Electronic Notes in Theoretical Computer Science 74 (2003)
URL: http://www.elsevier.nl/locate/entcs/volume74.html LastPage pages

On the ubiquity of certain total type structures (Extended abstract)

John Longley [1,2]

LFCS, School of Informatics
University of Edinburgh
Edinburgh, UK

Abstract

It is a fact of experience from the study of higher type computability that a wide range of plausible approaches to defining a class of (hereditarily) total functionals over $\mathbb{N}$ results in a surprisingly small handful of distinct type structures. Among these are the type structure C of Kleene-Kreisel *continuous functionals*, its recursive substructure RC, and the type structure HEO of the *hereditarily effective operations*. However, the proofs of the relevant equivalences are often non-trivial, and it is not immediately clear why these particular type structures should arise so ubiquitously.

In this paper we present some new results which go some way towards explaining this phenomenon. Our results show that a large class of realizability-style constructions always give rise to C, RC or HEO (as appropriate). The proofs make essential use of a technique due to Dag Normann. In this extended abstract, we content ourselves with giving precise statements of our theorems, and a brief outline of the method of proof.

Several new results, and some previously known ones, can be obtained as instances of our theorems, but more importantly, the proofs apply uniformly to a whole family of constructions, and provide strong evidence that the above three type structures are highly canonical mathematical objects.

1 Introduction

In our survey paper [12], we discussed the general problem of trying to identify the natural notions of "computability" for operations of higher type, and reviewed the many different approaches to higher type computability that have

[1] Research partly supported by EPSRC grant GR/N64571 "A proof system for correct program development". The author wishes to thank the organizers of the Domains VI meeting for the opportunity to present this work, and also Andrej Bauer, Ulrich Berger, Martín Escardó, Dag Normann and Gordon Plotkin for illuminating discussions.
[2] Email: jrl@inf.ed.ac.uk

been proposed in the literature over the last fifty years. As argued in [12], it is far from clear *a priori* what one ought to mean by "computability" in the higher type setting, but it is a pleasing fact of experience that a wide range of plausible approaches leads in practice to a relatively small handful of distinct notions. Thus, for example, if we restrict attention to classes of effectively computable *functionals* (i.e. extensional operations) of simple types over the natural numbers, it seems that all known approaches essentially boil down to one of six notions: two notions of "total computable functional", and four notions of "partial computable functional". The situation is summarized at the end of [12], and will be expounded more fully in two sequel papers.

Each of these six notions admits a wide range of different mathematical characterizations, and in each case we have a bunch of theorems asserting that these various characterizations all lead to the same class of functionals. (This is rather analogous to the situation in ordinary recursion theory, where definitions via Turing machines, lambda calculus, recursion schemes etc. all gives rise to the same class of computable first order functions.) In the higher type setting, many of these theorems are non-trivial and surprising, since they often relate constructions which appear totally different in character. Equivalence results of this kind contribute to the impression that the notion of computability in question is in some sense a natural or fundamental one.

In this paper we will present some new results in this vein, concentrating in particular on certain classes of *hereditarily total* functionals. (Informally, these are total functionals which act on total functionals which act on total functionals ... and so on down to the ground type.) A class of such functionals can be conveniently embodied by a *total type structure* over the set $\mathbb{N}$ of natural numbers. A little terminology and notation will be helpful at this point. We will be considering the *simple types* σ generated by the grammar:

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

We inductively define $\overline{n+1}$ to be the type $\overline{n} \to \overline{0}$, and we refer to the types $\overline{n}$ as the *pure types*. For the purposes of this paper, a *type structure* T will consist of a family of sets T_σ, one for each type σ, such that $T_{\overline{0}} = \mathbb{N}$, and $T_{\sigma\to\tau}$ is some set of total functions from T_σ to T_τ. We sometimes write T_n in place of $T_{\overline{n}}$, and more generally replace $\overline{n}$ by n when it appears as a subscript.

Several natural ways of constructing type structures of this kind are surveyed in [12], along with a number of equivalence results relating such constructions. A consideration of these results leads us to observe that there are three type structures which seem to arise with a remarkable frequency:

(i) Constructions based on some idea of "continuous functions acting on continuous data" tend to lead consistently to the type structure C of Kreisel's *continuous functionals* [7], or equivalently Kleene's *countable functionals* [5].

(ii) Constructions based on an idea of "effective functions on continuous data" tend to lead to the type structure RC of *recursive continuous func-*

tionals (this can be regarded as an effective substructure of C).

(iii) Constructions based on an idea of "effective functions on effective data" tend to lead to the type structure HRC of *hereditarily recursively continuous functionals*, or equivalently to the type structure HEO of *hereditarily effective operations* [7,16].

The words "continuous" and "effective" here can be made precise in various different ways; the point is that in each of the three cases, all reasonable choices lead to the same structure.

We will give some precise definitions of all three of these type structures in Section 3.4 below. The structure C is not itself a candidate for a class of "computable" functions (unless one allows one's "programs" to be infinite objects), but it nevertheless plays an important role in the study of higher type computability. The other two type structures, RC and HEO, represent the two reasonable notions of "total computable functional" alluded to in our first paragraph. [3] These two notions are quite different in flavour and are in some sense incomparable: there are functionals in HEO that have no counterpart in RC, and *vice versa.* We refer the reader to [12] for further background in this area.

It thus appears that for definitions of a class of total computable functionals (setting aside the notion of Kleene computability), the main dichotomy is between approaches in which functions defined on arbitrary continuous objects of the appropriate type, and approaches in which they defined on just the effective objects of the appropriate type. Other aspects of the definitions which we might have expected to be significant, such as how we choose to represent functions intensionally and what kinds of "computation" we allow, usually turn out not to make any difference. (Curiously, this is in stark opposition to the situation for notions of *partial* computable functional, where these other considerations turn out to be critical, but the dichotomy between "continuous" and "effective" is of minor importance.)

What we have sketched here is simply an impression that emerges from a number of individual results connecting up different constructions. It is therefore natural to ask whether one can give some kind of explanation for *why* these three type structures arise so ubiquitously, or whether our observations (i)–(iii) can be made precise in some way. The purpose of this paper is to present some results which go some way towards realizing these objectives. We will obtain some general theorems saying that *any* construction that follows a certain pattern will result in the type structure C (respectively RC, HEO). Our

[3] There is also another important notion of total computable functional, namely the one given by Kleene's schemata S1–S9 [6]. Strictly speaking, these schemata give several related notions of computability, since they can be interpreted in many different type structures. However, in the author's view, these notions of computability are best treated as derived from one of the notions of *partial* computable functional we consider. We will develop this point of view in more detail in the sequel to [12]. For the purpose of the present paper, we will leave to one side the notion of Kleene computability.

theorems will cover many (though not all) of the constructions hitherto considered in the literature, and so our approach will provide a unified explanation (and indeed a uniform proof) of several of the previously known equivalence results. In addition, a wealth of new characterizations of these type structures (many of them non-trivial in themselves) can be obtained as further instances of our theorems. Perhaps most importantly, by virtue of their generality, our results will furnish some kind of justification for the above statements (i)–(iii), and (in the author's view) provide strong confirmation that the structures C, RC and HEO are highly canonical mathematical objects.

