_1[E]$ computes all hyperarithmetical functions.

It may be better to formulate the above distinction in categorical terms (see Section 6.1). We could define an *effective operation* as a partial arrow $\mathbf{A}^{\mathbf{A}} \leftarrow D \to \mathbf{A}$ in a category of assemblies on A. The above theorem then constructs a pca $A[F]$ for which the category of assemblies is a reflective subcategory of the assemblies on A, in which the partial functional $F : A^A \to A$ is *realizable* as partial arrow $\mathbf{A[F]}^{\mathbf{A[F]}} \leftarrow D \to \mathbf{A[F]}$. This corresponds to (i) in the above theorem. In fact, this reflective subcategory is induced by a local operator on the realizability topos $\mathrm{RT}(A)$, and this local operator turns out to be the smallest such that F is realizable. This is the content of Theorem 6.1 and corresponds to (ii) of the above theorem.

Proof. We will construct $A[F]$ as $A[f]$ for

$$f = \bigcup_\alpha f_\alpha,$$

where $\{f_\alpha\}_\alpha$ is a compatible family of partial functions indexed by the ordinals. We define this family by transfinite induction. For all $a, b \in A$, we let:

- $f_0 = \emptyset$;
- $f_{\alpha+1}(a) = b$ whenever there exists $g \in \mathrm{Tot}_{A[f_\alpha]}$ such that $a \in I_1^{A[f_\alpha]}(g)$ and $F(g) = b$;
- For $\lambda > 0$ a limit ordinal, $f_\lambda = \bigcup_{\alpha<\lambda} f_\alpha$.

It is not hard to see that this indeed defines a compatible family of partial functions. So $f = \bigcup_\alpha f_\alpha$ is a partial function, and we define $A[F] := A[f]$. We denote the application in $A[F]$ by $\cdot^F$.

For part (i), suppose $g \in \mathrm{Tot}_{A[F]}$, $a \in I_1^{A[F]}(g)$, and $F(g)$ is defined. Then it is not hard to see that there is α such that $g \in \mathrm{Tot}_{A[f_\alpha]}$, $a \in I_1^{A[f_\alpha]}(g)$. Then $f_{\alpha+1}(a) = F(g)$, hence $f(a) = F(g)$. Indeed, F is an effective operation in $A[F]$.

For part (ii), suppose $\gamma : A \to B$ is a decidable applicative morphism such that $F \in E_\gamma$. Let $r_F \in I_2^\gamma(F)$. Denote the realizer for γ by r. Let d be the decider for γ. Let c be such that for all $x \in \gamma(a)$, $v \in \gamma(u)$, $\mathsf{c}xy \in \gamma([a] * u)$. Let c' be such that for all $y \in \gamma(a)$, $x \in \gamma(u)$, $\mathsf{c}'xy \in \gamma(u * [a])$. Let q_0, q_1 be such that for all $x \in \gamma(a)$, $\mathsf{q}_0 x \in \gamma(\mathsf{p}_0 a)$, $\mathsf{q}_1 x \in \gamma(\mathsf{p}_1 a)$. By the recursion theorem, let $e \in B$ be such that

$$exv \lesssim \mathsf{if}\ \mathsf{dq}_0\big(re(\mathsf{c}xv)\big)\ \mathsf{then}\ \mathsf{q}_1\big(re(\mathsf{c}xv)\big) \tag{14}$$

$$\mathsf{else}\ ex\big(\mathsf{c}'v\big(r_F\big(r\big(\mathsf{q}_1\big(re(\mathsf{c}xv)\big)\big)\big)\big)\big). \tag{15}$$

Then one can verify that for $v \in \gamma([])$, we have for all $e \in \gamma(a)$, $x \in \gamma(a')$ such that $a \cdot^f a' \downarrow$:

$$exv \in \gamma\big(a \cdot^f a'\big).$$

Essential is that if $a \in I_1^{A[F]}(f)$, then for $b \in \gamma(a)$, $rb \in I_1^\gamma(f)$.

So by the above, $\langle xy\rangle xyv$ realizes γ as an applicative morphism $\gamma_F : A[F] \to B$, and the fact that $\gamma = \gamma_F \circ \iota_F$ is obvious.

Uniqueness is easy to check. □

Remark 3.2. The construction of $A[F]$ can be thought of as a construction in stages: we say that $a \cdot^F b \downarrow$ *at stage* α if $a \cdot^{f_\alpha} b \downarrow$. In proofs of statements about $A[F]$, we will often use induction on the stage at which a term denotes. The infinitary nature of computation relative in a functional, as remarked by Kleene in [8], is reflected by this construction in (transfinitely many) stages.

Example 3.2. We again take a look at the functional E from Example 3.1. For any pca A, E is an effective operation in $A[E]$. As a consequence, the computable predicates in $A[E]$ are closed under existential quantification: suppose $p \in A[E]$ is a total function in 2 variables, taking values in $\{\mathsf{T}, \mathsf{F}\}$. Then:

$$Q(n) := (\exists m)\, pnm$$

is clearly also a recursive predicate in $A[E]$, so it has some index $q \in A[E]$.

For $A = \mathscr{K}_1$, this produces a familiar (see [8]) result. Namely, it follows (by taking complements) that every arithmetical subset of $\mathbb{N}$ is computable in $\mathscr{K}_1[E]$. We will see below that the total functions in $\mathscr{K}_1[E]$ are in fact precisely the hyperarithmetical functions.

Remark 3.3. One may want to define, for functionals $F, G : A^A \to A$, a pca $A[F, G]$ in which F, G are both effective operations. It will not work to define this as $A[F][G]$. This follows the same reasoning as in Remark 3.1. In $A[F][G]$, only $F \restriction \mathrm{Tot}_{A[F]}$ is realizable, whereas we want $F \restriction \mathrm{Tot}_{A[F][G]}$ to be realizable. However, if one codes any tuple of functionals into one functional H (e.g. $H(g) = \mathsf{p}F(g)G(g)$), then $A[H]$ will satisfy the required properties and the analogue of Theorem 2.2 for such a tuple holds.

