Universiteit Utrecht

DEPARTMENT OF MATHEMATICS

MASTER'S THESIS

Code-free Recursion & Realizability

Author:
Eric Faber (3365026)

Supervisor:
Dr. Jaap van Oosten

Second examiner:
Dr. Wouter Swierstra

June 10, 2014

Contents

Abstract

This thesis is an elaborate account of the theory of partial combinatory algebras (pcas) and their associated categorical structures called categories of assemblies and realizability toposes. From the viewpoint of "abstract Turing machines", we build up the theory of pcas, generalizing some constructions from ordinary recursion theory, such as relative computability in an oracle (of type 1 and higher). In later chapters, we show how this notion of generalized relative computability can be used to study realizability toposes, with special attention to the effective topos. We also treat geometric morphisms between realizability toposes, and fill in a gap in the theory of applicative morphisms (morphisms between pcas) in relation to geometric morphisms.

Chapter 1

Introduction

This thesis contains a detailed account of the research that I have been doing for the past year as part of my Master's programme in Mathematical Sciences. In one sentence, I have studied certain categorical structures that are associated to fundamental notions of computation. This short chapter serves as a brief introduction to the material.

Mathematical notions of computation, or rather of computability, were first studied in a field called *computability theory*, or *recursion theory*. It was founded around 1936 by work of Alonzo Church, Alan Turing, Stephen Kleene, Emil Post and Kurt Gödel. Independently, they proposed several mathematical *models of computation*, which were all found to produce the same set of *computable functions*. Consequently, they established the first *undecidability results*, by showing the existence of problems that "cannot be solved by mechanical computation".

Intuitively, their notion of a computation coincides with "abstractly running a computer program" written in a programming language like C or Python. With "abstractly", we mean that one should not worry about the amount of memory or any other physical constraint. Imagine such a computer program: it takes an input, runs some algorithm, and either it eventually produces an output or it keeps running forever. Both input and output are basically a string of 0's and 1's in the computer's memory, which is nothing more than just a number. Therefore, a computer program can be viewed as a *function* $\mathbb{N} \to \mathbb{N}$.

The set of those functions that can be computed by a computer program are called the *recursive*, or *computable* functions. A function need not be defined on every number (on a certain input, a program might loop forever), so such functions can also be partial. A result in computability theory defines this set of functions independently of any notion of computation. It is the smallest set that includes the basic arithmetical functions and is closed under composition, definition by recursion and unbounded search.

One can view the above as an arithmetical or numerical approach to defining a computable function. However, a lot of the theory of computable functions, *computability theory*, or *recursion theory*, is developed from another approach, which is rather combinatorical. Observe that computer programs are also just strings of 0's and 1's stored in memory, so basically numbers. We can view computer programs as numbers acting on numbers, yielding a number, or looping forever. Mathematically, this is just a partial map $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Therefore, computable functions "act on themselves", whence the name *recursion theory*. A good reference for classical recursion theory, which is worthwhile studying, is [Rog87].

To define the above partial maps, one first needs to establish a notion of computation. One may wonder whether we can provide axioms for them instead, yielding a recursion theory that is *code-free*. This is one of the motivations behind the notion of a *partial combinatory algebra* (pca) , which we will study in chapter 2. The definition of a pca is very simple, but very powerful: every non-trivial instance

of such a pca computes all classical recursive functions $\mathbb{N} \to \mathbb{N}$ for some suitable coding. The account presented here is based on the treatment in [vO08].

Several theorems and notions in classical recursion theory can be generalized to the theory of pcas. For example, we can describe relative recursion (section 2.6), and effective operations (section 2.7). In this thesis, we will also generalize a theory of recursive functionals due to Kleene to pcas (section 2.8.1).

We will also look at morphisms of pcas, and present a new definition of an important kind of morphism: that of a computationally dense morphism. It turns out that there is connection between this type of morphisms and relative recursion that we can describe in a process called *lifting* .

In chapter 3 we embark on more complicated structures that arise from pcas. There is a great similarity between programming language semantics and those structures. Anyone who has ever learned a programming language knows that data can have different *types*, for example one can deal with integers, floats, strings and even functions. A lot of programming languages admit the creation of new types. The compiler of such a language then deals with the translation of these data types to numbers on which we can perform computations and back. A mathematical analogue of this is the notion of an *assembly* . It consists of a set X, to be viewed as "elements of type X", and a total relation $E \subseteq X \times A$, where A is a partial combinatory algebra. So to each element of x, one or more elements of a pca are associated on which we can perform computations. They are "representatives" of x in A. A function $f : X \to Y$ that takes elements of type X and yields elements of type Y is then computable with respect to A if it is computable on the underlying representatives.

The set of all such assemblies, together with the "computable" morphisms, form a *category of assemblies* on A, which we will define in section 3.1. It turns out that this category has a very rich and interesting structure. Moreover, there is an interplay between morphisms of partial combinatory algebras and functors between the corresponding categories of assemblies. This is studied in section 3.3.

A category of assemblies can be extended to a *topos* by completing it in a universal way. Toposes are categories that provide enough constructions for "doing mathematics", like the category of sets. They are models of higher-order logic, and it is possible to define for example the real numbers, continuous functions, etc. Different toposes give rise to different such models. For example, there are toposes in which every function from the reals to the reals is continuous, but this is not true in every topos. It makes them interesting to study from a logical viewpoint. Toposes have a strong geometrical nature, so they establish a connection between geometry and logic in a very natural way (here [MLM94] is an excellent introduction).

In chapter 4, we will describe the toposes that arise from categories of assemblies, called *realizability toposes*. The first-order logic of realizability toposes coincides with *realizability* , which is an interpretation of constructive proofs based on partial combinatory algebras, and already described by Kleene for ordinary recursion theory. A good reference for most of the theory on realizability and the derived categorical-theoretic constructions is [vO08]. In this thesis, we give an extensive presentation of the construction of a realizability topos on a partial combinatory algebra, and state the main results on the connection between morphisms of partial combinatory algebras and morphisms between the corresponding toposes. A recent result by Peter Johnstone [Joh13] is also included in this account. Using our new definition of computationally dense morphisms, we can present these results in an elegant way.

We will describe the connection between relative recursion as defined in section 2.6 and the corresponding realizability toposes. The notion of *lifting* allows us to partially characterize geometric inclusions of certain realizability toposes (theorem 5.6.3), and especially subtoposes of the effective topos (corollary 5.6.4).

This research project started with a question about a set of $\mathbb{N}$-indexed functions defined in [vO], called $\mathcal{J}$-representable functions for a specific local operator $\mathcal{J}$. We will show in theorem 5.5.4 that

this set has a structure slightly weaker than that of a partial combinatory algebra. However, our theory of recursion in a type 2 functional allows us to prove that it is isomorphic to a partial combinatory algebra that happens to consist of precisely the *hyperarithmetical* functions (corollary 5.5.7).

This thesis is intended for someone with an interest in mathematical logic who already has some acquaintance with category theory and foundations of mathematics. I hope it serves as a first introduction to the theory of pcas, realizability and especially realizability toposes. Also, I hope that some of my own ideas and viewpoints provide a lead for further research.

Chapter 2

Partial Combinatory Algebras

In this chapter, we start by defining the notion of a *partial combinatory algebra* and state the main results that we will need. A lot of notation that is used in later sections is defined here.

One of the motivations for defining partial combinatory algebras is to axiomatize basic recursion theory to some extent, in a *code-free* way. We'll see that a lot of basic results, such as the recursion theorem, can be shown in a more general setting. However, we'll also see that a lot of the fine structure of ordinary recursion is not captured by a partial combinatory algebra.

In section 2.5, we will discuss a notion of morphism between pcas called an *applicative morphism*. Many properties of such morphisms that we will need in later chapters are introduced here. We also suggest a generalization of applicative morphisms that fills a gap in the existing theory. This will turn out to be useful in section 5.1.

Lastly we discuss a generalization of relative recursion for pcas in section 2.6. We will apply this to effective operations in section 2.7. In section 2.8, we apply it to define a notion of computable functionals for pcas, analogous to Kleene's computable functionals.

2.1 History

Partial combinatory algebras are first described by Moses Schönfinkel in [Sch24]. It was an attempt to resolve an issue in logic that arose from the use of *variables* and *substitution*. It is well-known that naive set theory gives rise to paradoxes (like Russel's paradox) that can only be resolved by looking closely at the very foundations of set theory, or mathematics in general, and providing strict rules (axioms) for the construction of sets.

In the same way, the "naive" use of variables and the concept of "substituting a variable by a value" gives rise to paradoxes and problems in logic. The solution of Moses Schönfinkel, and also of Alonzo Church with his $\lambda-$calculus, was basically to eliminate them by defining a certain pre-logic, consisting of terms that are combined using *combinators* that satisfy certain axioms. This leads to a very primitive idea of what a function is. Substitution of a variable is replaced by *application* to another function. Anyone familiar with a functional programming language might recognize this. There, "everything is a function" by default, and there are no variables in a certain sense.