The results of this paper are thus of a somewhat more sweeping character than many of the previous results in the area, since they apply uniformly to a whole class of constructions. We will therefore need some general framework for describing the class of constructions we have in mind. Roughly speaking, the constructions that are covered by our results are those that can be naturally presented as standard *realizability constructions* (see Section 2 below). In order to articulate our results, we will make use of a framework from realizability involving *typed partial combinatory algebras*, introduced by the author in [11]. As we shall see, many of the known constructions of C and RC, and virtually all known constructions of HEO, are easily shown to be equivalent to constructions that fit within our realizability framework. A plentiful supply of additional constructions is also provided by many of the structures considered in denotational semantics.

Since one typically has to work quite hard to obtain such results even for individual realizability constructions, the possibility of proving general theorems of this kind came as a considerable surprise, at least to the present author. Even in the light of the known equivalence results, nothing would previously have led him to suspect that such a general result might hold. The proofs of our theorems are non-trivial, and make essential use of a technique due to Normann [14], who used it to show that the total computable functionals arising from the standard Scott model were all definable in the language PCF. Normann's proof is already somewhat technical, and our own proofs are even more so, since several further ingredients need to be added. In this extended abstract, we will content ourselves with a very brief outline of our method of proof and the role of Normann's work; the full proof will appear in an expanded version of this paper.

1.1 Motivations

In general (as we have argued in [12, Section 4]), the *partial* notions of higher type computability are both intrinsically better behaved than the total notions and also have a greater relevance for computer science. We should therefore say a few words concerning some plausible motivations for the work undertaken in this paper, beyond its intrinsic conceptual interest.

One kind of motivation for the study of total computable functionals comes from the study of formal systems for constructive logic, such as first order Heyting arithmetic HA or its higher order counterpart HA^ω (see for example Troelstra [16] and Beeson [2]). For metamathematical purposes, one often considers interpretations of such systems (such as realizability interpretations) that embody some kind of constructive or computational content. Frequently, these interpretations are based on some notion of a constructive or computable operation of finite type. In the case of HA^ω with extensionality, in particular, one is naturally led to consider the class of the hereditarily total functionals that arise from such a notion, so in order to study properties of our interpretation we will surely want to know what this class of functionals is. The results of the present paper will answer this question in a large number of cases, allowing existing knowledge about the resulting class of functionals to be applied to the interpretation in question.

A second possible motivation comes from the study of the notion of *totality* in programming languages, as discussed for example in Plotkin [15]. In practice, higher order programming languages allow us to write non-terminating programs, and so give rise to notions of *partial* computable operation. However, for much of the time one wants to write terminating programs, and these are certainly easier to reason about since we do not have to worry about the possibility of "undefinedness", so it is of interest to consider the class of *total* computable operations that the language supports. As explained in [15], there are various subtle issues involved in defining a good notion of totality for programs of higher type; but in any case, we will certainly be interested in knowing what class of total operations we obtain from the language in question. Once again, in many cases the results of the present paper will enable us to answer this question, and will enable us to apply existing knowledge about the resulting class of functionals to the programming language under consideration.

1.2 Structure of paper

The remainder of the paper is structured as follows. In Section 2 we set up the realizability framework within which we will work, based around the notion of typed partial combinatory algebras (TPCAs) and the extensional collapse construction. In Section 3 we illustrate this framework with some concrete examples, and summarize some known constructions of our three type structures that fit within this framework. In Section 4 we introduce the key ideas of *effective* and *continuous* TPCAs. We use these to give a precise statement of our results, and briefly discuss our method of proof and the role played by Normann's ideas. We also mention a few applications and specific instances of our results. Finally, in Section 5 we try to assess the significance of our results, point out some of their limitations, and suggest some problems for further research.

2 A realizability framework

This section is concerned with setting up the general framework needed to express our results. In Section 2.1 we present a variant of the definition of *typed partial combinatory algebras* (TPCAs) which we introduced in [11] (under the name of *partial combinatory type structures*). In Section 2.2 we review the standard *extensional collapse* construction that we shall study, and in Section 2.3 we define a useful "relative" version of this construction. Concrete examples of all these notions will be given in Section 3.

Let us fix some notational conventions. We use ρ, σ, τ to range over types, and consider $\to$ to be right-associative, so that $\rho \to \sigma \to \tau$ means $\rho \to (\sigma \to \tau)$. In connection with potentially non-denoting expressions we write $e\!\downarrow$ to mean "e is defined"; $e = e'$ to mean "e, e' are defined and equal" (strict equality); and $e \simeq e'$ to mean "if either e or e' is defined then so is the other and they are equal" (Kleene equality).

2.1 Typed partial combinatory algebras

Many constructions giving rise to C, RC and HEO involve structures of the following kind. We will regard the following definition as providing our basic notion of a "universe of higher type operations".

Definition 2.1 [TPCA] A *typed partial combinatory algebra* (or *TPCA*) A consists of

- a set A_σ for each type σ;
- for each σ, τ, a partial function $\cdot_{\sigma\tau} : A_{\sigma\to\tau} \times A_\sigma \rightharpoonup A_\tau$ (known as *application*)

such that there exist elements

$$\begin{array}{rcll}
\mathbf{k}_{\sigma\tau} & \in & A_{\sigma\to\tau\to\sigma} & \text{(for each } \sigma, \tau) \\
\mathbf{s}_{\rho\sigma\tau} & \in & A_{(\rho\to\sigma\to\tau)\to(\rho\to\sigma)\to\rho\to\tau} & \text{(for each } \rho, \sigma, \tau) \\
\mathbf{y}_\rho & \in & A_{(\rho\to\rho)\to\rho} & \text{(for each type } \rho = \sigma \to \tau) \\
\widehat{0}, \widehat{1}, \widehat{2}, \ldots & \in & A_{\overline{0}} & \\
\mathbf{suc} & \in & A_{\overline{0}\to\overline{0}} & \\
\mathbf{rec}_\sigma & \in & A_{\sigma\to(\overline{0}\to\sigma\to\sigma)\to\overline{0}\to\sigma} & \text{(for each } \sigma)
\end{array}$$

for which the following conditions hold for all elements x, y, z, f of the appropriate types and for all $n \in \mathbb{N}$:

$$\begin{array}{rcl}
\mathbf{k} \cdot x \cdot y & = & x \\
\mathbf{s} \cdot x \cdot y & \downarrow & \\
\mathbf{s} \cdot x \cdot y \cdot z & \simeq & (x \cdot z) \cdot (y \cdot z) \\
\mathbf{y} \cdot f & \downarrow & \\
\mathbf{y} \cdot f \cdot x & \simeq & f \cdot (\mathbf{y} \cdot f) \cdot x
\end{array}$$

$$\begin{aligned} \mathbf{suc} \cdot \widehat{n} &= \widehat{n+1} \\ \mathbf{rec} \cdot x \cdot f \cdot \widehat{0} &= x \\ \mathbf{rec} \cdot x \cdot f \cdot \widehat{n+1} &\simeq f \cdot \widehat{n} \cdot (\mathbf{rec} \cdot x \cdot f \cdot \widehat{n}) \end{aligned}$$

We frequently omit type subscripts where these can be inferred from the context (we have taken this liberty even within the definition itself), and treat '$\cdot$' as left-associative, so that $x \cdot y \cdot z$ means $(x \cdot y) \cdot z$. Strictly speaking, the choice of elements $\mathbf{k}, \mathbf{s}, \mathbf{y}, \cdots$ is not part of the data for a TPCA — only the existence of such elements is required. Nevertheless, we will sometimes implicitly assume that a given TPCA comes equipped with a suitable choice of such elements.

Note that in the more usual definition of TPCA (see [11] or [8]), only the special elements $\mathbf{k}$ and $\mathbf{s}$ are required — in this terminology, a structure of the above kind would be a "TPCA with recursors and numerals". However, in the present paper we will for convenience include the existence of recursors and numerals in the definition of TPCA. The idea is that all the TPCAs we consider must have enough computational structure to support "general recursive computation".