4. Effective operations and Kleene computability

In [8], Kleene developed a recursion theory for functionals, which included a notion of computability relative to a functional. A short overview of this theory, as well as comparisons to other notions of higher-type computation can be found in [10].

In this section we will show that a total function $f : \mathbb{N} \to \mathbb{N}$ is computable relative to a functional $F : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ (in Kleene's sense) if and only if it has an index in $\mathscr{K}_1[F]$. This result will be almost immediate, since (as we will

see) Kleene's indexed set of functions computable in F is basically an explicit model of $\mathscr{K}_1[F]$. Our construction has the advantage of being independent of any explicit model of $\mathscr{K}_1$ and avoids a lot of technicalities. Moreover, there is the immediate generalization to arbitrary pcas and the connection to realizability.

In his paper, Kleene defined an $\mathbb{N}$-indexed set of functions that may take tuples of arguments of all finite types. The definition consists of nine inductive rules, named (S1)–(S9). A function with index e is denoted by $\{e\}$. A function $g : \mathbb{N} \to \mathbb{N}$ is then computable *relative to* the functional $F : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ if there is an index $e \in \mathbb{N}$ such that for all $n \in \mathbb{N}$:

$$\{e\}(n, F) = g(n).$$

Instead of defining Kleene's (S1)–(S9) in full generality, we will just state eight derived rules that define the indexed set of functions computable relative to some functional F according to Kleene. The rule (S7) is missing since that applies specifically to type 1 arguments, and we only consider a specific type 2 oracle.

Definition 4.1. Let $F : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ be a functional. Also fix some primitive recursive coding of tuples $\langle - \rangle : \mathbb{N}^{<\omega} \to \mathbb{N}$. We define an indexed set of partial functions $\mathbb{N}^{<\omega} \to \mathbb{N}$ by the following rules: for all $n, m \in \mathbb{N}$:

(S1) Successor function: $\{\langle 1, 1\rangle\}(n) = n + 1$.
(S2) Constant function: $\{\langle 2, 1, m\rangle\}(n) = m$.
(S3) Projection: for all $n_1, \ldots, n_m \in \mathbb{N}$, $\{\langle 3, m\rangle\}(n_1, \ldots, n_m) = n_1$.
(S4) Composition: For $g, h \in \mathbb{N}$, all $k, n_1, \ldots, n_k \in \mathbb{N}$:

$$\big\{\langle 4, k, g, h\rangle\big\}(n_1, \ldots, n_k) \simeq \{g\}\big(\{h\}(n_1, \ldots, n_k), n_1, \ldots, n_k\big).$$

(S5) Primitive recursion: For any $g, h \in \mathbb{N}$, if $m = \langle 5, 2, g, h\rangle$ then for all k:

$$\{m\}(0, n) \simeq \{g\}(n), \tag{16}$$

$$\{m\}(k + 1, n) \simeq \{h\}\big(k, \{m\}(k, n), n\big). \tag{17}$$

(S6) Permutation of arguments: For any $g \in \mathbb{N}$, $1 \leqslant k < r$ and $n_1, \ldots, n_r \in \mathbb{N}$:

$$\big\{\langle 6, r, k, g\rangle\big\}(n_1, \ldots, n_r) \simeq \{g\}(n_{k+1}, n_1, \ldots, n_k, n_{k+2}, \ldots, n_r).$$

(S8) Application of F: For every $h, e \in \mathbb{N}$:

$$\big\{\langle 8, 1, h\rangle\big\}(n) \simeq F\big(\lambda k.\{h\}(k, n)\big).$$

(S9) Index vocation: for all $m, k, l, n_1, \ldots, n_{k+l} \in \mathbb{N}$

$$\big\{\langle 9, k, l\rangle\big\}(m, n_1, \ldots, n_{k+l}) \simeq \{m\}(n_1, \ldots, n_k).$$

Above, $a \simeq b$ has the same meaning as before: the right-hand side is defined if and only if the left-hand side is defined, and when this is the case both sides are equal.

In what follows, we assume that we have fixed some functional $F : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$, and defined the partial map $\{-\}(-) : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ according to the above.

Theorem 4.1. *The partial map* $\{-\}(-) : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *defines a partial combinatory algebra* $\mathscr{K}_1^F$.

Proof. Tedious exercise using the clauses in Definition 4.1. □

The following can be viewed as the type 2 analogue of Corollary 2.1(3).

Theorem 4.2. *There is an isomorphism of pcas* $\mathscr{K}_1[F] \cong \mathscr{K}_1{}^F$.

Proof. By Church's thesis, the identity relation on $\mathbb{N}$ is an applicative morphism $\gamma : \mathscr{K}_1 \to \mathscr{K}_1{}^F$. It is easy to see that $F \in E_\gamma$. By Theorem 3.1, the identity relation on $\mathbb{N}$ is an applicative morphism $\gamma_F : \mathscr{K}_1[F] \to \mathscr{K}_1{}^F$.

It is not hard to show that we can define the application in $\mathscr{K}_1{}^F$ computably in $\mathscr{K}_1[F]$ using the clauses of Definition 4.1 and the recursion theorem when needed. So the identity relation on $\mathbb{N}$ is also an applicative morphism $\mathscr{K}_1{}^F \to \mathscr{K}_1[F]$. Therefore $\mathscr{K}_1[F] \cong \mathscr{K}_1{}^F$. □

Corollary 4.1. *Let E be as in Example 3.1 for $A = \mathscr{K}_1$. The total functions in $\mathscr{K}_1[E]$ are precisely the hyperarithmetical functions.*

Proof. This follows from Theorem 4.2 and the corresponding fact for $\mathscr{K}_1{}^E$, which is Theorem XLVIII in [8]. □