Church's λ-calculus was invented around the time that his student, Alan Turing, defined the mathematical notion of a *computation* with what is now known as a Turing machine. Surprisingly, it turned out that any function that was computable in Turing's sense, was also computable in the $\lambda-$calculus. In de 50s and 60s, it was Haskell Curry and his students who combined this fact with the work of Moses Schönfinkel, giving birth to the field of *combinatory logic*. This ultimately led to the invention of functional programming languages (that nowadays become increasingly useful). One of the most famous of these is, historically sound, named *Haskell*.

For the rest of this thesis, we will view partial combinatory algebras as "generalized Turing machines", or very elementary programming languages. For more on combinatory logic and its origins, I refer to [CF68].

2.2 Partial Applicative Structures

We start by defining a *partial applicative structure*, analogous to the treatment in [vO08].

Definition 2.2.1. Let A be a set. A *partial applicative structure* on A is a partial function $A \times A \to A$, written $(a,b) \mapsto ab$. Whenever ab is defined, we say that ab *denotes*, or write $ab\downarrow$.

If A is endowed with a partial applicative structure, we can define a set of *terms* $T(A)$. For V an infinite set of variables, $T(A)$ is the smallest set such that:

- $A \cup V \subseteq T(A)$
- Whenever $t,s \in T(A)$, $(ts) \in T(A)$.

For t a term, write $\mathrm{FV}(t)$ for the set of free variables in t. Whenever we write $t(x_1,\dots,x_n)$ for t a term, we mean $\mathrm{FV}(t) \subseteq \{x_1,\dots,x_n\}$ and the substitution $t(a_1,\dots,a_n)$ has the obvious meaning and is a term.

Terms without free variables are called *closed* and can *denote* an element of A: for $t \in T(A)$, we write $t \downarrow a$ to express that t denotes the element $a \in A$. It is defined by:

- For all $a \in A$, $a \downarrow a$.
- For terms t,s, if $t \downarrow b$ and $t \downarrow c$, and $bc\downarrow$ in the applicative structure, then $(ts)\downarrow(bc)$.

Whenever there is an a such that $t \downarrow a$, we write $t\downarrow$ to say that the term t denotes. For a closed term t, we write $t\uparrow$ if t does not denote.

Terms without free variables are just compositions of the applicative map. To improve readability, I'll write down explicit terms using the convention of *associating to the left*:

$$abc := ((ab)c).$$

For terms t,s, we write $t \simeq s$ if $t\downarrow a$ iff $s \downarrow a$ for any $a \in A$. We write $t \precsim s$ if $s\downarrow a \Rightarrow t\downarrow a$. Also, whenever we write $t = s$ for closed terms t,s, we usually mean that there exists a such that $t\downarrow a$ and $s\downarrow a$.

Terms become interesting when we have combinatory completeness of the applicative structure.

Definition 2.2.2. A *partial combinatory algebra*, or *pca*, is a set A endowed with an applicative structure, such that there are $\mathsf{k},\mathsf{s} \in A$ with the following properties: for all $a,b,c \in A$:

(i) $\mathsf{s}ab\downarrow$

(ii) $\mathsf{k}ab \downarrow a$

(iii) $\mathsf{s}abc \simeq ac(bc)$.

When the above is the case, we say that the partial applicative structure is *combinatory complete* .

Examples

(i) Any singleton $\{*\}$ with the trivial applicative structure $** \downarrow *$ is a pca. One can show that any pca with $\mathsf{k} = \mathsf{s}$ is trivial in this sense.

(ii) Ordinary recursion theory gives an applicative structure on $\mathbb{N}$:

$$ab := \varphi_a(b)$$

where φ_a is the partial recursive function $\mathbb{N} \to \mathbb{N}$ with index $a \in \mathbb{N}$ for some model of computation. This pca is known as *Kleene's first model* and denoted by $\mathcal{K}_1$.

(iii) Any term model of untyped λ-calculus (modulo β-equality) yields a total pca.

(iv) There is a general method of constructing a total pca from a *directed complete partial order* (DCPO), which yields several examples. See [Hof] for a short description.

Note Partial combinatory algebras are also called "Schönfinkel Algebras" (at least by Peter Johnstone in [Joh13]), and in [Wag69] a theory on "Uniformly Reflexive Structures" is developed. The latter describes precisely the pcas that are *decidable*, which we will define below.

Every pca allows for abstraction, similar to abstraction of λ-terms in λ-calculus:

Lemma 2.2.3. *Let A be a pca. Then for any term $t(x, x_1, \ldots, x_n)$ there is a term $\langle x\rangle\, t(x_1, \ldots, x_n)$ such that every substitution instance of $\langle x\rangle\, t$ denotes, and for all $a, a_1, \ldots, a_n \in A$*

$$(\langle x\rangle\, t)(a_1, \ldots, a_n)a \simeq t(a, a_1, \ldots, a_n).$$

Conversely, if A has a partial applicative structure for which the above holds, then A is a pca.

Proof. We define $\langle x\rangle\, t$ according to the recursive definition of terms.

If t is a constant or a variable y different from x, let $\langle x\rangle\, t = \mathsf{k}t$.

If $t = x$, let $\langle x\rangle x = \mathsf{skk}$.

If $t = t_1 t_2$, let $\langle x\rangle\, t = \mathsf{s}(\langle x\rangle\, t_1)(\langle x\rangle\, t_2)$.

One can check that every substitution instance of $\langle x\rangle\, t$ denotes, and the fact that

$$(\langle x\rangle)\, t(a_1, \ldots, a_n)a \simeq t(a, a_1, \ldots, a_n)$$

is also easily verified by induction.

For the converse, we can take:

$$\mathsf{k} = \langle x\rangle(\langle y\rangle\, x)$$
$$\mathsf{s} = \langle x\rangle(\langle y\rangle(\langle z\rangle\, xz(yz))).$$

One can verify that these terms satisfy the conditions in 2.2.2. □

The above lemma gives us a tool to define a lot of terms and operations in pcas. The following will be an overview for this, for most proofs I'll refer to [vO08]. In what follows I'll often abbreviate $\langle x_1\rangle(\langle x_2\rangle(\cdots(\langle x_n\rangle\, t)\cdots))$ by $\langle x_1 \cdots x_n\rangle\, t$, and we work in a pca A.

Another characterisation of pcas is given by the following proposition.

Proposition 2.2.4 (Feferman)**.** *A pas A is a pca if and only if for any $n \in \mathbb{N}$, and any term $t(x_1, \ldots, x_{n+1})$ there is an element $a \in A$ such that for all $a_1, \ldots, a_{n+1} \in A$ the following holds:*

(i) $aa_1a_2 \cdots a_n \downarrow$

(ii) $aa_1a_2\cdots a_na_{n+1} \simeq t(a_1,\ldots,a_{n+1})$

Proof. Assume A is a pca. Define, by lemma 2.2.3:

$$a = \langle x_1x_2\cdots x_{n+1}\rangle t(x_1,\ldots,x_{n+1}).$$

By the lemma, $aa_1\cdots a_n = (\langle x_{n+1}\rangle t)(a_1,\ldots,a_n)\downarrow$ and

$$aa_1\cdots a_na_{n+1} = (\langle x_{n+1}\rangle t)(a_1,\ldots,a_n)a_{n+1} \simeq t(a_1,\ldots,a_n,a_{n+1}).$$

Conversely, if the latter statement holds, we can define $\mathsf{k}\in A$ as the element corresponding to the term $t(x_1,x_2)=x_1$ and $\mathsf{s}\in A$ as the element corresponding to $t(x_1,x_2,x_3)=x_1x_3(x_2x_3)$. Then k,s satisfy the properties of definition 2.2.2. □

Remark For a pca A, every element $a\in A$ defines a partial function $A\rightharpoonup A$ given by $x\mapsto ax$ whenever ax is defined. When there is no chance of confusion, I might call this a *partial recursive function* with index a. Sometimes I might even write "the partial recursive function a" whenever the behaviour of a as a partial function is of particular importance.

The following corollary corresponds to theorem 2.6 in [Wag69] and should remind the reader of the S_n^m−theorem.

Corollary 2.2.5 (Wagner)**.** *Let A be a pca. For each $n>0$, there exists $s_n\in A$ such that for all $a\in A$, $a_1,\ldots,a_n\in A$, $m>0$ and $b_1,\ldots,b_m\in A$*

$$s_naa_1\cdots a_n\downarrow$$

and

$$s_naa_1\cdots a_nb_1\cdots b_m \simeq aa_1\cdots a_nb_1\cdots b_m.$$

Proof. Let s_n be the element as in proposition 2.2.4 corresponding to the term

$$t(x,x_1,\ldots,x_n,y) = xx_1\cdots x_ny.$$

□

2.3 Basic Constructions in Partial Combinatory Algebras

This section will be an overview of several constructions that can be done in any partial combinatory algebra. It will provide us with a great set of tools to work with later on.