Note also that the equation for $\mathbf{y}$ is weaker than the more familiar fixed point equation $\mathbf{y} \cdot f = f \cdot (\mathbf{y} \cdot f)$. This extra generality is exploited by some of the natural examples.

2.2 The extensional collapse construction

From any TPCA we may obtain a total type structure over $\mathbb{N}$ (in the sense of the Introduction) by means of the following construction. Recall that a *partial equivalence relation* or *PER* on a set X is just a symmetric, transitive relation on X (that is, an equivalence relation on a subset of X). If $\sim$ is a PER on X, we write $X/{\sim}$ for the set of equivalence classes for $\sim$, and $[x]$ for the equivalence class (if there is one) containing a particular $x \in X$.

Definition 2.2 [Extensional collapse] Given any TPCA A, define partial equivalence relations $\sim_\sigma$ on the sets A_σ as follows:

- $x \sim_0 y$ iff $x = y = \widehat{n}$ for some n;
- $f \sim_{\sigma\to\tau} g$ iff for all $x, y \in A_\sigma$, $x \sim_\sigma y$ implies $f \cdot x \sim_\tau g \cdot y$.

We now define $\mathsf{EC}(A)$, the *extensional collapse* of A, to be the unique type structure isomorphic to $A/\sim$. More precisely, we require that there exist bijections $\beta_\sigma : A_\sigma/\sim_\sigma \to \mathsf{EC}(A)_\sigma$ for each σ, such that

- for all $n \in \mathbb{N}$ we have $\beta_0[\widehat{n}] = n$;
- for all $f \sim_{\sigma\to\tau} f$ and $x \sim_\sigma x$ we have $\beta_{\sigma\to\tau}[f](\beta_\sigma[x]) = \beta_\tau[f \cdot x]$.

Thus, the PERs $\sim_\sigma$ pick out the "hereditarily extensional" elements of A, and the construction of $\mathsf{EC}(A)$ turns these into a type structure. (The isomorphisms β_σ are needed only because in our definition of a type structure

T in Section 1 we required that $T_{\sigma\to\tau}$ is concretely a set of functions from T_σ to T_τ).

It is worth noting that $\mathsf{EC}(A)$ corresponds precisely to the usual type structure over the natural number object in the category $\mathbf{PER}(A)$, obtained by repeated exponentiation (see [11]). Since we may regard $\mathbf{PER}(A)$ as the "standard realizability model" over A, we can think of the definition of $\mathsf{EC}(A)$ as the *standard realizability construction* of a type structure. If $\beta_\sigma[x] = t$, we often say that $x \in A$ *realizes* $t \in \mathsf{EC}(A)$. Apart from this, however, no formal knowledge of $\mathbf{PER}(A)$ and realizability will be presupposed in this paper.

Remark 2.3 (i) A superficially more general definition could be given by allowing a non-empty set of realizers for each natural number n rather than just a single element $\widehat{n}$. In other words, we could represent $\mathbb{N}$ by any PER $\approx_0$ on A_0 for which the successor and recursor operations were realizable in A. This would then induce PERs $\approx_\sigma$ at higher types as above, and we would obtain a type structure $\mathsf{EC}(A, \approx)$ isomorphic to $A/\approx$. However, it is easy to show that for any such representation $\approx_0$ of $\mathbb{N}$ there is an equivalent single-valued representation corresponding to some choice of numerals $\widehat{n}$; in particular, $\mathsf{EC}(A)$ defined as above (for this choice of numerals) coincides exactly with $\mathsf{EC}(A, \approx)$.

(ii) Another, even more superficial, generalization could be obtained by allowing the representing type of the natural numbers to be some type γ other than $\overline{0}$. We may reduce a TPCA A of this more general kind to a TPCA A^γ of the above kind just by setting $A^\gamma_\sigma = A_{\sigma[\gamma]}$, where

$$\overline{0}[\gamma] \;=\; \gamma, \qquad (\sigma \to \tau)[\gamma] \;=\; \sigma[\gamma] \to \tau[\gamma].$$

Remark 2.4 We would like to stress the fact that it is in general quite hard to establish relationships between type structures $\mathsf{EC}(A)$ and $\mathsf{EC}(B)$, even when A and B themselves are known to be closely related. For instance, one can make precise in various ways a relation $A \preceq B$ between TPCAs, informally meaning "A can be simulated in B", or "B has at least the computational power as A" (see e.g. [11]); we can loosely regard $\preceq$ as a kind of "inclusion" for TPCAs. One may likewise define a suitable relation $T \preceq U$ for type structures. However, in general $A \preceq B$ does not imply that $\mathsf{EC}(A) \preceq \mathsf{EC}(B)$; indeed, as we pass from some TPCA A to some $B \succeq A$, the extensional collapse may get bigger, or smaller, or remain the same, or jump sideways. Even if we have $A \preceq B \preceq C$ with $\mathsf{EC}(A) = \mathsf{EC}(C)$, any of these four relationships between $\mathsf{EC}(A)$ and $\mathsf{EC}(B)$ may arise. (The study of higher-type computability furnishes examples of all these situations, but we will not spell these out here.) All this is because the construction $\mathsf{EC}(-)$ is highly *non-functorial* — it does not interact well with "morphisms" between TPCAs — and the root of the problem is of course the "contravariance" built into the definition of $\sim_\sigma$.

Our reason for labouring these points is to emphasize that, in general, one has to establish what $\mathsf{EC}(A)$ is *separately* for each TPCA A — it is not often that one is able to deduce such results cheaply from other known results of this kind, except in rather trivial cases. Thus, from the point of view of the operation $\mathsf{EC}(-)$, we have to consider each TPCA as living in its own bubble, so that extensional collapse results, while they may be non-trivial and beautiful theorems in themselves, typically have a rather "isolated" character and do not connect up with one another in any obvious way.

The main novelty of the results presented in this paper is that they offer, in certain cases, a way to break the isolation of individual TPCAs, and to establish the extensional collapse of a whole class $\mathcal{K}$ of TPCAs all at once. This gives us a much more powerful result than would be given by simply establishing the extensional collapse for the "minimal" and "maximal" elements of $\mathcal{K}$, for instance.

2.3 A relative version

In the definition of TPCAs above, operations of higher type are treated as entities of the same kind as the data they act on: both are simply elements of A. Thus, TPCAs are a suitable framework for notions such as "continuous operations acting on continuous data", or "effective operations acting on effective data". However, some refinements to this framework are needed if we wish to consider notions such as "effective operations acting on continuous data" — or, more generally, a notion of operations of some restricted class acting on data in some wider class. We may capture such notions by means of a *relative* version of the extensional collapse construction, which we now introduce.

Definition 2.5 [Relative extensional collapse]

(i) A *sub-TPCA* A' of a TPCA A consists of a family of subsets $A'_\sigma \subseteq A_\sigma$, one for each σ, such that (for some $\mathbf{k}, \mathbf{s}, \mathbf{y}, \widehat{n}, \mathbf{suc}, \mathbf{rec}$ suitable for A) we have

- for all ρ, σ, τ and all $n \in \mathbb{N}$, $\mathbf{k}_{\sigma\tau}, \mathbf{s}_{\rho\sigma\tau}, \mathbf{y}_\sigma, \widehat{n}, \mathbf{suc}, \mathbf{rec}_\sigma \in A'$;
- A' is closed under application, in the sense that

$$f \in A'_{\sigma\to\tau} \wedge x \in A'_\sigma \wedge f \cdot x{\downarrow} \;\; \Rightarrow \;\; f \cdot x \in A'_\tau.$$

(ii) A *substructure* T' of a type structure T consists of a family of subsets $T'_\sigma \subseteq T_\sigma$, one for each σ, such that $T'_0 = T_0 = \mathbb{N}$ and T' is closed under application in the evident sense.