The above corollary is what we can deduce from Kleene's original paper on recursive functionals. However, we can do a little better using the results of Hinman's approach in [4]. There, the domains of the $\mathbb{N}$-indexed partial functions $\mathbb{N} \to \mathbb{N}$ "computable in E" are precisely the Π^1_1-sets. Here we can prove something similar. In the context of pcas and applicative morphisms, domains of partial recursive functions are not particularly well-behaved (we have to impose more conditions on morphisms to preserve this structure, see for example the discussion of dominances in [9]). Theorem 5.1 also exemplifies this fact. We therefore avoid domains, and show the following instead:

Proposition 4.1. *The application in $\mathscr{K}_1[E]$ is Π^1_1-complete.*

Proof. We show here that the application function is Π^1_1, since that does not follow immediately from Hinman's results. Let $r \in \mathbb{N}$ be the uniform index for f in $\mathscr{K}_1[f]$, for any f. Recall that $\mathscr{K}_1[E] = \mathscr{K}_1[g]$, where we defined g in stages. We will show that g is Π^1_1. Let e be an index such that whenever ϕ is the characteristic function of the graph of a partial function $f : A \to A$, then for all $b, x \in \mathscr{K}_1$:

$$eb \cdot^{\phi} x \simeq b \cdot^{f} x.$$

It is easy to see that such e exists; since $\mathbb{N}$ is enumerable, we can just "wait" for values of f to appear in the graph.

Define the following predicate on $\mathbb{N} \times \mathbb{N}^{\mathbb{N}}$:

$$\Phi\big([b,i],X\big) = (\forall s\,\exists t\,\forall y) : \big(eb \cdot^X s \downarrow \wedge \big(eb \cdot^X t = 0 \vee i = 1\big) \wedge \big(eb \cdot^X y \neq 0 \vee i = 0\big)\big). \tag{18}$$

It is not hard to see that Φ is Π^1_1 (in fact it is arithmetical) (to work out the details, one needs to use Kleene's T-predicate and existential quantification over dialogues). Let G be the characteristic function of the graph of g. Now we claim:

$$G\big([b,i]\big) \iff (\forall X)(\exists p,j)\big(\Phi\big([p,j],X\big) \Longrightarrow X\big([p,j]\big)\big) \implies X\big([b,i]\big). \tag{19}$$

Proof of claim. First, observe:

$$\Phi\big([b,i],G\big) \iff (\forall s)(\exists t\,\forall y) eb \cdot^G s \downarrow \wedge \big(eb \cdot^G t = 0 \vee i = 1\big) \wedge \big(eb \cdot^G y \neq 0 \vee i = 0\big) \tag{20}$$

$$\iff (\forall s)(\exists t\,\forall y) b \cdot^g s \downarrow \wedge \big(b \cdot^g t = 0 \vee i = 1\big) \wedge \big(b \cdot^g y \neq 0 \vee i = 0\big) \tag{21}$$

$$\iff G\big([b,i]\big). \tag{22}$$

This proves the "⇐" direction of (19) (we substitute G for X).

Now assume $G([b, i])$, that is:

$$(\forall s)(\exists t\ \forall y) b \cdot^g s \downarrow \wedge \big(b \cdot^g t = 0 \vee i = 1\big) \wedge \big(b \cdot^g y \neq 0 \vee i = 0\big).$$

Suppose we have X such that

$$(\forall p, j) : \Phi\big([p, j], X\big) \quad \Longrightarrow \quad X\big([p, j]\big).$$

We need to prove that $X([b, i])$. We do this by showing that

$$(\forall p, j) : G\big([p, j]\big) \quad \Longrightarrow \quad X\big([p, j]\big). \tag{23}$$

This is done by induction on stages (see Remark 3.2). Recall that g is constructed as $g = \bigcup_\alpha g_\alpha$. Denote the corresponding graphs by G_α. Then for $\alpha = 0$,

$$(\forall p, j) : G\big([p, j]\big) \quad \Longrightarrow \quad X\big([p, j]\big)$$

clearly holds, since $g_0 = \emptyset$. If it holds up to α, then:

$$\begin{aligned}
G_{\alpha+1}[p, j] \quad &\Longrightarrow \quad (\forall s)(\exists t\ \forall y) p \cdot^{g_\alpha} s \downarrow \wedge \big(p \cdot^{g_\alpha} t = 0 \vee j = 1\big) \wedge \big(p \cdot^{g_\alpha} y \neq 0 \vee j = 0\big) && (24)\\
&\Longrightarrow \quad (\forall s)(\exists t\ \forall y) ep \cdot^{X} s \downarrow \wedge \big(ep \cdot^{X} t = 0 \vee j = 1\big) \wedge \big(ep \cdot^{X} y \neq 0 \vee j = 0\big) && (25)\\
&\Longrightarrow \quad \Phi\big([p, j], X\big) && (26)\\
&\Longrightarrow \quad X\big([p, j]\big). && (27)
\end{aligned}$$

For limit ordinals λ, $G_\lambda[p, j] \Leftrightarrow G_\alpha[p, j]$ for some $\alpha < \lambda$ so that follows trivially.

Hence we have shown (23), from which $X[b, i]$ follows by assumption. □

Since the right-hand side of (19) is Π^1_1, we have shown that the graph of g is Π^1_1. Then for the application, we find:

$$\begin{aligned}
a \cdot^g b = c \quad \Longleftrightarrow \quad &\big(\exists k\ \exists u = [u_0, \ldots, u_k]\ \exists v = [v_0, \ldots, v_k]\big) : && (28)\\
&\bigwedge_i G\big([u_i, v_i]\big) \wedge a\big([b, u_0, \ldots, u_{i-1}]\big) = \mathsf{p}\mathsf{F}u_i \wedge a\big([b] * u\big) = \mathsf{p}\mathsf{T}c && (29)
\end{aligned}$$

which is clearly Π^1_1, since any Π^1_1-formula is closed under quantification.