2.3.1 Booleans, Pairing and Natural Numbers

We have an identity element in A, given by $\mathsf{i}=\mathsf{skk}$. For any $a\in A$, it satisfies $\mathsf{i}a=a$. Now observe that for any $a,b\in A$:

$$\mathsf{ki}ab = \mathsf{i}b = b.$$

We define $\overline{\mathsf{k}}:=\mathsf{ki}$. The elements k and $\overline{\mathsf{k}}$ will often act as boolean values for *true* and *false* respectively. For instance, for any two closed *terms* u,v we can define t such that for all $q\in A$:

$$tq \simeq \mathsf{if}\ q\ \mathsf{then}\ u\ \mathsf{else}\ v$$

In fact, let:

$$t := \langle q\rangle q(\langle w\rangle u)(\langle w\rangle v)\mathsf{k}.$$

Then $t\mathsf{k} \simeq u$ and $t\overline{\mathsf{k}} \simeq v$. We can also define the term:

$$t' := \langle qxy \rangle qxy.$$

Such that for all $a, b \in A$:

$$t'qab = \text{ if } q \text{ then } a \text{ else } b.$$

Whenever we write if … then … else … we mean a term as one of the above. We often write T and F for the elements k (true) and $\overline{\mathsf{k}}$ (false), as well as Not for the the index that yields the negations:

$$\mathsf{Not}v = \text{ if } v \text{ then } \mathsf{F} \text{ else } \mathsf{T}.$$

We define the *pairing combinator* p by:

$$\mathsf{p} := \langle xyz \rangle zxy.$$

Then we have that for all $a, b \in A$, $\mathsf{p}ab\mathsf{k} = a$, and $\mathsf{p}ab\overline{k} = b$. Therefore we have projections:

$$\mathsf{p}_0 := \langle w \rangle w\mathsf{k}$$
$$\mathsf{p}_1 := \langle w \rangle w\overline{\mathsf{k}}$$

that satisfy $\mathsf{p}_0\mathsf{p}ab = a$, $\mathsf{p}_1\mathsf{p}ab = b$.

Simulation of ordinary recursion It turns out that every non-trivial partial combinatory algebra can "simulate" the partial combinatory algebra $\mathcal{K}_1$. We first need to define representatives of the natural numbers.

Definition 2.3.1 (Curry numerals)**.** Let A be a non-trivial pca. Then we define for every $n \in \mathbb{N}$ the *Curry numeral* $\overline{n} \in A$ as follows:

- $\overline{0} = \mathsf{i} = \mathsf{skk}$
- $\overline{n+1} = \mathsf{p}\overline{\mathsf{k}}\overline{n}$.

Observe that if $\mathsf{k} \neq \overline{k}$, then $\mathsf{p}_0\overline{1} = \overline{\mathsf{k}} \neq \mathsf{k} = \mathsf{ik} = \mathsf{p}_0\overline{0}$, hence $\overline{0} \neq \overline{1}$. The former holds for any non-trivial pca, since $\mathsf{kks} = \mathsf{k} \neq \mathsf{s} = \overline{\mathsf{k}}\mathsf{ks}$ in that case. One can now easily show by induction that in a non-trivial pca, the Curry numerals are all distinct elements.

The following propositions can be found in section 1.3 of [vO08], for a proof I refer to that source.

Proposition 2.3.2. *For a non-trivial pca A, there are $S, P, Z \in A$ such that for all $n \in \mathbb{N}$:*

$$S\overline{n} = \overline{n+1}$$
$$P\overline{0} = \overline{0}$$
$$P\overline{n+1} = \overline{n}$$
$$Z\overline{0} = \mathsf{T}$$
$$Z\overline{n+1} = \mathsf{F}.$$

In every pca A, we can make definitions by recursion:

Proposition 2.3.3. *For every $a, R \in A$, there is an $f \in A$ such that*

$$f\overline{0} = a$$
$$f\overline{n+1} = R\overline{n}(f\overline{n}).$$

Moreover, we can obtain f recursively in a, R, i.e. there is an $\mathcal{R} \in A$ such that for all $a, R \in A$, $f := \mathcal{R}aR$ satisfies the above.

Proposition 2.3.4 (Recursion theorem)**.** *There are are $y, z \in A$ such that for all $f \in A$:*

1. $yf \simeq f(yf)$
2. $zf \downarrow$ *and for all* $x \in A$: $zfx \simeq f(zf)x$.

Proposition 2.3.5. *For every pca A, there is for every partial recursive function F of k-variables an element $f \in A$ such that for all $n_1, \ldots, n_k \in \mathbb{N}$:*

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

Note that the right hand side is defined if and only if $F(n_1, \ldots, n_k)$ is defined.

Coding of finite sequences We use the Curry numerals to code finite sequences in a convenient way. We assume that A is a non-trivial pca. We can define for any $n \in \mathbb{N}$ the functions $J^n : A^n \to A$ by:

- $J^0(a) = a$
- $J^{n+1}(a_0, \ldots, a_n) = pa_0J^n(a_1, \ldots, a_n)$.

For any n elements $a_0, \ldots, a_{n-1} \in A^n$, we can define the tuple $[a_0, \ldots, a_n] \in A$ by:

- $[] = \mathsf{p}\overline{00}$ (the case $n = 0$)
- $[a_0, \ldots, a_{n-1}] = \mathsf{p}\overline{n}J^n(a_1, \ldots, a_n)$.

Many operations on such tuples are recursive. As an example, I'll show that there is $t \in A$ such that for all $u = [u_0, \ldots, u_{n-1}]$, $i < n$:

$$tu\overline{i} = [u_i, \ldots, u_{n-1}]. \tag{2.1}$$

Suppose $u = [u_0, \ldots, u_{n-1}]$. Then by proposition 2.3.3, there is $f \in A$ such that:

$$f\overline{0} = u$$
$$f\overline{i+1} = p(P(p_0(f\overline{i})))(p_1(p_1(f\overline{i}))).$$

In fact, we have $f = \mathcal{R}uR$ where $R = \langle xy\rangle p(P(p_0(y)))(p_1(p_1y))$. One can now verify that for $t = \langle xy\rangle\mathcal{R}xRy$, (2.1) holds for all $u = [u_0, \ldots, u_{n-1}]$, $i < n$.

Using the same methods, one can prove that there are $\mathsf{lh}, b, c \in A$ such that for all tuples $u = [u_0, \ldots, u_{n-1}]$, $i < n$, $j \leq n$:

$$\mathsf{lh}\,u = \overline{n}$$
$$bu\overline{i} = u_i$$
$$cu\overline{j} = [u_0, \ldots, u_{j-1}].$$

We can also concatenate two tuples $u = [u_0, \ldots, u_{n-1}]$, $v = [v_0, \ldots, v_{m-1}]$, that is, there is $d \in A$ such that for every such pair of tuples, we have:

$$duv = [u_0, \ldots, u_{n-1}, v_0, \ldots, v_{m-1}].$$

To improve readability, we will often write $u^{<j}$ for $[u_0, \ldots, u_{j-1}]$ and $u * v$ for

$$[u_0, \ldots, u_{n-1}, v_0, \ldots, v_{m-1}]$$

.

2.4 Decidable, Total, and Almost total PCAs

An important property that not all pcas share is *decidability*. Informally, a pca is decidable if it is able to distinguish its own elements.

Definition 2.4.1. A partial combinatory algebra A is called *decidable* if there exists $d \in A$ such that for all $a, b \in A$:

$$dab = \begin{cases} \mathsf{T} & \text{if } a = b \\ \mathsf{F} & \text{otherwise}. \end{cases}$$

Clearly, the pca $\mathcal{K}_1$ is decidable, as well as the trivial pca. We have the following proposition, I adapted the proof from theorem 3.1 in [Wag69].

Proposition 2.4.2. *Suppose A is a non-trivial decidable pca. Then there exists $q \in A$ such that qq is undefined.*

Proof. Pick any $a, b \in A$ with $a \neq b$ (since A is non-trivial, we can pick k and s). For d as in definition 2.4.1, let $q = \langle x \rangle d(xx)bab$. Then:

$$qx \simeq d(xx)bab = \begin{cases} b & \text{if } xx \neq b \text{ and } xx\downarrow \\ a & \text{if } xx = b \\ \uparrow & \text{if } xx\uparrow \end{cases}$$

Then $qq = a$ implies that $d(qq)bab\downarrow = b$, but that implies $qq = b$, which is contradictory. Conversely, $qq = b$ implies that $d(qq)bab\downarrow = a$, which is again a contradiction.
Since the right hand side is a or b whenever it denotes, qq is undefined. □

Consider the following definition.

Definition 2.4.3. A pca A is *total* if for all $a, b \in A$, $ab\downarrow$.