(iii) If A' is a sub-TPCA of A, we define $\mathsf{EC}(A; A')$, the *relative extensional collapse* of A and A', to be the substructure of $\mathsf{EC}(A)$ consisting of those elements t that are realized by at least one element of A'.

In fact, $\mathsf{EC}(A; A')$ corresponds precisely to the type structure over the natural numbers in the *relative realizability* model $\mathbf{PER}(A; A')$ (see e.g. [1]).

Note that the substructure $\mathsf{EC}(A; A')$ need not be (even isomorphic to) a type structure in its own right, since it need not be *extensional*. That is, there could be two elements $f, g \in \mathsf{EC}(A; A')_{\sigma\to\tau}$ which are distinct as functions

$\mathsf{EC}(A)_\sigma \to \mathsf{EC}(A)_\tau$, but which restrict to the same function $\mathsf{EC}(A;A')_\sigma \to \mathsf{EC}(A;A')_\tau$. However, in the examples of interest in this paper, $\mathsf{EC}(A;A')$ will be the substructure RC of C, which is in fact extensional and so can itself be regarded as a standalone type structure. This is a consequence of the Kleene-Kreisel *density theorem* — see [12] or, for full details, [13].

As a trivial point, note that $\mathsf{EC}(A;A) = \mathsf{EC}(A)$.

3 Examples of TPCAs

We now illustrate our general framework by means of some examples. In Sections 3.1–3.3 we consider various classes of TPCAs which give some indication of the scope of the notion. In Section 3.4 we review some of the known results concerning the extensional collapses of these models.

3.1 Untyped PCAs

Any ordinary (untyped) partial combinatory algebra can be viewed as a TPCA. Recall that a partial combinatory algebra (or PCA) consists of a set A equipped with a partial function $\cdot : A \times A \rightharpoonup A$, such that there exist elements $\mathbf{k}, \mathbf{s} \in A$ satisfying the first three conditions given in Definition 2.1. We may view any PCA A as a TPCA, by setting $A_\sigma = A$ and $\cdot_{\sigma,\tau} = \cdot$ for all σ, τ. In the untyped setting, a suitable recursor $\mathbf{y}$ and a system of numerals can be constructed from $\mathbf{k}, \mathbf{s}$ by well-known means (see e.g. [9, Chapter 1]).

Furthermore, we may say A' is a *sub-PCA* of A if $A' \subseteq A$ contains some $\mathbf{k}, \mathbf{s}$ suitable for A and is closed under application in the sense of Definition 2.5(i). Clearly, if A' is a sub-PCA of A, then viewing A, A' as TPCAs, A' is a sub-TPCA of A.

The following examples are particularly important for our purposes:

Example 3.1 *Kleene's first model*, denoted K_1. Here the underlying set is $\mathbb{N}$, and by definition $m \cdot n \simeq \phi_m(n)$, where $\phi_0, \phi_1, \ldots$ is some fixed effective enumeration of the partial recursive functions.

Example 3.2 *Kleene's second model*, denoted K_2. Here the underlying set is $\mathbb{N}^\mathbb{N}$, the set of all total functions $\mathbb{N} \to \mathbb{N}$. To define application in K_2, we need some auxiliary notation. First, let $\langle \cdots \rangle$ be some fixed effective coding of finite sequences of natural numbers as natural numbers, and for any $g \in \mathbb{N}^\mathbb{N}$, let $\tilde{g} \in \mathbb{N}^\mathbb{N}$ be the *course-of-values* function defined by

$$\tilde{g}(n) \;=\; \langle g(0), \ldots, g(n-1) \rangle.$$

Next, define a partial function $| : \mathbb{N}^\mathbb{N} \times \mathbb{N}^\mathbb{N} \rightharpoonup \mathbb{N}$ by

$$f \mid g \;\simeq\; f(\tilde{g}(r)), \text{where } r \text{ is the least number s.t. } f(\tilde{g}(r)) > 0.$$

(The motivation for this definition is explained in [12, Section 3.3].) We also define functions g_m (for $m \in \mathbb{N}$) by $g_m(0) = m$, $g_m(n+1) = g(n)$. Thus, from any f, g we may obtain a *partial* function $\Lambda m.\ f \mid g_m : \mathbb{N} \rightharpoonup \mathbb{N}$. The application operation for K_2 is now defined by

$$f \cdot g \ = \ \begin{cases} \Lambda m.\ f \mid g_m \text{ if this is a total function,} \\ \text{undefined} \quad \text{otherwise.} \end{cases}$$

Notice that the functions $\mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}}$ that are representable in K_2 (i.e. are of the form $\Lambda g.\ f \cdot g$ for some $f \in K_2$) are precisely the *continuous* functions with respect to the usual Baire topology on $\mathbb{N}^{\mathbb{N}}$.

Moreover, by restricting to the total *recursive* functions $\mathbb{N} \to \mathbb{N}$, we obtain a sub-PCA K_2^{eff} of K_2. (In this paper we will uniformly use the superscript ${}^{\mathit{eff}}$ to denote an effective substructure of some given structure.)

Other PCAs, whose definition will not be required in this paper, include: Scott's *graph model* $\mathcal{P}\omega$; Plotkin's $\mathbb{T}^{\omega}$; van Oosten's $\mathcal{B}$; the effective sub-PCAs of all these; and various term models of untyped lambda calculi, such as Λ^0/β, the set of closed lambda terms modulo β-equivalence. Several of these PCAs are discussed e.g. in [10].

3.2 CCC models

Many of the cartesian closed categories considered in denotational semantics naturally give rise to TPCAs. The general idea is as follows: we start with a cartesian closed category $\mathcal{C}$, within which we can find an object $\bar{N}$ with a canonical inclusion $\mathbb{N} \hookrightarrow \mathrm{Hom}(1, \bar{N})$. We may then interpret our types σ by objects $[\![\, \sigma \,]\!]$ as follows:

$$[\![\bar{0}]\!] \ = \ \bar{N}, \qquad [\![\, \sigma \to \tau \,]\!] \ = \ [\![\, \tau \,]\!]^{[\![\, \sigma \,]\!]}.$$

Now define a structure A by taking $A_\sigma = \mathrm{Hom}(1, [\![\, \sigma \,]\!])$ for each σ, with $\cdot_{\sigma\tau}$ the function induced by the evaluation morphism $[\![\, \sigma \to \tau \,]\!] \times [\![\, \sigma \,]\!] \to [\![\, \tau \,]\!]$. The existence of suitable $\mathbf{k}, \mathbf{s}$ follows from the fact that $\mathcal{C}$ is cartesian closed, and the numerals $\widehat{n}$ are given by our canonical inclusion. Furthermore, if $\mathcal{C}$ contains successor, recursor and fixed point morphisms in an obvious sense, then A will contain suitable elements $\mathbf{suc}, \mathbf{rec}, \mathbf{y}$ and will hence be a TPCA.