To see that the application is Π^1_1 complete, see [4], Section VI.1. There one can find a proof in which a well-known Π^1_1-complete set is reduced to a *semi-recursive* set computable from E in Hinman's sense (i.e. a domain of a partial recursive function). This proof is easily adapted to a proof of Π^1_1-completeness for our case. □

5. Strict pcas

In this short section we wish to prove a result that was announced in the Introduction: the fact that any pca as we have defined above is isomorphic to a *strict pca*. This result by itself seems to have little to do with the rest of the content of this paper. However, in Proposition 6.3 we use the above results to give an example of a pca that seems far from being strict. This forms a pleasing contrast with the theorem below. Moreover, it is Kleene's (S1)–(S9) that inspired the proof of the theorem. Indeed, it is an easy observation that the application function defined in Definition 4.1 makes sense for any pca (except for primitive recursion), because we have pairing combinators and Curry numerals. We can therefore use clauses similar to Kleene's (S1)–(S9) to define a new application function from an old one, which is exactly what is done below to "strictify" any partial combinatory algebra.

Remark 5.1. Whenever we talk about "isomorphism" of partial combinatory algebras, we mean isomorphic as objects of the category of pcas with morphisms the applicative morphisms. However, it is not hard to see that in this case "isomorphic" coincides with "isomorphic" as algebras, i.e. the two inverse applicative morphisms have to be functions.

First, recall the definition of a strict pca:

Definition 5.1. Let A be a partial applicative structure. Then A is called a *strict pca* when there are $\mathsf{k}, \mathsf{s} \in A$ such that for all $a, b, c \in A$:

(1) $\mathsf{s}ab \downarrow$,
(2) $\mathsf{k}ab = a$,
(3) $\mathsf{s}abc \simeq ac(bc)$.

We also call the applicative structure on A *strictly combinatory complete*.

This definition might be preferable in some cases, for example when one would like that the set of domains of partial computable functions in a pca A form a *dominance* in $\mathrm{RT}(A)$ (see [9]). However, as mentioned before, these domains are not part of the structure of a pca in the category PCA.

The following theorem supports this: in the context of applicative morphisms, the notions of a pca and a strict pca are essentially the same.

Theorem 5.1. *Every pca is isomorphic to a strict pca.*

Proof. Let A be a pca. If A is trivial, then the statement is obvious, so we assume A non-trivial.

The idea is to define an explicit pca structure A' on the set A, such $a \mapsto \{a\}$ is an applicative morphism $A' \to A$. In this explicit structure, we also make sure that we can compute the application in A.

To define the new applicative structure on A, we use tuples and Curry numerals from A to ensure that what we define is actually computable in A. We denote the new application by $\cdot : A \times A \to A$, and the resulting partial applicative structure by A'. The definition is by induction on the following clauses:

(1) Constant function: for all $a, b, \in A$:

$$[\overline{1}] \cdot a = [\overline{1}, a], \tag{30}$$

$$[\overline{1}, a] \cdot b = a. \tag{31}$$

(2) S-combinator: For $a, b, c \in A$:

$$[\overline{2}] \cdot [a, b, c] \simeq a \cdot c \cdot (b \cdot c).$$

(3) Currying of parameters: for all $e, a, b, c \in A$:

$$[\overline{3}, e] \cdot a = [\overline{3}, e, a], \tag{32}$$

$$[\overline{3}, e, a] \cdot b = [\overline{3}, e, a, b], \tag{33}$$

$$[\overline{3}, e, a, b] \cdot c \simeq e \cdot [a, b, c]. \tag{34}$$

(4) Application in A:

$$[\overline{4}] \cdot a = [\overline{4}, a], \tag{35}$$

$$[\overline{4}, a] \cdot b \simeq ab. \tag{36}$$

Since A is non-trivial, the above is well defined. Strict combinatory completeness of A is satisfied by taking $\mathsf{k}' = [\overline{1}]$, $\mathsf{s}' = [\overline{3}, [\overline{2}]]$. Now $a \mapsto \{a\}$ is realized by $[\overline{4}]$ as an applicative morphism $A \to A'$. In the other direction, we have to simulate the above defined application in A. Using the fact that we can distinguish Curry numerals, determine length of tuples, and the recursion theorem, we can obtain an index $r \in A$ that performs the above case distinction, and in case the input is well formed (i.e. one of the tuples $[\overline{1}, \ldots], \ldots, [\overline{4}, \ldots]$ and an element in A), yields the same value as the application function in A'. □

Remark 5.2. Note that the proof above is entirely constructive; to define the application it only uses natural induction and explicit constructions with the combinators k and s.

Another observation is that if we would define A' using only the clauses (1)–(3), we obtain a strict pca A', together with an applicative morphism $A' \to A$ given by $a \mapsto \{a\}$, that has "forgotten" a lot of the structure of A. We can then adjoin certain partial functions in A to A' to get some of them back (such as the application, so that we obtain A again). In the case of a decidable pca, for instance, we would lose decidability in this way. So, not only can we adjoin functions to a pca as an oracle, we are also able to forget some functions in a pca.

6. Local operators

In [15], a notion of realizability is studied that uses a specific local operator J on the Effective Topos. We will show here that some of the results in that paper can be derived easily using the functional E from Example 3.1 (for $A = \mathscr{K}_1$) and Theorem 3.1.

We first give a definition of local operators in realizability toposes on a pca A, which we assume fixed from now on. The topos-theoretic origin is omitted here, for more about that we refer to [14] or [12]. We introduce the following notation: For $P, Q \subseteq A$ subsets, let

$$P \to Q = \big\{a \in A \mid (\forall b \in P)ab \downarrow \text{ and } ab \in Q\big\}.$$

Definition 6.1. A *local operator* is a function $J : \mathscr{P}(A) \to \mathscr{P}(A)$ such that there are $e_1, e_2, e_3 \in \mathbb{N}$ that satisfy:

(1) $e_1 \in \bigcap_{P \subseteq A} P \to JP$;
(2) $e_2 \in \bigcap_{P \subseteq A} JJP \to JP$;
(3) $e_3 \in \bigcap_{P,Q \subseteq A} (P \to Q) \to (JP \to JQ)$.