Then the proposition above has an immediate corollary:

Corollary 2.4.4. *A non-trivial decidable pca is never total.*

One may wonder if a pca is non-total, but still able to decide whether ab is defined for arbitary elements a, b. It turns out that this is not the case:

Proposition 2.4.5. *A pca A is total if and only if there exists $H \in A$ such that*

$$Hab = \begin{cases} \mathsf{T} & \textit{if } ab\downarrow \\ \mathsf{F} & \textit{otherwise}. \end{cases}$$

Proof. [1] "⇒": If A is total, let

$$H = \langle xy \rangle \mathsf{T}.$$

"⇐": Suppose there exists such $H \in A$. Assume that A is not total, i.e. there are $a, b \in A$ such that ab is undefined. I'll derive a contradiction. Let u be such that for all $x \in A$:

$$ux = (\text{ if } Hxx \text{ then } (\langle x \rangle ab) \text{ else } (\langle x \rangle \mathsf{T}))\mathsf{k}.$$

Then ux is defined if and only if xx is undefined. But then uu is defined if and only if uu is undefined, a contradiction. □

Lastly, I define the notion of an *almost total* pca. It will turn out that totalness is not a categorical property of pcas, but almost totalness is.

Definition 2.4.6. A partial combinatory algebra A is called *almost total* if every partial recursive function can be extended to a total function, i.e., for every $e \in A$, there is $g \in A$ such that gx is defined for all $x \in A$, and

$$gx \precsim ex.$$

The following lemma shows that in an almost total pca, we can in fact extend every partial function to a total one in a recursive way.

Lemma 2.4.7. *Let A be an almost total pca. Then there exists $g \in A$ such that for all $e \in A$, ge is such that gex is defined for all $x \in A$ and*

$$gex \precsim ex \text{ for all } x.$$

Proof. Define f by:

$$f = \langle y \rangle (\mathsf{p}_0 y)(\mathsf{p}_1 y).$$

Since A is almost total, there is $h \in A$ such that h defines a total function and extends the partial function defined by f. Now define $g = \langle xy \rangle h(pxy)$. Then for $e \in A$ arbitrary, if ex is defined we have:

$$gex = h(pex) = f(pex) = ex$$

and $gex = h(pex)$ denotes for all x since h is total. □

Observe that every total pca is almost total. By the following proposition, a non-trivial decidable pca is never almost total:

Proposition 2.4.8. *Suppose A is a non-trivial decidable pca. Then there is $e \in A$ such that ex is undefined for some x, and for which there is no $g \in A$ such that gx is always defined and*

$$gx \precsim ex$$

for all x.

Proof. Let $a, b \in A$ be such that $a \neq b$. Let e be an index such that for all x:

$$ex \simeq x\mathsf{k}$$

[1] This proof occurred to me while reading the proof of theorem 3.4 in [Wag69].

Assume that $g \in A$ is such that $gx\downarrow$ for all x and $gx \precsim ex$. I'll derive a contradiction. By the recursion theorem, there is $h \in A$ such that $hy \simeq d(gh)aba$, so that for all $y \in A$:

$$hy = \begin{cases} b \text{ if } gh = a \\ a \text{ otherwise .} \end{cases}$$

Observe that h is total, so $eh = h\mathsf{k}\downarrow$. Then we have:

$$eh = h\mathsf{k} = \begin{cases} b & \text{if } gh = a \\ a & \text{if } gh \neq a \end{cases} = \begin{cases} b & \text{if } eh = a \\ a & \text{if } eh \neq a. \end{cases}$$

which is a contradiction. □

2.5 Morphisms between PCAs

In the following we will define an *applicative morphism* as in definition 1.5.3 of [vO08]. The notion is due to John Longley and was first defined in his thesis [Lon95]. For $\gamma : A \to B$ a relation, we define $\gamma(a) := \{b \mid (a,b) \in \gamma\}$.

Definition 2.5.1. Let A and B be pcas. A total relation $\gamma : A \to B$ is an *applicative morphism* if there exists $r \in B$, such that whenever $a, a' \in A$ and $aa'\downarrow$, we have for all $b \in \gamma(a)$, $b' \in \gamma(a')$ that

$$rbb'\downarrow \text{ and } rbb' \in \gamma(aa')$$

The applicative morphism γ is said to be *realized* by such an element r.

There is a composition of morphisms: For $\gamma : A \to B$ and $\delta : B \to C$, the composition of γ and δ can be defined by:

$$\delta\gamma(a) = \bigcup_{b \in \gamma(a)} \delta(b).$$

One can verify that this is indeed an applicative morphism, i.e. that it has a realizer. There is an identity morphism $\iota : A \to A$ for every pca A, it is given by $\iota(a) = \{a\}$ and realized by i.

We can define a binary relation $\preceq$ on the set of applicative morphisms $A \to B$. For $\delta, \gamma : A \to B$ applicative morphisms, $\delta \preceq \gamma$ when there is $r \in B$ such that for all $a \in A, b \in \delta(a)$, $rb\downarrow$ and $rb \in \gamma(a)$.

A proof of the following proposition can be found in [vO08]:

Proposition 2.5.2. *The relation $\preceq$ defined above is a preorder, and this order is preserved by composition on both sides. Therefore, partial combinatory algebras together with applicative morphisms form a preorder enriched category, denoted PCA.*

The category PCA does not have much categorical structure. A lot of properties of pcas are not preserved under isomorphism or equivalence. However, the category behaves well with corresponding structures called *assemblies* and *realizability toposes*, which we will see below. We will see that totalness is not preserved under isomorphism of pcas. However, almost totalness (definition 2.4.6) is:

Proposition 2.5.3. *Suppose A, B are isomorphic partial combinatory algebras, i.e. there are applicative morphisms $\gamma : A \to B$, $\delta : B \to A$ such that*

$$\gamma \circ \delta = \iota_B \ , \delta \circ \gamma = \iota_A,$$

where ι_B, ι_A are the identity morphisms on B and A. If A is almost total, then B is.

Proof. Let $b \in B$ be arbitrary. We will find an index g such that gx is defined for all $x \in B$ and

$$gx \precsim bx$$

for all x.

Suppose γ is realized by r and δ is realized by t. Take $a \in \delta(b)$. Then for all b', and $a' \in \delta(b')$:

$$bb' \downarrow \Rightarrow taa' \downarrow, taa' \in \delta(bb').$$

Since A is almost total, there is $h \in A$ such that hy is defined for all $y \in A$ and

$$hy \precsim tay$$

for all $y \in A$. Now take $f \in \gamma(h)$ arbitrary. For all $b' \in B$, we have that $b' \in \gamma \circ \delta(b')$. So $b' \in \gamma(a')$ for some $a' \in \delta(b')$. Therefore:

$$rfb' \downarrow, rfb' \in \gamma(ha').$$

Moreover, if $bb' \downarrow$, then:

$$rfb' \in \gamma(ha') = \gamma(taa') \subseteq \gamma \circ \delta(bb') = \{bb'\}$$

hence $rfb' = bb'$. So if we define $g = rf$, then it defines an extension of b to a total function. □

The following proposition shows that in the category PCA, a total pca is the same thing as an almost total pca.

Proposition 2.5.4. *Every almost total pca is isomorphic to a total pca.*

Proof. Let A be an almost total pca. We'll define a total pca A_{tot} as follows. Let $g \in A$ be as in lemma 2.4.7, i.e. for every $e \in A$, we have that ge is total and extends the partial function indexed by e.

Define an applicative structure $*$ on the set A as follows:

$$a * b = gab.$$

Then A_{tot} is the total applicative structure defined by $*$ on A. Observe that

$$\mathsf{k} * a * b = g(g\mathsf{k}a)b = \mathsf{k}ab = a.$$

Now let $\mathsf{s}' := \langle xyz \rangle g(gxz)(gyz)$. Then:

$$\mathsf{s}' * a * b * c := g(g(g\mathsf{s}'a)b)c = g(\langle z \rangle g(gaz)(gbz))c = g(gac)(gbc) = a * c * (b * c).$$

Therefore A_{tot} is a total pca. Now $\gamma : A \to A_{\text{tot}}$ defined by $\gamma(a) = \{a\}$ is an applicative morphism realized by the identity $i' = \mathsf{s}' * \mathsf{k} * \mathsf{k}$ of A_{tot}, and $\delta : A_{\text{tot}} \to A$ defined by $\delta(a) = \{a\}$ is an applicative morphism realized by g.

Therefore A_{tot} and A are isomorphic. □

We can summarize the above in the following lemma:

Lemma 2.5.5. *A pca is almost total if and only if it is isomorphic to a total pca.*

Decidability is preserved under equivalence of pcas.