Intuitively, these conditions will hold in any category that is a "model of PCF" (see Section 3.3 below). This includes many categories of domains based on order-theoretic notions (e.g. Scott domains, stable domains), as well as more intensional categories such as the *sequential algorithms* model of Berry and Curien, and many of the categories of games considered by Abramsky, Hyland *et al.* Typically, in all these models, we take $\bar{N}$ to be the object playing the role of the domain $N_\perp$. Furthermore, most of these models have obvious "effective submodels" which give rise to sub-TPCAs.

As in [12], we will write P for the TPCA obtained in this way from the standard Scott model (consisting of the partial continuous functionals over $N_\perp$), and $\mathsf{P}^{\mathit{eff}}$ for its effective submodel. We will also write L for the TPCA arising from the category of *continuous lattices* (where we take $\bar{N}$ to be the "flat" lattice of natural numbers with a bottom and top element), and $\mathsf{L}^{\mathit{eff}}$ for the effective submodel of L.

3.3 Syntactic models

Another class of TPCAs are those arising as term models for programming languages, particularly extensions of Plotkin's PCF. As in [12], let us define (combinatory) PCF to be the language consisting of the well-typed expressions built up via application from the constants

$$\begin{aligned}
\mathtt{K}_{\sigma\tau} &: \sigma \to \tau \to \sigma \\
\mathtt{S}_{\rho\sigma\tau} &: (\rho \to \sigma \to \tau) \to (\rho \to \sigma) \to (\rho \to \tau) \\
\mathtt{Y}_{\sigma} &: (\sigma \to \sigma) \to \sigma \\
\mathtt{if} &: \overline{0} \to \sigma \to \sigma \\
\mathtt{0} &: \overline{0} \\
\mathtt{suc} &: \overline{0} \to \overline{0} \\
\mathtt{pre} &: \overline{0} \to \overline{0}
\end{aligned}$$

We write $\widehat{k}$ as an abbreviation for the term $\mathtt{suc}^k\mathtt{0}$, and define the reduction relation $\rightsquigarrow$ for PCF to be the smallest transitive relation on PCF terms satisfying the following clauses, in which we assume all terms are well-typed:

- $\mathtt{K}\,UV \rightsquigarrow U, \qquad \mathtt{S}\,UVW \rightsquigarrow (UW)(VW), \qquad \mathtt{Y}\,U \rightsquigarrow U(\mathtt{Y}\,U).$
- $\mathtt{pre}\ \mathtt{0} \rightsquigarrow \mathtt{0}, \qquad \mathtt{pre}\ \widehat{k+1} \rightsquigarrow \widehat{k}.$
- $\mathtt{if}\ \mathtt{0}\,U\,V \rightsquigarrow U, \qquad \mathtt{if}\ \widehat{k+1}\,U\,V \rightsquigarrow V.$
- If $U \rightsquigarrow U'$ then $UV \rightsquigarrow U'V$.
- If $U \rightsquigarrow U' : \overline{0}$ then $c\,U \rightsquigarrow c\,U'$ for $c = \mathtt{suc}, \mathtt{pre}, \mathtt{if}$.

Suppose now that $\mathcal{L}$ is any language obtained from PCF by extending the definition with (finitely many) new typed constants and reduction rules: for example, PCF with "parallel-or", or PCF with a non-functional operator such as "catch". Let $\sim$ be any congruence on terms of $\mathcal{L}$; that is, an equivalence relation which contains $\rightsquigarrow$ and satisfies

$$M \sim M' \wedge N \sim N' \;\Rightarrow\; MN \sim M'N'$$

We may define a TPCA $\mathcal{L}/{\sim}$, called the *term model of $\mathcal{L}$ modulo $\sim$*, by taking $(\mathcal{L}/{\sim})_\sigma$ to be the set of $\mathcal{L}$-terms modulo $\sim$; $\cdot_{\sigma\tau}$ to be the function induced by application of $\mathcal{L}$-terms; and $\mathbf{k}, \mathbf{s}, \mathbf{y}, \widehat{n}, \mathbf{suc}$ to be the equivalence classes of $\mathtt{K}, \mathtt{S}, \mathtt{Y}, \widehat{n}, \mathtt{suc}$. It is a simple exercise to write a suitable PCF program for $\mathbf{rec}$.

Note that the smallest interesting congruence here will be the one generated by $\rightsquigarrow$, and the largest will be the one corresponding to *observational equivalence* of $\mathcal{L}$-terms.

3.4 Known constructions of particular type structures

We next review some known results concerning the total type structures obtained as the extensional collapse of the above models. These will give us some possible definitions of our three type structures of interest. We refer the reader to [12] for further details of relevant equivalence results.

The type structure C of *continuous functionals* may be obtained in any of the following ways:

- $\mathsf{C} = \mathsf{EC}(K_2)$. This is essentially Kleene's original definition of C via *associates* (see e.g. [12, Section 3.3]). Modulo unimportant details, an associate for a functional is essentially a realizer in K_2.
- $\mathsf{C} = \mathsf{EC}(\mathsf{P})$. This characterization, due to Ershov, can be seen as a cleaned-up version of Kreisel's original definition, and is the construction that is usually favoured as the definition of C in the more recent literature (see e.g. [13]). The equivalence between this and the K_2 characterization is non-trivial. Since all the relevant domains P_σ are known to be retracts of Plotkin's $\mathbb{T}^\omega$, it follows easily from this characterization that also $\mathsf{C} = \mathsf{EC}(\mathbb{T}^\omega)$.
- $\mathsf{C} = \mathsf{EC}(\mathsf{L})$. This is due to Scott, and is quite close to Ershov's characterization. Since all continuous lattices are known to be retracts of Scott's $\mathcal{P}\omega$, it follows easily that $\mathsf{C} = \mathsf{EC}(\mathcal{P}\omega)$.
- Most of the other known constructions of C are of a topological or semi-topological character: for instance, one obtains C as the type structure over $\mathbb{N}$ in the cartesian closed category of compactly generated Hausdorff spaces, or of filter spaces, limit spaces, etc. These characterizations do not fit into the realizability framework considered here.

Many of the above constructions can be effectivized in a natural way, and hence allow us to define an "effective" substructure of C. In fact, all reasonable effectivizations seem to lead to the same effective substructure $\mathsf{RC} \subset \mathsf{C}$ of *recursive continuous functionals*. Thus:

- $\mathsf{RC} = \mathsf{EC}(K_2; K_2^{\mathit{eff}})$.
- $\mathsf{RC} = \mathsf{EC}(\mathsf{P}; \mathsf{P}^{\mathit{eff}}) = \mathsf{EC}(\mathbb{T}^\omega; \mathbb{T}^{\omega\,\mathit{eff}})$.
- $\mathsf{RC} = \mathsf{EC}(\mathsf{L}; \mathsf{L}^{\mathit{eff}}) = \mathsf{EC}(\mathcal{P}\omega; \mathcal{P}\omega^{\mathit{eff}})$.
- A few of the topological models, such as the category of filter spaces, have a natural effective submodel, and these too give rise to RC.

Now suppose that for each of these models we consider the effective submodel as a TPCA in its own right, so that we are only considering "effective operations acting on effective data". In this case, the extensional collapse construction yields a third class of functionals: the type structure HRC of *hereditarily recursive continuous functionals*. Here, a purely topological characterization is not available, but instead we have an alternative realizability characterization.