We define a pre-order on local operators as follows: we say $J \leqslant J'$ whenever there is $e \in A$ such that:

$$e \in \bigcap_{P \subseteq A} JP \to J'P.$$

We write $J \cong J'$ if $J \leqslant J'$ and $J' \leqslant J$. There is a least local operator $J_\bot$, given by the identity: $J_\bot(P) = P$. There is also a biggest local operator given by $J_\top(P) = A$.

The lattice of local operators (up to isomorphism) corresponds to the lattice of subtoposes of the realizability topos on A, denoted $\mathrm{RT}(A)$. Some of these subtoposes are realizability toposes $\mathrm{RT}(B)$ themselves, in that case we have found a *geometric inclusion* $\mathrm{RT}(B) \to \mathrm{RT}(A)$.

Definition 6.2 ([5])**.** An applicative morphism $\gamma : A \to B$ is called *computationally dense* if there exists $m \in B$ such that:

$$\text{for every } b \in B, \text{ there exists } a \in A \text{ such that whenever } b\gamma(a') \downarrow \text{ for some } a',$$
$$m\gamma(aa') \subseteq b\gamma(a'). \tag{37}$$

This should be read as follows: relative to γ, any application of an element b in B to an element in A (i.e. in $\gamma(a')$) can be computed in A (by an element a), up to a last application in B that is uniform in b (the element m). Think of m as "translating" the result of the computation from A to B.

There is a one-to-one correspondence between geometric morphisms $\mathrm{RT}(B) \to \mathrm{RT}(A)$ and computationally dense morphisms $A \to B$ (the above intuition might give a feel for why the arrows get reversed!).

We can also characterize geometric *inclusions* $\mathrm{RT}(B) \to \mathrm{RT}(A)$: they correspond precisely to computationally dense applicative morphisms $\gamma : A \to B$ that satisfy the additional condition

(in) there exists $e \in B$ such that $(\forall b)(\exists a)\ \big(eb \in \gamma(a)$ and $m\gamma(a) = \{b\}\big)$,

where m is as in Definition 6.2 (this is a consequence of results in [5,7]). In that case the corresponding local operator in $\mathrm{RT}(A)$ is given by

$$J(P) = \left\{ a \in A \;\middle|\; m\gamma(a) \subseteq \bigcup_{a' \in P} \gamma\big(a'\big) \right\}.$$

The following corresponds to Proposition 2.2 in [13].

Proposition 6.1. *For $f : A \to A$ a partial function, the identity relation on A is a computationally dense applicative morphism $\iota_f : A \to A[f]$ that satisfies the condition* (in). *Therefore, there is a canonical geometric inclusion* $\mathrm{RT}(A[f]) \to \mathrm{RT}(A)$.

By our construction, we obtain for every (partial) functional $F : A^A \to A$ a subtopos $\mathrm{RT}(A[F]) \to \mathrm{RT}(A)$ corresponding to the local operator

$$J_F(P) = \big\{ a \in A \mid m \cdot^F a \in P \big\}.$$

One can check that this indeed defines a local operator according to Definition 6.1.

6.1. J-assemblies

In [6], Martin Hyland showed that for every function $f : \mathbb{N} \to \mathbb{N}$, there is a least local operator that forces f to be computable in the Effective Topos. This yields an embedding of the Turing degrees into the lattice of local operators. In [11], Wesley Phoa showed that the corresponding subtopos is equivalent to $\mathrm{RT}(\mathscr{K}_1^f)$, the realizability topos on the pca of recursive application with oracle f. This result can be generalized to arbitrary pcas and partial functions: for (partial) functions $f : A \to A$, the least subtopos of $\mathrm{RT}(A)$ in which f is realizable is equivalent to $\mathrm{RT}(A[f])$. For a proof, see [1].

We will now carry out a similar statement for effective operations. To make things easier, we will work on a category of assemblies. The consequences for realizability toposes are immediate for anyone familiar with them. In the following we assume that we have fixed some pca A.

Definition 6.3. Let J be a local operator for A. A *J-assembly* is a pair (X, E) where $E : X \to A$ is a total relation.

For (X, E), (Y, F) J-assemblies, a *morphism* of J-assemblies is a function $f : X \to Y$ together with a $t \in A$ such that

$$t \in \bigcap_{x \in X} E(x) \to JF\big(f(x)\big).$$

We say that t *tracks* f relative to J.

One can check that the composition $g \circ f$ of morphisms of J-assemblies $f : (X, E) \to (Y, F)$, $g : (Y, F) \to (Z, G)$ is again a morphism of J-assemblies, so that J-assemblies form a category $\mathrm{Ass}(A)_J$. The category $\mathrm{Ass}(A)_J$

has a lot of structure: it is regular, Cartesian closed and has finite colimits (in fact, it is a *quasi-topos*). For (X, E), (Y, F) J-assemblies, the exponential $(Z, G) := (Y, F)^{X,E}$ is given by:

$$Z = \big\{ f : X \to Y \mid f \text{ is a morphism } (X, E) \to (Y, F) \big\}, \tag{38}$$

$$G(f) = \{ t \in A \mid t \text{ tracks } f \}. \tag{39}$$

The assembly $\mathbf{A} = (A, \mathrm{id}_A)$ can be seen as a representation of the pca A in the category of J-assemblies. The object $\mathbf{A}^{\mathbf{A}}$ consists of the morphisms $\mathbf{A} \to \mathbf{A}$ in $\mathrm{Ass}(A)_J$. For $J = J_\perp$, these are precisely the computable functions in A. Also, one can check that the arrows

$$\mathbf{A}^{\mathbf{A}} \to \mathbf{A}$$

in $\mathrm{Ass}(A)_{J_\perp}$ are precisely the effective operations in A. We have the following theorem:

Theorem 6.1. *Let $F : A^A \to A$ be a total functional, and assume that F defines a morphism*

$$\mathbf{A}^{\mathbf{A}} \to \mathbf{A}$$

in $\mathrm{Ass}(A)_J$. *Then* $J_F \leqslant J$.