Theorem 2.5.6 ([Lon95])**.** *Suppose A, B are equivalent partial combinatory algebras. Then if A is decidable, B is.*

Proof. Assume A, B are equivalent partial combinatory algebras with A decidable. Let $\mathsf{d} \in A$ be the witness for decidability. There are applicative morphisms $\gamma : A \to B$, and $\delta : B \to A$, realized by r, t respectively, such that

$$\iota_A \simeq \delta \circ \gamma \text{ and } \iota_B \simeq \gamma \circ \delta.$$

First, let $t \in \delta(\mathsf{T}_B)$ and $f \in \delta(\mathsf{F}_B)$. Pick $v \in \gamma(\langle xy\rangle \mathsf{d}xytf)$. Let $a, a' \in A$ be arbitrary. Then for $b \in \gamma(a), b' \in \gamma(a')$, we have:

$$r(rvb)b' \in \gamma(\mathsf{d}aa'tf) = \begin{cases} \gamma(t) & \text{if } a = a' \\ \gamma(f) & \text{if } a \neq a' \end{cases}$$
$$\subseteq \begin{cases} \gamma \circ \delta(\mathsf{T}_B) & \text{if } a = a' \\ \gamma \circ \delta(\mathsf{F}_B) & \text{if } a \neq a'. \end{cases}$$

Let g be a realizer of $\gamma \circ \delta \preceq \iota_B$. Define $e \in B$ by $e = \langle xy\rangle g(r(rvx)y)$. Then for all $a, a' \in A$, $b \in \gamma(a), b' \in \gamma(a')$, we have:

$$ebb' = \begin{cases} \mathsf{T}_B & \text{if } a = a' \\ \mathsf{F}_B & \text{if } a \neq a'. \end{cases}$$

Claim. *For $b, b' \in B$, we have that $b \neq b' \Rightarrow \delta(b) \cap \delta(b') = \emptyset$. Similarly, for $a, a' \in A$, we have $a \neq a' \Rightarrow \gamma(a) \cap \gamma(a') = \emptyset$.*

Proof of claim. Suppose $b \neq b'$, but there exists $a \in \delta(b) \cap \delta(b')$. Then $\gamma(a) \subseteq \gamma \circ \delta(b)$ and $\gamma(a) \subseteq \gamma \circ \delta(b')$. Pick $c \in \gamma(a)$. Then with g as above, $gc \in \{b\}$ and $gc \in \{b'\}$, a contradiction.

The statement for A follows analogously, since we also have $\delta \circ \gamma \preceq \iota_A$. □

Let h be a realizer of $\iota_B \preceq \gamma \circ \delta$. It is easy to see that for all $b, b' \in B$, $b = b'$ if and only if $hb = hb'$. For each b, $hb \in \gamma(a)$ for some $a \in \delta(b)$. By the claim above, such a is unique. Using the claim again, we can conclude that for all $a, a' \in A$, $b, b' \in B$, if $hb \in \gamma(a)$, $hb' \in \gamma(a')$, then $hb = hb'$ if and only if $a = a'$.

So define $d \in B$ by $d = \langle xy\rangle e(hx)(hy)$. Then for all $b, b' \in B$:

$$dbb' = \begin{cases} \mathsf{T}_B & \text{if } hb = hb' \\ \mathsf{F}_B & \text{if } hb \neq hb' \end{cases} = \begin{cases} \mathsf{T}_B & \text{if } b = b' \\ \mathsf{F}_B & \text{if } b \neq b'. \end{cases}$$

So d is a witness for decidability of B. □

2.5.1 Computational Density and Decidability

An important special case of an applicative morphism is a *computationally dense* applicative morphism. It was discovered by Hofstra and van Oosten in [HvO03]. Intuitively, an applicative morphism $\gamma : A \to B$ is computationally dense if any computation in B on elements of A (so elements in the image of γ) can be done in A. For readability, we introduce some notation; for B a pca, and $P \subseteq B$ a subset, we define for all $b \in B$:

$$bP = \begin{cases} \{bb' \mid b' \in P\} & \text{if } (\forall b' \in P)bb' \downarrow \\ \text{undefined} & \text{otherwise} \end{cases}$$

We write $bP \downarrow$ whenever $(\forall b' \in P)bb' \downarrow$. When making statements about bP, we usually mean $bP \downarrow$ as well.

Definition 2.5.7 ([HvO03])**.** An applicative morphism $\gamma : A \to B$ between pcas is called *computationally dense* if there exists $m \in B$ such that:

$$(\forall b \in B)(\exists a \in A)(\forall a' \in A) : \text{if } b\gamma(a') \neq \emptyset \\ \text{then } aa' \downarrow, m\gamma(aa') \downarrow, \text{ and } m\gamma(aa') \subseteq b\gamma(a').$$

For γ a computationally dense applicative morphism, we let $\Lambda : B \to A$ denote a function such that

$$(\forall b \in B)(\forall a' \in A) : \text{if } b\gamma(a') \downarrow \\ \text{then } \Lambda(b)a' \downarrow, m\gamma(\Lambda(b)a') \downarrow, \text{ and } m\gamma(\Lambda(b)a') \subseteq b\gamma(a').$$

Unless otherwise stated, we pick this Λ using the axiom of choice on Set.

Computationally dense applicative morphisms express a strong relation between the computational nature of the pcas. This has a deep consequence for their realizability toposes that we will see later.

The following proposition is a recent result by P. Johnstone in [Joh13]. I included an adapted proof here, for the notation in [Joh13] is quite different.

Proposition 2.5.8 ([Joh13])**.** *An applicative morphism $\gamma : A \to B$ is computationally dense if and only if there exists $r \in B$ such that for all $b \in B$, there exists $a \in A$ so that $r\gamma(a) = \{b\}$.*

Proof. We assume that γ is realized by $t \in B$.

"$\Rightarrow$": Assume that computational density is realized by m, and let $\Lambda : B \to A$ be as in definition 2.5.7. Let $b \in B$ be arbitrary, and let $a = \Lambda(\langle x \rangle b)$. Then $aa \downarrow$ and:

$$m\gamma(aa) = m\gamma(\Lambda(\langle x \rangle b)a) \subseteq (\langle x \rangle b)\gamma(a) = \{b\}$$

and $m\gamma(aa) \downarrow$. So $r = m$ satisfies the statement.

"$\Leftarrow$": Let $r \in B$ be as on the right hand side of the equivalence. Let $q_0 \in \gamma(\mathsf{p}_0), q_1 \in \gamma(\mathsf{p}_1)$ and define $m = \langle x \rangle r(tq_0x)(tq_1x)$. Let $b \in B$ be arbitrary. Let $h \in A$ be so that whenever $b' \in \gamma(h)$, we have $rb' = b$, and define $a = \mathsf{p}h$.

Suppose a' is such that $b\gamma(a') \downarrow$. Then whenever $c \in \gamma(aa') = \gamma(\mathsf{p}ha')$, we have:

$$mc = r(\underbrace{tq_0c}_{\in\gamma(h)})(\underbrace{tq_1c}_{\in\gamma(a')}) \in b\gamma(a').$$

So m witnesses computational density. □

Peter Johnstone calls the property on the right hand side of the equivalence "quasi-surjective". A strengthening of this notion is the following:

Definition 2.5.9. An applicative morphism $\gamma : A \to B$ is *effectively quasi-surjective* if there exists e, r such that

$$(\forall b \in B)(\exists a \in A)eb \in \gamma(a) \wedge r\gamma(a) = \{b\}.$$

Clearly, if e, r realize that an applicative morphism if effectively quasi-surjective, then r realizes that it is quasi-surjective.

We will need the following proposition later on, but here is the best place to state it:

Proposition 2.5.10. *An applicative morphism is effectively quasi-surjective if and only if there exists e, m such that m realizes computational density as in definition 2.5.7 and:*

$$(\forall b \in B)(\exists a \in A)eb \in \gamma(a) \wedge m\gamma(a) = \{b\}.$$

Proof. We assume that $\gamma : A \to B$ is an applicative morphism realized by $t \in B$. The "$\Leftarrow$" direction is trivial.

For "$\Rightarrow$": Let e, r be as in 2.5.9. Take $m = \langle x \rangle r(tq_0x)(tq_1x)$ as in the proof of 2.5.8. Take $q \in \gamma(p)$, $f \in \gamma(\mathsf{F})$ Let $e' = \langle y \rangle t(tq(e(\langle x \rangle y)))f$. Then for all $b \in B$, $e'b \in \gamma(\mathsf{p}a\mathsf{F})$ for some a such that $e\langle x \rangle b \in \gamma(a)$ and $r\gamma(a) \subseteq \{\langle x \rangle b\}$.

From the latter, we have for all $b' \in \gamma(\mathsf{p}a\mathsf{F})$:

$$mb' = r(tq_0b')(tq_1b') = (\langle x \rangle b)f = b.$$

Therefore e', m are witnesses for the statement on the right-hand side. □

In 5.1, we will see that geometric inclusions of realizability toposes correspond precisely to effectively quasi-surjective applicative morphisms. In corollary 2.5.17 we give another characterisation of these applicative morphisms.

Another special case of an applicative morphism that expresses a deeper relation in the computational nature of pcas is a *decidable applicative morphism*:

Definition 2.5.11. Let A, B be pcas. Denote their canonical boolean values by $\mathsf{T}_A, \mathsf{F}_A$ and $\mathsf{T}_B, \mathsf{F}_B$. An applicative morphism $\gamma : A \to B$ is called *decidable* if there is $d \in B$ (a *decider* for γ) such that for all $b \in \gamma(\mathsf{T}_A)$, $db = \mathsf{T}_B$, and for all $c \in \gamma(\mathsf{F}_A)$, $dc = \mathsf{F}_B$.