- $\mathsf{HRC} = \mathsf{EC}(K_2^{\mathit{eff}})$.
- $\mathsf{HRC} = \mathsf{EC}(\mathsf{P}^{\mathit{eff}}) = \mathsf{EC}(\mathbb{T}^{\omega\,\mathit{eff}})$ (Ershov).
- $\mathsf{HRC} = \mathsf{EC}(\mathsf{L}^{\mathit{eff}}) = \mathsf{EC}(\mathcal{P}\omega^{\mathit{eff}})$ (Scott).
- Remarkably, HRC coincides with the type structure HEO of *hereditarily effective operations*, defined as $\mathsf{EC}(K_1)$. This highly non-trivial result is known as the *generalized Kreisel-Lacombe-Shoenfield (KLS) theorem.*[4] It was noted by Kreisel in [7], but the first published proof appeared in [16]. A proof of the result in the form in which we have stated it appears in [3].

We will henceforth use the name HEO for this type structure, as it is somewhat more familiar than HRC. In addition, the characterization via K_1 plays an important role in the proof of our main result.

As noted in the Introduction, HEO is "incomparable" with RC and indeed with C. For instance, the *Kleene tree functional* is present in HEO_2 but has no counterpart in RC or C, whereas the *fan functional* in RC_3 has no counterpart in HEO. We refer the reader to [12] for further details.

The above is not an exhaustive list of the known characterizations of our type structures, and we shall have occasion to mention a few others below.

4 Continuous and effective TPCAs

The pattern that emerges from the foregoing results is that the type structure obtained from an extensional collapse construction depends solely on whether the TPCA or TPCAs in question are "full continuous" or "effective". This leads us to look for precise notions of *full continuous* TPCA and *effective* TPCA that will allow us to obtain theorems of roughly the following kind:

- If A is a full continuous TPCA, then $\mathsf{EC}(A) = \mathsf{C}$.
- If A is a full continuous TPCA and A' an effective sub-TPCA of A, then $\mathsf{EC}(A; A') = \mathsf{RC}$.
- If A is an effective TPCA then $\mathsf{EC}(A) = \mathsf{HEO}$.

We now propose some such definitions, which seem to us to capture the intuitive notions we have in mind, and introduce a little more machinery relevant to these. This will allow us, in Section 4.3, to give precise statements of our main results, together with a brief overview of the method of proof. In Section 4.4 we will mention some specific applications of our results.

4.1 Realizations of TPCAs

We will start with the following general notion:

[4] The ordinary KLS theorem essentially states that $\mathsf{HRC}_2 = \mathsf{HEO}_2$.

Definition 4.1 Suppose A is a TPCA, B an untyped PCA, and suppose $\widehat{0}, \widehat{1}, \ldots$ is a suitable system of numerals in A, and $\widetilde{0}, \widetilde{1}, \ldots$ a system of numerals in B.

A *realization of A over B* consists of a family of relations $\Vdash_\sigma \subseteq B \times A_\sigma$ (one for each σ) with the following properties:

- For all $a \in A_\sigma$ there exists b such that $b \Vdash_\sigma a$.
- For each σ, τ, there is an element $\alpha_{\sigma\tau} \in B$ such that for all $b, b' \in B$, $a \in A_{\sigma \to \tau}$, $a' \in A_\sigma$,
$$b \Vdash_{\sigma\to\tau} a \;\wedge\; b' \Vdash_\sigma a' \;\wedge\; a \cdot a' \downarrow \;\Rightarrow\; \alpha_{\sigma\tau} \cdot b \cdot b' \Vdash_\tau a \cdot a'.$$
(We may think of this as saying "*$\alpha_{\sigma\tau}$ tracks $\cdot_{\sigma\tau}$*".)
- There is an element $\chi \in B$ such that for all $m, n \in \mathbb{N}$,
$$m \Vdash_0 \widehat{n} \;\Rightarrow\; \chi \cdot m = \widetilde{n}.$$

We say A is *realizable over B* if there exists a realization of A over B.

In fact, this definition can be phrased more concisely using the terminology of [11]: a realization of A over B is an applicative morphism $\gamma : A \longrightarrow\!\!\!\!\!\rightarrow B$ such that $\gamma\delta \preceq \epsilon$, where δ, ϵ are the canonical applicative morphisms from K_1 to A, B respectively. The definition can also be generalized further, allowing B to be an arbitrary TPCA, though this will not be necessary for our present purposes.

The condition involving χ ensures that the representation of numerals in A is in some sense "standard". To see why this condition is important, consider the situation where $A = K_1^T$ for T some non-trivial Turing degree, and $B = K_1$. Intuitively, we do not expect A to be realizable over B. However, it turns out that there is an applicative morphism $K_1^T \longrightarrow\!\!\!\!\!\rightarrow K_1$ (see [9, Chapter 3]), so in some sense we can simulate non-effective computations in K_1. This is of no use in practice, though, because we have no way of "decoding" the result of the computation to obtain its value as a natural number; this deficiency corresponds to the absence of a suitable element χ.

The following facts are easily checked:

Proposition 4.2 *(i) If A is realizable over B and B is realizable over C (where A is a TPCA and B, C are PCAs), then A is realizable over C.*

(ii) K_1 (considered as a TPCA) is realizable over any PCA B.

We now propose the following definition:

Definition 4.3 A TPCA is *effective* if it is realizable over K_1.

In view of the foregoing proposition, the notion of an effective TPCA is quite robust: for example, if B is any PCA realizable over K_1 (such as K_2^{eff}), then a TPCA is effective iff it is realizable over B. It is easy to check that all the "effective models" mentioned earlier — $\mathsf{P}^{\mathit{eff}}, \mathsf{L}^{\mathit{eff}}, \mathbb{T}^{\omega\,\mathit{eff}}, \mathcal{P}\omega^{\mathit{eff}}, K_2^{\mathit{eff}}$ and of course K_1 itself — are effective TPCAs. In addition, any term model of the kind described in Section 3.3 is an effective TPCA, since we may give a realizability relation by means of a Gödel-numbering of terms.

It is perhaps not immediately clear what the notion of a continuous TPCA should be. We propose the following:

Definition 4.4 A TPCA is *continuous* if it is realizable over K_2.

A little experimentation reveals that all the familiar examples of "full continuous PCAs" (e.g. $\mathcal{P}\omega, \mathbb{T}^\omega, \mathcal{B}$) are realizable over K_2, and K_2 is itself realizable over some of these, so this notion of continuous TPCA seems to be reasonably robust, and also covers all the examples we have in mind. The intuition is that in a continuous TPCA, each object embodies only a countable amount of information (and hence can be coded by an element of K_2); moreover all the relevant operations must act continuously with respect to this information content (and hence be realizable within K_2).

We say a continuous TPCA is *full continuous* if, intuitively, it contains all set-theoretic functions $\mathbb{N} \to \mathbb{N}$ in a "standard" way. More precisely:

Definition 4.5 Suppose A is a continuous TPCA, with $\Vdash_\sigma$ a realization over K_2. We say A is *full continuous* if

- for every $f : \mathbb{N} \to \mathbb{N}$, there exists $a \in A_1$ such that for all $n \in \mathbb{N}$, $a \cdot \widehat{n} = \widehat{f(n)}$.
- moreover a realizer for some such element a may be computed from f within K_2. More precisely, there is an element $\psi \in K_2$ such that for all $f \in \mathbb{N}^\mathbb{N}$, we have $\psi \cdot f \Vdash_1 a$ for some $a \in A_1$ satisfying the foregoing condition, where $\cdot$ denotes application in K_2.

This condition can also be expressed in terms of applicative morphisms, though it is not quite trivial to do so. The second half of the condition ensures that the first half holds in a suitable "constructive" sense; this appears to be essential for the proofs of our theorems.

We also need the notion of an effective sub-TPCA of a full continuous TPCA. Recall here that we have a sub-PCA K_2^{eff} of K_2, whose carrier set is the set of total recursive functions.