Proof. We can assume that J preserves inclusions, otherwise we let

$$J'P = \bigcup_{Q \subseteq P} JP$$

and show that $J' \cong J$, which is an easy exercise using Definition 6.1.

By computational density, there exists $t \in A$ such that for all $v, x \in A$:

$$m \cdot^F (tvx) \lesssim v \cdot^F x.$$

Since m is independent of F, we also know that if the right-hand side halts at some stage, the left-hand side also halts at that stage.

We pick e_1, e_2, e_3 as in Definition 6.1.

Suppose that $F : \mathbf{A}^{\mathbf{A}} \to \mathbf{A}$ is tracked by r as morphism of J-assemblies.

Choose $\mathsf{m} \in A$ (using the recursion theorem) so that:

$$\mathsf{m}xy \lesssim \mathsf{if}\ \mathsf{p}_0\big(m([x] * y)\big) \tag{40}$$

$$\qquad \mathsf{then}\ e_1\big(\mathsf{p}_1\big(m([x] * y)\big)\big) \tag{41}$$

$$\qquad \mathsf{else}\ e_2\big(e_3\big(\langle z\rangle \mathsf{m}x\big(y * [z]\big)\big) r\big(\langle w\rangle\big(\mathsf{m}(tvw)[]\big)\big)\big), \tag{42}$$

$$\qquad \text{where } v = p_1\big(m([x] * y)\big). \tag{43}$$

Here [] is the empty sequence. We will prove by induction on the stage at which $m \cdot^F x \downarrow$ that for all x,

$$\mathsf{m}x[] \in J\big\{m \cdot^F x\big\}. \tag{44}$$

Stage 0: in that case $m([x]) = \mathsf{p}\mathsf{T}c$ for some c. Then

$$\mathsf{m}x[] = e_1\big(\mathsf{p}_1\big(m([x] * [])\big)\big) \in J\{c\}. \tag{45}$$

Stage $\lambda > 0$: Let $u = [u_0, \ldots, u_n]$ be the halting dialogue between m and x. It is easy to see that

$$\mathsf{m}x[u_0, \ldots, u_n] \in J\{m \cdot^F x\}.$$

Now suppose that $0 < i \leqslant n+1$ is such that

$$\mathsf{m}\big([x] * u^{<i+1}\big] \in J\{m \cdot^F x\}. \tag{46}$$

We know that:

$$m\big([x] * u^{<i}\big) = \mathsf{pF}v,$$

where v is such that at some stage $\alpha < \lambda$, for each w, $v \cdot^F w \downarrow$. So for each w, also $m \cdot^F (tvw) \downarrow$ at stage α. Let f be the function $w \mapsto v \cdot^F w$. By induction hypothesis,

$$\mathsf{m}(tvw)[] \in J\{v \cdot^F w\}$$

for each w, so $\langle w\rangle\mathsf{m}(tvw)[]$ tracks f as morphism of J-assemblies. So

$$r\big(\langle w\rangle\big(\mathsf{m}(tvw)l\big)\big) \in J\{F(f)\}$$

and $F(f) = u_i$. By (46), we have:

$$\langle z\rangle\mathsf{m}\big([x] * u^{<i} * [z]\big) \in \{F(f)\} \to J\{m \cdot^F x\}$$

and therefore

$$e_2\big(e_3\big(\langle z\rangle\mathsf{m}x\big(u^{<i} * [z]\big)\big)r\big(\langle w\rangle\big(\mathsf{m}(tvw)[]\big)\big)\big) \in J\{m \cdot^F x\}$$

hence it follows that

$$\mathsf{m}\big([x] * u^{<i}\big) \in J\{m \cdot^F x\}. \tag{47}$$

By reverse induction, it follows that

$$\mathsf{m}x[] \in J\{m \cdot^F x\},$$

so this finishes the induction step, so we have (44).

Therefore

$$\langle x\rangle\mathsf{m}x[] \in \bigcap_{P \subseteq A} J_F P \to J P$$

and we are done. □

6.2. Pitts' local operator

In this section we will take a look at a specific local operator that is studied in [15], which we will denote by J_P. It turns out that we can use the results of the previous section to say something about its complexity. This operator was introduced by Andrew Pitts in his thesis (Example 5.8, [12]). Denote by $\nabla(\mathbb{N})$ the assembly

$$(\mathbb{N}, T), \quad \text{where } T(n) = \mathbb{N} \text{ for all } n.$$

We can define the following subobject of $\nabla(\mathbb{N})$:

$$(\mathbb{N}, R) \rightarrowtail \nabla(\mathbb{N}), \quad \text{where } R(n) = \{m \in \mathbb{N} \mid m \geqslant n\}.$$

Pitts' local operator J_P is defined as the least local operator in $\mathrm{RT}(\mathscr{K}_1)$ that forces the above arrow to be an isomorphism of J_P-assemblies. Explicitly, it is given by:

$$J_P(S) = \bigcap \left\{ Q \subseteq \mathbb{N} \;\middle|\; \{0\} \wedge S \subseteq Q \text{ and } \{1\} \wedge \left(\bigcup_{n \in \mathbb{N}} \big(R(n) \to Q\big) \right) \subseteq Q \right\},$$

where for $R, S \subseteq \mathbb{N}$:

$$R \wedge S = \{\mathsf{p} r s \mid r \in R, s \in S\}.$$

Write $\mathbf{N}$ for the assembly $\mathbf{A}$ for $A = \mathscr{K}_1$, so

$$(\mathbb{N}, N), \quad \text{where } N(n) = \{n\}.$$

Recall the functional E from Example 3.1. For $A = \mathscr{K}_1$, we rather define it as

$$E(f) = \begin{cases} 0 & \text{if } (\exists n) f(n), \\ 1 & \text{otherwise.} \end{cases}$$

We have the following proposition:

Proposition 6.2. *There is an index $r_E \in \mathbb{N}$ that tracks E as morphism of J_P-assemblies $\mathbf{N}^{\mathbf{N}} \to \mathbf{N}$.*

Proof. Pick e_3 as in Definition 6.1.