Decidable applicative morphisms play an important role in general relative recursion. We also have the following proposition:

Proposition 2.5.12 ([vO08])**.** *Suppose $\gamma : A \to B$ is computationally dense. Then γ is decidable.*

Proof. Suppose γ is realized by $t \in B$. We take $r \in B$ as in proposition 2.5.8. Take $a, a' \in A$ such that $r\gamma(a) = \{\mathsf{T}_B\}$ and $r\gamma(a') = \{\mathsf{F}_B\}$. Define $c \in A$ by $c := \langle v \rangle vaa'$, take $e \in \gamma(c)$ and define $d := \langle x \rangle r(tex)$.

Then for $u \in \gamma(\mathsf{T}_A)$, $v \in \gamma(\mathsf{F}_B)$:

$$du = r(teu) \in r\gamma(a) = \{\mathsf{T}_B\}\ ,\ dv = r(tev) \in r\gamma(a') = \{\mathsf{F}_B\}$$

so d realizes decidability of γ. □

2.5.2 Adjoint Pairs of Applicative Morphisms

The pre-order $\preceq$ gives rise to the notion of an adjoint pair of applicative morphisms.

Definition 2.5.13. Let A, B be partial combinatory algebras, with $\gamma : A \to B$, $\delta : B \to A$ applicative morphisms. We say that γ is left-adjoint to δ, δ is right-adjoint to γ or $\gamma \dashv \delta$, when $\gamma\delta \preceq \iota_B$ and $\iota_A \preceq \delta\gamma$.

Definition 2.5.14 ([Lon95])**.** (i) We call an applicative morphism $\gamma : A \to B$ *projective* if there is an applicative morphism $\gamma' : A \to B$ such that $\gamma \simeq \gamma'$ and γ' is single valued, i.e. for each $a \in A$, $\gamma'(a)$ is a singleton set.

(ii) We call an applicative morphism $\delta : B \to A$ *discrete* if $\delta(b) \cap \delta(b') = \emptyset$ whenever $b \neq b'$.

The following theorem is a result by Longley (Theorem 2.5.3 in [Lon95]), and will come in handy.

Theorem 2.5.15. *Let $\gamma : A \to B, \delta : B \to A$ be applicative morphisms. Then the following holds:*

(i) If $\gamma\delta \preceq \iota_B$, then δ is discrete.

(ii) If $\gamma \dashv \delta$, then γ is projective.

The following theorem is an observation of mine that I have not encountered in any literature.

Theorem 2.5.16. *Let $\gamma : A \to B$ be an applicative morphism. Then γ is computationally dense if and only if there exists an applicative morphism $\delta : B \to A$ such that $\gamma\delta \preceq \iota_B$.*

Moreover, such γ is projective if and only if it has a right adjoint.

Proof. "⇒" Suppose m and $\Lambda : B \to A$ are as in definition 2.5.7. Assume γ is realized by t.

Define $\delta : B \to A$ by

$$\delta(b) = \{a \in A \mid m\gamma(a) = \{b\}\}.$$

Observe that for every $b \in B$, $\delta(b)$ is non-empty, since for any $a' \in A$, $\Lambda(\langle x\rangle b)a' \in \delta(b)$:

$$m\gamma(\Lambda(\langle x\rangle b)a') = \langle x\rangle b\gamma(a') = \{b\}.$$

We have to show that δ has a realizer.

Define $m' = \langle x\rangle m(tq_0x)(m(tq_1x))$ where $q_i \in \gamma(\mathsf{p}_i)$. Then whenever $a \in \delta(b)$, we have: for all b', a' such that $bb'\downarrow$, $m\gamma(a') = \{b'\}$, $c \in \gamma(\mathsf{p}aa')$:

$$m'c = m(tq_0c)(m(tq_1c)) = bb'.$$

Now let $g = \Lambda(m')$. Then for all $a' \in A$, if $m'\gamma(a')$ is defined, then:

$$m\gamma(ga') \subseteq m'\gamma(a').$$

For $a \in \delta(b), a' \in \delta(b')$, we have $m\gamma(a) \subseteq \{b\}$ and $m\gamma(a') \subseteq \{b'\}$. Hence if $bb'\downarrow$:

$$\begin{aligned} m\gamma(g(\mathsf{p}aa')) &= m'\gamma(\mathsf{p}aa') \\ &\subseteq m(tq_0\gamma(\mathsf{p}aa'))(m(tq_1\gamma(\mathsf{p}aa'))) \subseteq m\gamma(a)(m\gamma(a')) \subseteq \{bb'\}. \end{aligned}$$

It follows that $g(\mathsf{p}aa') \in \delta(bb')$ (whenever $bb'\downarrow$), so $\langle xy\rangle g(\mathsf{p}xy)$ realizes $\delta : B \to A$.

It is easy to see that m realizes $\gamma\delta \preceq \iota_B$.

"⇐": Suppose we have such $\delta : B \to A$, and suppose $\gamma\delta \preceq \iota_B$ is realized by r. Then for $b \in B$, take $a \in \delta(b)$. Then:

$$r\gamma(a) = \{b\}$$

hence γ is computationally dense by proposition 2.5.8.

For the last statement, we can assume that γ is single-valued. Let $j = \Lambda(\mathsf{i})$. Then for all $a' \in A$:

$$m\gamma(ja') \subseteq i\gamma(a') = \gamma(a').$$

Hence $ja' \in \delta(b')$ for the unique $b' \in \gamma(a')$, hence $ja' \in \delta \circ \gamma(a')$, therefore j realizes $\iota_A \preceq \delta\gamma$. So δ is a right adjoint.

Conversely, if γ has a right adjoint, it is projective by theorem 2.5.15. □

Corollary 2.5.17. *An applicative morphism $\gamma : A \to B$ is effectively quasi-surjective if and only if there exists an applicative morphism $\delta : B \to A$ such that $\gamma\delta \simeq \iota_B$.*

Proof. "⇒": By proposition 2.5.10, we have $e, m \in B$ such that m witnesses computational density in the sense of definition 2.5.7 and

$$(\forall b \in B)(\exists a)eb \in \gamma(a), m\gamma(a) = \{b\}.$$

By the proof of theorem 2.5.16,

$$\delta(b) = \{a \mid m\gamma(a) = \{b\}\}$$

is an applicative morphism $B \to A$, and m realizes $\gamma\delta \preceq \iota_B$. It is easy to see that e realizes $\iota_B \preceq \gamma\delta$.

"⇐": If r realizes $\gamma\delta \preceq \iota_B$ and e realizes $\iota_B \preceq \gamma\delta$, then e, r realize that γ is effectively quasi-surjective in the sense of definition 2.5.9. □

One may wonder whether we can strengthen theorem 2.5.16 by proving that every computationally dense morphism has a right adjoint. It turns out that this is not the case. We'll sketch a counterexample here (thanks to Jaap van Oosten for pointing this out). We denote by $\mathcal{K}_2$ the pca known as *Kleene's second model* . It has as underlying set the set of functions $\mathbb{N}^{\mathbb{N}}$ and application is defined by:

$$\alpha \cdot \beta \simeq \gamma \iff (\forall n)(\exists k)(\forall l < k)\alpha(\langle n, \beta(0), \ldots, \beta(k)\rangle) = \gamma(n) + 1 \wedge \alpha(\langle n, \beta(0), \ldots, \beta(l)\rangle) = 0$$

Where $\langle \ldots \rangle$ denotes some effective coding of tuples. The proof that this yields a pca is a tedious exercise and can be found in [vO08], section 1.4.3. It turns out that the restriction of the above applicative structure to the subset $\mathbb{N}^{\mathbb{N}}_{\text{rec}} \subset \mathbb{N}^{\mathbb{N}}$ consisting of the computable functions is also a pca, that we denote by $\mathcal{K}_{2\text{rec}}$.

Example 2.5.18. *The relation $\gamma : \mathcal{K}_{2\,rec} \to \mathcal{K}_1$ defined by:*

$$\gamma(f) = \{e \mid e \text{ is an index for } f\}$$

is a computationally dense applicative morphism. Moreover, γ is not projective.

Proof. The fact that γ is an applicative morphism is an easy programming exercise.

Let us prove that it is computationally dense. Let $r \in \mathcal{K}_1$ be such that:

$$re \simeq e0.$$

Then it is easy to see that for every $b \in \mathcal{K}_1$, there exists $f \in \mathcal{K}_{2\text{rec}}$ such that

$$r\gamma(f) = \{b\}$$

(take f any computable function such that $f(0) = b$). Then r satisfies the statement in proposition 2.5.8, so γ is computationally dense.

To show that γ is not projective, we will assume it is and derive a contradiction. So suppose $\gamma' : \mathcal{K}_{2\text{rec}} \to \mathcal{K}_1$ is a single valued applicative morphism and $\gamma \simeq \gamma'$.