Definition 4.6 Suppose A is a full continuous TPCA, with $\Vdash_\sigma, \alpha_{\sigma\tau}, \chi, \psi$ as in Definitions 4.1 and 4.5, and A' is a sub-TPCA of A. We say A' is an *effective sub-TPCA* of A if

- for all $a \in A'_\sigma$, there exists $f \in K_2^{\mathit{eff}}$ with $f \Vdash_\sigma a$;
- for all σ, τ, the elements $\alpha_{\sigma\tau}, \chi, \psi$ are in K_2^{eff}.

Since K_2^{eff} is itself an effective PCA, it is easy to see that if A' is an effective sub-TPCA of A then A' is an effective TPCA in its own right.

4.2 Realizations of type structures

We will also need the notion of a realization of a type structure over a PCA. This is an obvious adaptation of Definition 4.1; indeed, in the more general framework of applicative morphisms, we can make do with a single notion of realization.

Definition 4.7 Let T be a type structure, B be a PCA. A *realization* of T over B is a family of relations $\Vdash_\sigma \subseteq B \times T_\sigma$ such that

- for all $x \in T_\sigma$ there exists some $b \Vdash_\sigma x$;
- For each σ, τ there is an element $\theta_{\sigma\tau} \in B$ such that for all $b, b' \in B$, $f \in T_{\sigma\to\tau}$, $x \in T_\sigma$, we have
$$b \Vdash_{\sigma\to\tau} a \;\wedge\; b' \Vdash_\sigma a' \;\wedge\; a \cdot a' \downarrow \;\Rightarrow\; \theta_{\sigma\tau} \cdot b \cdot b' \Vdash_\tau a \cdot a'.$$

Finally, we will need a notion of when two realizations of a type structure are *isomorphic*. Intuitively, this is so when suitable translations between the realizations are computable within the PCA in question. (Again, this is simply an instance of a general concept of applicative isomorphism in the framework of [11].)

Definition 4.8 If $\Vdash$ and $\Vdash'$ are both realizations of T over B, we say $\Vdash$, $\Vdash'$ are *isomorphic* if for all σ there exist u_σ, v_σ such that for all $x \in T_\sigma$ and $b, b' \in B$ we have
$$b \Vdash x \;\Rightarrow\; u_\sigma \cdot b \Vdash' x, \qquad b' \Vdash' x \;\Rightarrow\; v_\sigma \cdot b' \Vdash x.$$
If moreover u_σ, v_σ belong to some sub-PCA B' of B, we say $\Vdash$, $\Vdash'$ are *isomorphic relative to B'*.

Clearly, a realization $\Vdash$ of a TPCA A over B induces a realization $\Vdash_*$ of $\mathsf{EC}(A)$ over B: using the notation of Definition 2.2, we define $b \Vdash_{*\sigma} x$ if there exists $a \in A_\sigma$ with $b \Vdash_\sigma a$ and $\beta[a] = x$. Likewise, for any sub-TPCA A' of A, a realization of A over B induces a realization of $\mathsf{EC}(A; A')$ over B.

Our three type structures have realizations arising from their definitions, which we may regard as the "standard" ones:

Definition 4.9 (i) The *standard realization of* C is the realization of $\mathsf{C} = \mathsf{EC}(K_2)$ over K_2 induced by the identity realization on K_2.

(ii) The *standard realization of* RC is the realization of $\mathsf{RC} = \mathsf{EC}(K_2; K_2^{\mathit{eff}})$ over K_2 induced by the identity.

(iii) The *standard realization of* HEO is the realization of $\mathsf{HEO} = \mathsf{EC}(K_1)$ over K_1 induced by the identity.

4.3 The main theorems

We are now at last able to give precise statements of our main results. The theorems for C and RC can be stated just as we would like them:

Theorem 4.10 (Ubiquity of C) *Let A be any full continuous TPCA. Then $\mathsf{EC}(A) = \mathsf{C}$; moreover, the induced realization of $\mathsf{EC}(A)$ is isomorphic to the standard realization of C.*

Theorem 4.11 (Ubiquity of RC) *Let A be any full continuous TPCA, and A' an effective sub-TPCA of A. Then $\mathsf{EC}(A;A') = \mathsf{RC}$; moreover, the induced realization of $\mathsf{EC}(A;A')$ is isomorphic relative to K_2^{eff} to the standard realization of RC.*

The third theorem is considerably more delicate than the others, and at present we able to prove it only under some slightly technical hygiene conditions:

Theorem 4.12 (Ubiquity of HEO) *Let A be any effective TPCA satisfying the following conditions (where $\Vdash_\sigma, \alpha_{\sigma\tau}$ are as in Definition 4.1):*

(A) *Each $\alpha_{\sigma\tau}$ satisfies $m \Vdash_{\sigma\tau} a \;\wedge\; n \Vdash_\sigma b \;\wedge\; \alpha_{\sigma\tau}(m,n) \Vdash_\tau c \;\Rightarrow\; a \cdot b = c$.*

(B) *The relation $m \Vdash_0 \widehat{n}$ is r.e. in m, n.*

Then $\mathsf{EC}(A) = \mathsf{HEO}$; moreover, the induced realization of $\mathsf{EC}(A)$ is isomorphic to the standard realization of HEO.

Condition (A) here is very mild, and in all naturally arising examples that we know can be arranged to hold by choosing the functions $\alpha_{\sigma\tau}$ suitably. Condition (B) holds in most known models, but unfortunately rules out a few important models such as $\mathcal{P}\omega^{\mathit{eff}}$ and K_2^{eff} (though in these cases the conclusion is known to hold anyway), as well as the effective Nakajima tree model of the untyped lambda calculus. We are very hopeful that it may be possible to weaken (B) to “the relation $m \Vdash_0 \widehat{n}$ is Π^0_2”, which holds in all the models of interest that we are aware of. It appears, however, that our method of proof will not allow us to dispense with either (A) or (B) completely.

Our proofs of the above theorems are long and complex, and make essential use of a technique developed by Normann in [14]. We will briefly outline the method of proof here, taking the case of HEO as an example. Full details of the proofs will appear in an extended version of this paper.

Normann showed that in the extensional collapse of $\mathsf{P}^{\mathit{eff}}$, every equivalence class contains at least one PCF-definable element of $\mathsf{P}^{\mathit{eff}}$. From this, it is not too hard to show that if Q is the term model for PCF (modulo any reasonable congruence), then $\mathsf{EC}(\mathsf{Q}) = \mathsf{EC}(\mathsf{P}^{\mathit{eff}}) = \mathsf{HEO}$, and moreover all these realizations of HEO are isomorphic to the standard one. We next note that PCF can be in some sense “interpreted” in any TPCA A whatever — this is clear enough from the similarity between the definition of TPCA and that of PCF. (In the parlance of denotational semantics, this interpretation of PCF will be *sound* but not in general *adequate*.) This amounts to saying that Q is realizable over A. In the situation of the Theorem 4.12, we are considering an effective TPCA A. That is, we have realizations

$$\mathsf{Q} \longrightarrow\!\!\!\!\!\!\!+\; A \longrightarrow\!\!\!\!\!\!\!+\; K_1$$

where $\mathsf{EC}(\mathsf{Q}) = \mathsf{EC}(K_1) = \mathsf{HEO}$. Here one may naturally think of Q as the "weakest" effective TPCA and K_1 as the "strongest", with A sandwiched between them.