By Corollary 1.6 of [15], there is a partial recursive function G such that for all

$$x_0 \in J_P\{a_0\}, \ldots, x_{n-1} \in J_P\{a_{n-1}\},$$

we have

$$G\big(\langle x_0, \ldots, x_{n-1}\rangle\big) \in J_P\{0\} \quad \text{if for some } i < n, a_i = 0, \tag{48}$$

$$G\big(\langle x_0, \ldots, x_{n-1}\rangle\big) \in J_P\{1\} \quad \text{otherwise.} \tag{49}$$

Now let t_E be an index in $\mathscr{K}_1$ defined by:

$$t_E e n \simeq G\big(\langle e(0), \ldots, e(n)\rangle\big).$$

Then whenever e realizes a total function $f : \mathbf{N} \to \mathbf{N}$, we have $(\exists n) f(n) = 0$ if and only if $(\exists n) en \in J_P\{0\}$, which holds if and only if there is n such that

$$t_E e \in \{m \mid m \geqslant n\} \to J_P\{0\}$$

since otherwise

$$t_E e \in \{m \mid m \geqslant 0\} \to J_P\{1\}.$$

By definition of J_P, we have in the first case that $t_E e \in J_P\{0\}$, and in the latter case $t_E e \in J_P\{1\}$. Now t_E tracks E as morphism of J_P-assemblies $\mathbf{N}^{\mathbf{N}} \to \mathbf{N}$. □

The following corollary corresponds to Theorem 2.2 in [15].

Corollary 6.1. *The morphisms of J_P-assemblies $\mathbf{N} \to \mathbf{N}$ are precisely the hyperarithmetical functions.*

Proof. By Theorem 6.1, $J_E \leqslant J_P$. Therefore it is easily seen that every morphism of J_E assemblies $\mathbf{N} \to \mathbf{N}$ is a morphism of J_P-assemblies. Now it follows that every hyperarithmetical function is a morphism of J_P-assemblies by Corollary 4.1.

For the converse, observe that every morphism $f : \mathbf{N} \to \mathbf{N}$ is Π^1_1 since:

$$f(n) = m \quad \Longleftrightarrow \quad (\forall B)\{\mathsf{p}0m\} \subseteq B \wedge \{1\} \wedge \big(\{e \mid (\exists n)(\forall m \geqslant n) em \in B\}\big) \subseteq B \to e(\mathsf{a}n) \in B,$$

where e is a realizer for f as a morphism $\mathbf{N} \to \mathbf{N}$. Notice that the above expression is Π^1_1. Since f is total, it is also Σ^1_1.

Therefore the morphisms of J_P-assemblies $\mathbf{N} \to \mathbf{N}$ are precisely the hyperarithmetical functions. □

6.3. J_P-representable functions

In [15], the morphisms of J_P-assemblies $\mathbf{N} \to \mathbf{N}$ are called *J_P-representable*. Since for every $n, m \in \mathbb{N}$

$$n \neq m \quad \Longleftrightarrow \quad J_P\{n\} \cap J_P\{m\} = \emptyset, \tag{50}$$

the following is well defined:

Definition 6.4. A partial function $F : \mathbb{N} \to \mathbb{N}$ is *J_P-representable* if there exists $e \in \mathbb{N}$ such that for all $m \in \mathbb{N}$:

$$\phi_e(n) \in J_P\{m\} \quad \Longleftrightarrow \quad F(n) = m.$$

Here ϕ_e denotes the eth partial computable function from ordinary recursion theory (or the application in $\mathscr{K}_1$).

By defining $\psi_e = F$ for such e, we obtain an indexed set of partial functions and a new application

$$e \cdot n \mapsto \psi_e(n)$$

on $\mathbb{N}$.

A natural question is whether the application $e \cdot n \simeq \psi_e(n)$ is combinatory complete in the sense of Definition 2.3. This turns out to be the case:

Proposition 6.3. *The application $e \cdot n \simeq \psi_e(n)$ defines a pca $\mathscr{P}_1$ on $\mathbb{N}$.*

Proof. This is an easy exercise on pcas and Definition 6.1. □

The pca $\mathscr{P}_1$ defined above is an example of a pca that is not strict (although this might be difficult to show) and it seems very hard to define it in a similar way from J_P that would make it strict. For this specific example, Theorem 5.1 is therefore a reassuring result (although in this case, it could also have been proven, with some care, using the theorem below).

Finally, we have the following theorem:

Theorem 6.2. *The pca $\mathscr{P}_1$ is isomorphic to $\mathscr{K}_1[E]$.*

Proof. Let $\gamma : \mathscr{K}_1 \to \mathscr{P}_1$ be the applicative morphism $n \mapsto \{n\}$ (it is easily verified that this is a decidable applicative morphism).

Now $E \in E_\gamma$ by Proposition 6.2. Therefore, by Theorem 3.1, γ factors through $\mathscr{K}_1[E]$, hence we have a decidable applicative morphism $\gamma_E : \mathscr{K}_1[E] \to \mathscr{P}_1$, given by the identity on $\mathbb{N}$.

This morphism also goes the other way: this follows from the fact that the application in $\mathscr{K}_1[E]$ is Π^1_1-complete (Proposition 4.1), and the fact that the application in $\mathscr{P}_1$ is Π^1_1. □

7. Higher types

One may wonder whether Theorem 4.2 also holds for computability in higher type functionals (so, type 3 and higher). The answer to that question is positive: for functionals $x_1, \ldots, x_n$ of certain types, one can define in a similar fashion a pca

$$\mathscr{K}_1[x_1, \ldots, x_n]$$

that is again isomorphic to the pca of recursive application relative to $x_1, \ldots, x_n$ in Kleene's sense. The idea is to define $\mathscr{K}_1[x_1, \ldots, x_n] = \mathscr{K}_1[f]$, where f is defined in stages, simultaneously for $x_1, \ldots, x_n$, so that the resulting applicative structure is closed under application of any of the functionals. However, we do not get much more than that. In the above, we have shown that constructing the pca of recursive application relative to a type 2 functional F is the same thing as "forcing F to be a realizable as effective operation" (taking note of Remark 3.1), or forcing F to be realizable in the corresponding category of assemblies or realizability topos. To see what goes wrong, we will sketch the case for a single type 3 functional $\Phi : A^{A^A} \to A$.