Since $\gamma \preceq \gamma'$, there is an index $r \in \mathcal{K}_1$ such that:

$$(\forall f)(\exists e) e \in \gamma(f) \wedge r\gamma(f) = \{e\}.$$

Let t be an index of Kleene's $T-$predicate, i.e. for all $e, a \in \mathcal{K}_1$, tea is total and:

$$(\exists s)\, teas \neq 1 \iff ea\downarrow .$$

Then:

$$ea\downarrow \iff r(tea) \neq r(\langle x\rangle 1).$$

This is in contradiction with undecidability of the halting problem. □

In the above we have found a computationally dense applicative morphism that is not projective, hence by theorem 2.5.16, it does not have a right adjoint.

2.5.3 Proto-applicative Morphisms

Theorem 2.5.16 and the discussion below it show that computationally dense applicative morphisms correspond to "semi-adjoint" pairs of applicative morphisms. We would like to present this type of applicative morphisms as an actual adjoint pair of some sorts. In this section we introduce a new notion that does exactly that. For a pca A, we write $\mathcal{P}^*(A)$ for the set of non-empty subsets of A, and for $\alpha, \beta \in \mathcal{P}*(A)$ we define

$$\alpha\beta = \{ab \mid a \in \alpha, b \in \beta\}$$

whenever $(\forall a \in \alpha)(\forall b \in \beta) ab\downarrow$.

Definition 2.5.19. A *proto-applicative morphism* $\Gamma : A \to B$ is a function

$$\Gamma : \mathcal{P}^*(A) \to \mathcal{P}^*(B)$$

together with an element $r \in B$ that satisfies:

$$r\Gamma(\alpha)\Gamma(\beta) \subseteq \Gamma(\alpha\beta)$$

for all $\alpha, \beta \in \mathcal{P}^*(A)$ such that $\alpha\beta$ is defined. Again we say that r *realizes* Γ.

One can verify that proto-applicative morphisms are closed under composition. We therefore obtain a category $\mathrm{PCA}_{\mathrm{proto}}$, with pcas as objects and proto-applicative morphisms as arrows.

We order proto-applicative morphisms analogous to applicative morphisms; for

$$\Gamma, \Delta : A \to B$$

proto-applicative morphisms, we say $\Gamma \preceq \Delta$ if there exists $t \in B$ such that for all $\alpha \in \mathcal{P}^*(A)$:

$$t\Gamma(\alpha) \subseteq \Delta(\alpha).$$

Again we say that t *realizes* $\Gamma \preceq \Delta$. This defines a preorder on proto-applicative morphisms. The identity function on $\mathcal{P}^*(A)$ is a proto-applicative morphism that we denote by I_A. We similarly define an adjoint pair of proto-applicative morphisms:

$$\Gamma \dashv \Delta : B \to A \iff \Gamma \circ \Delta \preceq I_B \text{ and } I_A \preceq \Delta \circ \Gamma.$$

Any applicative morphism $\gamma : A \to B$ gives rise to a proto-applicative morphism $\mathcal{P}(\gamma)$:

$$\mathcal{P}(\gamma)(\alpha) = \bigcup_{a \in \alpha} \gamma(a).$$

Observe that $I_A = \mathcal{P}(\iota_A)$. We now have the following theorem:

Theorem 2.5.20. *An applicative morphism $\gamma : A \to B$ is computationally dense if and only if $\mathcal{P}(\gamma)$ has a right adjoint.*

Proof. First, suppose $\mathcal{P}(\gamma)$ has a right adjoint $\Delta : B \to A$. Define

$$\delta(b) = \Delta(\{b\}).$$

Then δ is an applicative morphism $B \to A$, and

$$\gamma \circ \delta(b) = \bigcup_{a \in \Delta\{b\}} \gamma(a) = \mathcal{P}(\gamma) \circ \Delta(\{b\})$$

Since $\mathcal{P}(\gamma) \circ \Delta \preceq I_B$, $\gamma \circ \delta \preceq \iota_B$, hence γ is computationally dense.

Conversely, suppose $\gamma : A \to B$ is computationally dense, witnessed by $m \in A$ as in definition 2.5.7. Define $\Delta : B \to A$ by:

$$\Delta(\beta) = \{a \in A \mid m\gamma(a) \subseteq \beta\}.$$

The fact that Δ is a proto-applicative morphism, and that $\mathcal{P}(\gamma) \circ \Delta \preceq I_B$ can be shown in the same way as in the proof of theorem 2.5.16. To prove $I_A \preceq \Delta \circ \mathcal{P}(\gamma)$, we take $j = \Lambda(\mathsf{i})$ as defined in the last part of the same proof. Then for $\alpha \in \mathcal{P}^*(A)$, we have for all $a' \in \alpha$:

$$m\gamma(ja') \subseteq \mathsf{i}\gamma(a') = \gamma(a') \subseteq \mathcal{P}(\gamma)(\alpha)$$

so $ja' \in \Delta \circ \mathcal{P}(\gamma)(\alpha)$, whence we see that j realizes

$$I_A \preceq \Delta \circ \mathcal{P}(\gamma).$$

□

Remark The above fits in a theory about *order pcas* , which are generalizations of pcas. These were introduced in [HvO03]. The category of pcas and applicative morphisms turns out to be a full subcategory of the Kleisli 2-category of a certain monad in the category of order pcas. For pcas, this monad corresponds to the operation $A \mapsto \mathcal{P}^*(A)$, where $\mathcal{P}^*(A)$ has the structure of an order-pca. The above can therefore be viewed as an additional motivation to study order pcas instead of pcas. However, in this thesis we will only consider pcas, and for those the above picture is somewhat simpler and easier to keep in mind. Still, the above generalizes to order-pcas, which has interesting consequences for the study of assemblies and realizability toposes. For more details on order-pcas, see [HvO03] or [vO08], section 1.8.

2.6 Relative Recursion in PCAs

In this section we will exhibit a method to construct from a pca A and a function $f : A \to A$ a pca $A[f]$ that contains precisely the partial recursive functions "computable" from f. The construction was introduced in [vO06] and generalizes ordinary relative recursion, where for each $f : \mathbb{N} \to \mathbb{N}$ there is a pca $\mathcal{K}_1^f$ with applicative structure defined by:

$$en = \Phi_e^f(n)$$

where Φ_e^f is the partial function with index e and oracle f for some model of computation with oracle. We first define what it means for a function to be *representable* with respect to an applicative morphism.

Definition 2.6.1. Let $\gamma : A \to B$ be an applicative morphism. Then a partial function $f : A \to A$ is *representable* with respect to γ if there is $r_f \in B$ such that for all $a \in A$, whenever $f(a)$ is defined we have for all $b \in \gamma(a)$

$$r_f b \downarrow \text{ and } r_f b \in \gamma(f(a)).$$

We say a partial function $f : A \to A$ is *representable* (in A) if it is representable with respect to the identity morphism $\iota : a \mapsto \{a\}$.

With the above definition, we have the following properties:

- For $\gamma : A \to B$ an applicative morphism, any representable total function $f : A \to A$ is representable with respect to γ.
- If $\gamma : A \to B$, $\delta : B \to A$ are such that $\delta\gamma = \iota$, then any total $f : A \to A$ representable with respect to γ is representable in A.

Note that for total functions, a representable function is the same thing as a recursive function. For partial functions, this is not the case!

The construction of $A[f]$ in [vO06] uses *dialogues* to model a computation where the pca A "interrogates" the function f.

Definition 2.6.2. Let A be a pca. For $f : A \to A$ a partial function, an *f-dialogue* between a and b is an element $u = [u_0, \ldots, u_{n-1}]$ of A, such that for all $i < n$, there is a $v_i \in A$ that satisfies

$$a([b] * u^{<i}) = p\mathsf{F}v_i \text{ and } u_i = f(v_i)$$

where $*$ and $u^{<i}$ refer to the operations on tuples defined in section 2.3.1.

We say that an $f-$dialogue u *terminates* if

$$a([b] * u) = p\mathsf{T}c \text{ for some } c \in A.$$

Now a partial applicative structure $A[f]$ with applicative map $\cdot^f : A \times A \to A$ is defined as follows: for $a, b \in A$, $a \cdot^f b = c$ if and only if there is a terminating $f-$dialogue u such that

$$a([b] * u) = c.$$

A proof of the following theorem can be found in [vO06] and [vO08].

Theorem 2.6.3. *For every non-trivial pca A and every partial endofunction f on A, the partial applicative structure $A[f]$ is a pca.*

Moreover, there is a decidable applicative morphism $\iota_f : A \to A[f]$, given by $a \mapsto \{a\}$, such that

1. *f is representable with respect to ι_f*
2. *For every decidable applicative morphism $\gamma : A \to B$ such that f is representable with respect to γ, there is a decidable applicative morphism $\gamma_f : A[f] \to B$ such that $\gamma_f \iota_f = \gamma$, and γ is unique with this property. Moreover, if $\delta : A[f] \to B$ is such that $\delta \iota_f \cong \gamma$, then $\delta \cong \gamma_f$.*

The following corollary is also formulated:

Corollary 2.6.4. *(i) If f is representable in A, then A and $A[f]$ are isomorphic.*

(ii) For partial functions $f, g : A \to A$, we have that $A[f][g]$ and $A[g][f]$ are isomorphic. therefore we can write $A[f, g]$ for such a pca.