This already makes it look plausible that $\mathsf{EC}(A) = \mathsf{HEO}$, but in fact the proof is still very difficult, involving an elaborate induction up the types. At each type level σ, the crux is to show that each $F \in \mathsf{HEO}_\sigma$ is present in $\mathsf{EC}(A)_\sigma$ in a suitable way. We do this by constructing a PCF term M that "computes" F, and showing that the image of M in A is a suitable realizer for F. Here the technique for constructing M comes from Normann's proof. Essentially, Normann considered the above situation in the case $A = \mathsf{P}^{\mathit{eff}}$, and showed by an ingenious and beautiful argument how to construct a PCF term for a given $F \in \mathsf{HEO}_\sigma$ in this case. The core of our task is to adapt Normann's method so that it works for a general effective PCA A. This requires various refinements and some delicate recursion-theoretic arguments, since certain useful features of $\mathsf{P}^{\mathit{eff}}$ such as monotonicity are not available to us in the general setting. Both the generalized KLS theorem and the Myhill-Shepherdson theorem play an important role in these arguments.

4.4 Applications of our results

As the range of examples of TPCAs in Section 3 makes clear, a large number of new characterizations of our type structures flow from our theorems. Of course, many of these are of no obvious interest and answer questions that no one has ever asked. However, in the case of TPCAs which themselves embody seemingly natural "notions of computability", our theorems do provide specific results which are of relevance for our general project of mapping out such notions and the relationships between them. We would particularly single out the van Oosten algebra $\mathcal{B}$ (a full continuous PCA) and its effective submodel, and also the type structure R of sequentially realizable functionals and its effective substructure — see [12]. Direct proofs of our results for these structures would probably be feasible but rather tiresome.

Perhaps most striking is the application of our results to syntactic models, such as term models for the untyped lambda calculus. The problem of identifying the type structures arising from such models was explicitly raised by Beeson in [2], and has occasionally been mentioned in passing as an open problem since then (see e.g. [9, page 250]). Until recently, the author believed that this problem was probably intractable at present (as would seem to be the case for the problem of identifying the *partial* type structures arising from these models). However, we can now see immediately that $\mathsf{EC}(\Lambda^0/\beta) = \mathsf{HEO}$ for any consistent theory T.

It is also worth pointing out that our results apply even to such unwieldy structures as term models for full-scale programming languages. Consider, for example, the set of closed programs of Standard ML of simple type (for instance, with respect to the initial dynamic basis). Under the congruence generated by the evaluation relation, these clearly form an effective TPCA, and so without knowing any more about the definition of ML, one can conclude (along the lines described in Section 1.1) that the hereditarily total functionals computable in ML are precisely those of HEO. A version of this result holds even in the presence of global state variables, though its formulation requires a little extra care.

5 Conclusion

We have shown that some large classes of "standard realizability" constructions all give rise to one of the well-known type structures $\mathsf{C}, \mathsf{RC}, \mathsf{HEO}$. As we have observed, even for individual TPCAs such results tend to have a non-trivial character, and so the possibility of obtaining such general results comes as a considerable surprise. Since our classes include a large number of mathematically natural constructions (as well as an even larger number of unnatural ones), we regard our results as providing strong evidence that these type structures are highly canonical mathematical objects. This is conceptually very reassuring since, at least in the author's view, it is usually rather difficult to convince oneself that any *single* extensional collapse construction, considered in isolation, defines a mathematically natural type structure: one is typically left with a nagging concern that the type structure obtained might be unreasonably sensitive to the kind of realizers involved.

We venture to suggest that our theorems in some way represent a new kind of result in the field of higher type computability. Whereas many of the main results in this field to date have been of the form that two *particular* constructions yield the same type structure, our results are of a rather more sweeping kind: *any* construction in some interesting (axiomatically defined) class gives rise to a certain type structure. Intuitively, rather than just considering a few isolated points or landmarks in the "space of notions of computability", we are able to take care of a whole tract of territory in a single swoop.

We feel that the possibility of proving such general results is an encouraging sign for our overall project of mapping out the higher type computability landscape, since they allow us to progress in much bigger strides, and save us from the need (or temptation) to prove an endless stream of individual equivalence results. It would be interesting to know whether any results in a similar vein can be obtained for *partial* type structures (see [12, Section 4]).

Although our results are quite general, one shortcoming is that they do not at present apply to *modified realizability* constructions. In the modified realizability construction on a TPCA A, one first picks out a substructure A' of "hereditarily total" elements of A, and then applies the standard extensional collapse to A'. The problem is that A' may not be a TPCA in the sense of the present paper, since it will usually not possess $\mathbf{y}$ combinators.

(In certain circumstances it is trivial to show that the standard and modified constructions agree, but this is not so in general.) Nevertheless, modified realizability constructions are very much in the same ballpark as the constructions we have considered, so one might reasonably hope that our theorems could be extended to cover them. Further support for this is provided by a striking and non-trivial result of Bezem [4], who shows that the modified extensional collapse of K_1 is also HEO, and likewise the modified extensional collapse of K_2 is C.

Extending our theorems to include modified realizability would in our view give a better sense of completion to our results, but at present we cannot see how to achieve this. A closer study of Bezem's proofs might well yield some useful insights here.

References

[1] Awodey, S., L. Birkedal, and D.S. Scott, *Local realizability toposes and a modal logic for computability*, Mathematical Structures in Computer Science **12** (2002), 319–334.

[2] Beeson, M., "Foundations of Constructive Mathematics," Springer-Verlag, 1985.

[3] Berger, U., *Total sets and objects in domain theory*, Annals of Pure and Applied Logic **60** (1993), 91–117.

[4] Bezem, M., *Isomorphisms between HEO and HRO^E, ECF and ICF^E*, Journal of Symbolic Logic **50** (1985), 359–371.

[5] Kleene, S.C., *Countable functionals*, in Heyting, A., editor, "Constructivity in Mathematics: Proceedings of the Colloquium held at Amsterdam, 1957", North-Holland (1959), 81–100.

[6] Kleene, S.C., *Recursive functionals and quantifiers of finite types I*, Transactions of the American Mathematical Society **91** (1959), 1–52.

[7] Kreisel, G., *Interpretation of analysis by means of functionals of finite type*, in Heyting, A., editor, "Constructivity in Mathematics: Proceedings of the Colloquium held at Amsterdam, 1957", North-Holland (1959), 101–128.

[8] Lietz, P., and T. Streicher, *Impredicativity entails untypedness*, Mathematical Structures in Computer Science **12** (2002), 335–347.

[9] Longley, J.R., "Realizability Toposes and Language Semantics," Ph.D. thesis, University of Edinburgh, 1995. Available as ECS-LFCS-95-332.

[10] Longley, J.R., *Matching typed and untyped realizability*, Electronic Notes in Theoretical Computer Science **23(1)** (1999).

[11] Longley, J.R., *Unifying typed and untyped realizability*, Unpublished note, 1999. Available at `http://www.inf.ed.ac.uk/home/jrl/unifying.txt`.

[12] Longley, J.R., *Notions of computability at higher types I*, to appear in "Logic Colloquium 2000, Proceedings", 109pp.

[13] Normann, D., *The continuous functionals*, in Griffor, E.R., editor, "Handbook of Computability Theory", North-Holland (1999), 251–275.

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

[15] Plotkin, G.D., *Full abstraction, totality and PCF*, Mathematical Structures in Computer Science **9(1)** (1997), 1–20.

[16] Troelstra, A.S., "Metamathematical Investigation of Intuitionistic Arithmetic and Analysis," Lecture Notes in Mathematics **344**, Springer-Verlag, 1973.