For such Φ, we can define a pca $A[\Phi]$ as $A[f]$, where f satisfies:

$$\text{for } e \in A,\ F : A^A \to A \text{ such that } (\forall g : A \to A)e \cdot^{g,f} \mathsf{T} = F(g), \quad f(e) = \Phi(F).$$

Here $\cdot^{g,f}$ is defined as $\cdot^h$ for $h(a) = \mathsf{p}g(a)f(a)$. An f satisfying the above equation can again be defined in (transfinitely many) stages. The resulting pca $A[\Phi]$ will satisfy the property that whenever e is an index such that

$$e \cdot^{g,\Phi} \mathsf{T} = F(g) \quad \big(\text{w.r.t. the application in } A[\Phi]\big)$$

for some functional $F : A^A \to A$, then we can compute $\Phi(F)$ in $A[\Phi]$, i.e. there is an index r such that $r \cdot^\Phi e = \Phi(F)$, uniformly in such e. The reason we have to define $A[\Phi]$ in this way is that Φ is defined on *total functionals* $A^A \to A$, and we want Φ to be defined on all total functionals that are "computable" in $A[\Phi]$. By a computable functional we mean here a total functional F that has in index in A which, when given an oracle g, computes $F(g)$.

Kleene uses a similar definition of "recursive application relative to Φ" in [8], as does Hinman in [4]. One can again show that for $A = \mathscr{K}_1$, the resulting applicative structures are the same in the sense of Theorem 4.2. The same holds for general n-tuples of higher types $x_1, \ldots, x_n$.

However, Φ is not an effective operation relative to $A[\Phi]$. For a pca A, an effective operation of type 3 should be the same thing as a morphism of assemblies

$$\mathbf{A}^{\mathbf{A}^{\mathbf{A}}} \to \mathbf{A}. \tag{51}$$

So it is a map that sends every effective operation of type 2 to A. It is not clear that Φ is an effective operation in this sense for $A[\Phi]$ as defined above, in fact this seems completely unrelated. For instance, there could be effective operations that are not computable functionals in the above sense (an example of this kind can be found in [3]) so there is no guarantee that Φ can be evaluated on those. Conversely, the methods we have developed above do not seem to give a way to "force Φ to be realizable" as a morphism of type (51). At the very least, it seems to require a more involved method of forcing than just a transfinite induction. Denote for instance by $A[\Phi]'$ the hypothetical pca in which Φ is realizable as an effective operation of type 3. When Φ has different values on F and G, but $F \restriction \mathrm{Tot}_A = G \restriction \mathrm{Tot}_A$, we will first have to make sure that either $F \restriction \mathrm{Tot}_{A[\Phi]'} \neq G \restriction \mathrm{Tot}_{A[\Phi]'}$, or that F or G (or both) are no longer realizable as effective operations relative to $A[\Phi]'$, and make sure that this property is preserved in the construction of $A[\Phi]'$.

Another direction could be to study other notions of computable functionals, and to see to what extent these generalize to pcas and especially the theory of categories of assemblies and realizability toposes. The overview [10] is a good starting point here.

References

[1] E. Faber and J. van Oosten, More on geometric morphisms between realizability toposes, *Theory Appl. Categ.* **29** (2014), 874–895.
[2] J. Frey, A characterization of realizability toposes, 2014, available at: arXiv:1404.6997.
[3] R.O. Gandy and J.M.E. Hyland, Computable and recursively countable functions of higher type, in: *Logic Colloquium 76*, Oxford, 1976, Studies in Logic and the Foundations of Mathematics, Vol. 87, North-Holland, Amsterdam, 1977, pp. 407–438.
[4] P.G. Hinman, *Recursion-Theoretic Hierarchies*, Perspectives in Mathematical Logic, Springer, Berlin, 1978.
[5] P. Hofstra and J. van Oosten, Ordered partial combinatory algebras, *Math. Proc. Cambridge Philos. Soc.* **134**(3) (2003), 445–463.
[6] J.M.E. Hyland, The effective topos, in: *The L.E.J. Brouwer Centenary Symposium*, Noordwijkerhout, 1981, Studies in Logic and the Foundations of Mathematics, Vol. 110, North-Holland, Amsterdam, 1982, pp. 165–216.
[7] P. Johnstone, Geometric morphisms of realizability toposes, *Theory Appl. Categ.* **28**(9) (2013), 241–249.
[8] S.C. Kleene, Recursive functionals and quantifiers of finite types. I, *Trans. Amer. Math. Soc.* **91** (1959), 1–52.
[9] J.R. Longley, Realizability toposes and language semantics, PhD thesis, University of Edinburgh, 1995.
[10] J.R. Longley, Notions of computability at higher types. I, in: *Logic Colloquium 2000*, Lecture Notes in Logic, Vol. 19, Association for Symbolic Logic, Urbana, IL, 2005, pp. 32–142.
[11] W. Phoa, Relative computability in the effective topos, *Math. Proc. Cambridge Philos. Soc.* **106**(3) (1989), 419–422.
[12] A.M. Pitts, The theory of triposes, PhD thesis, Cambridge University, 1981.
[13] J. van Oosten, A general form of relative recursion, *Notre Dame J. Formal Logic* **47**(3) (2006), 311–318 (electronic).
[14] J. van Oosten, *Realizability: An Introduction to Its Categorical Side*, Studies in Logic and the Foundations of Mathematics, Vol. 152, Elsevier B.V., Amsterdam, 2008.
[15] J. van Oosten, Realizability with a local operator of A.M. Pitts, *Theoret. Comput. Sci.* **546** (2014), 237–243.