(iii) For $\mathcal{K}_1$, the construction agrees with ordinary relative recursion, i.e. $\mathcal{K}_1^f$ and $\mathcal{K}_1[f]$ are isomorphic.

(iv) Every total pca is isomorphic to a nontotal one.

One fact about the proof of theorem 2.6.3 that needs to be pointed out is that the elements k, s that witness combinatory completeness of $A[f]$ can be chosen independently of f. So for every f, k, s are the same elements of A, only the applicative map is different. This can be understood in the same way as in ordinary recursion theory, where the indexing (i.e. the code of a program) does not depend on the oracle itself. In fact coding could never depend on the oracle, since the programmer does not know the oracle. We will now wrap this up in a proposition for easy reference. First we introduce some notation. For a term $t(x_1, \ldots, x_n)$, and $a_1, \ldots, a_n \in A$, we write $t^f(a_1, \ldots, a_n)$ for the interpretation of the closed term $t(a_1, \ldots, a_n)$ in $A[f]$. It may or may not be defined, whenever it is we say $t(a_1, \ldots, a_n)$ *denotes with oracle f*, or $t^f(a_1, \ldots, a_n)\downarrow$.

Proposition 2.6.5. *For any term*

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

there is a term

$$\langle x \rangle^{(-)} t(x_1, \ldots, x_n)$$

such that for all $a, a_1, \ldots, a_n \in A$, $f : A \to A$:

$$(\langle x \rangle^{(-)} t)(a_1, \ldots, a_n) \downarrow \ (\textit{in } A)$$
$$t^f(a, a_1, \ldots, a_{n+1}) \simeq \langle x \rangle^{(-)} t(a_1, \ldots, a_n) \cdot^f a$$

Proof. Let $\mathsf{k},\mathsf{s} \in A$ witness combinatory completeness of $A[f]$, for every f. Using primitive recursion, one can show that there is a term $C(x,y)$ of A such that $(x,y)([c]*u)$ is given by the following instructions:

$$C(x,y)([c]*u) \simeq \begin{cases} x([y]*u) & \text{if } (\forall i \le \mathsf{lh}u)\mathsf{Not}\mathsf{p}_0 x([y]*u^{<i}) \\ \mathsf{p}_1(x([y]*u^{<i}))([c]*u^{\ge i}) & \text{for } i \text{ least s.t. } \mathsf{p}_0(x([y]*u^{<i})), \text{ otherwise} \end{cases}$$

Then for $a,b,c \in A$, $C(a,b)\cdot^f c$ first computes $a\cdot^f b$, and whenever that finishes it goes on to compute $a\cdot^f b\cdot^f c$. In fact we have

$$C(a,b)\cdot^f c \simeq a\cdot^f b\cdot^f c.$$

We now follow the proof of 2.2.3:

If t is a constant or a variable y different from x, let $\langle x\rangle^{(-)} t = C(\mathsf{k},t)$.
If $t = x$, let $\langle x\rangle^{(-)} t = C(C(\mathsf{s},\mathsf{k}),\mathsf{k})$.
If $t = t_1 t_2$, let $\langle x\rangle^{(-)} t = C(C(\mathsf{s},(\langle x\rangle^{(-)} t_1)),(\langle x\rangle^{(-)} t_2))$.
One can now verify the statement by induction. □

2.7 Forcing Effective Operations of Type 2

In this section we will present a new application of generalized relative recursion as defined above. We will specify a method to construct from a partial combinatory algebra A and a functional $F : A^A \to A$, a pca $A[F]$ in which F is an *effective operation of type 2*.

We will very quickly define a *type structure* for any pca A. The set of elements of type σ will be denoted by A_σ. We define the *pure types* as follows:

- Elements of A have pure type $\overline{0}$.
- Elements of $A^{A_{\overline{n}}}$ have pure type $\overline{n+1}$.

One could also define functionals of composed type, for example the functionals $A^A \times A \to A$. We can however always (using constant functions and pairing), code those into functionals of pure type. Therefore, whenever I refer to a *type*, I usually mean pure type. Also, I might not always write a bar above the type of a functional (e.g. Let F be a type-2 functional). I mean the same thing, but the bar is basically there when we need later on that the types can be seen as elements of the pca A, i.e. they are coded as the Curry numerals. In that case, indeed, the types have a type themselves.

We now define the effective operations:

Definition 2.7.1 (Effective Operation)**.** For a pca A, $n \in \mathbb{N}$, we define a set Q_n of *effective operations of type n*, and a total relation $I_n : Q_n \to A$ by induction:

- $Q_0 = A$, $I_0(a) = \{a\}$.
- $I_{n+1}(F) = \{e \in A \mid F : Q_n \to A \text{ and } (\forall f \in Q_n)(\forall a \in I_n(f))\, ea = F(f)\}$
- $Q_{n+1} = \{F \mid I_{n+1}(F) \neq \emptyset\}$.

We also define effective operations relative to an applicative morphism $\gamma : A \to B$.

Definition 2.7.2 (Relative Effective Operation)**.** For an applicative morphism $\gamma : A \to B$ between pcas, $n \in \mathbb{N}$, we define a set Q_n^γ of *effective operations of type n relative to* γ, and a total relation $I_n^\gamma : Q_n^\gamma \to A$ by induction:

- $Q_0^\gamma = A$, $I_0^\gamma = \gamma$.

- $I^{\gamma}_{n+1}(F) = \{e \in B \mid F : Q_n \to A \text{ and } (\forall f \in Q_n)(\forall b \in I_n(f))\, eb \in \gamma(F(f))\}$
- $Q^{\gamma}_{n+1} = \{F \mid I_{n+1}(F) \neq \emptyset\}$.

Remark Observe that effective operations for a pca A are the same as effective operations relative to the identity ι_A. We will often write Q^A_n instead of $Q^{\iota_A}_n$. Also note that a *representable* total function $f : A \to A$ as in definition 2.6.1 is the same thing as an effective operation of type 1. In fact, we can rewrite theorem 2.6.3 for total functions as follows:

Theorem 2.7.3. *For every non-trivial pca A and every endofunction f on A, there is a pca $A[f]$ and an applicative morphism $\iota_f : A \to A[f]$ given by $a \mapsto \{a\}$ such that:*

(i) $f \in Q_1^{A[f]} = Q_1^{\iota_f}$

(ii) For every decidable applicative morphism $\gamma : A \to B$ such that $f \in Q^{\gamma}_1$, there is a decidable applicative morphism $\gamma_f : A[f] \to B$ such that $\gamma_f \iota_f = \gamma$, and γ is unique with this property. Moreover, if $\delta : A[f] \to B$ is such that $\delta \iota_f \cong \gamma$, then $\delta \cong \gamma_f$.

The construction presented in this section will allow us to prove the type 2 analogue of theorem 2.7.3. We define $A[F]$ as $A[g]$, where $g : A \to A$ is defined as a union of a family $\{g_\alpha\}_\alpha$ of compatible partial functions indexed by the ordinals. This family is defined by transfinite induction as follows:

- $g_0 = \emptyset$
- $g_{\alpha+1}(a) = b$ if a is an index for a total function $f : A \to A$ in $A[g_\alpha]$ and $F(f) = b$.
- $g_\lambda = \bigcup_{\alpha<\lambda} g_\alpha$ if λ is a limit ordinal.

Of course one has to check that the above yields a well-defined family of compatible functions $\{g_\alpha\}$, this is not hard. Now we let $g = \bigcup_\alpha g_\alpha$ and define $A[F] := A[g]$. We have the following theorem:

Theorem 2.7.4. *For every non-trivial pca A and every total functional $F : A^A \to A$, let $\iota_F : A \to A[F]$ be the applicative morphism given by $a \mapsto \{a\}$. Then:*

(i) $F \in Q_2^{A[F]} = Q_2^{\iota_F}$

(ii) For every decidable applicative morphism $\gamma : A \to B$ such that $F \in Q^{\gamma}_2$, there is a decidable applicative morphism $\gamma_F : A[F] \to B$ such that $\gamma_F \iota_F = \gamma$, and γ is unique with this property. Moreover, if $\delta : A[F] \to B$ is such that $\delta \iota_F \cong \gamma$, then $\delta \cong \gamma_F$.

Proof. Part (i) is easy to check.

For part (ii): Suppose $\gamma : A \to B$ is a decidable applicative morphism such that $F \in Q^{\gamma}_2$. Let $r_F \in I^{\gamma}_2(F)$. 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 $\mathsf{q}_0, \mathsf{q}_1$ 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

$$\begin{aligned} exv \simeq\ & \mathsf{if}\ \mathsf{dq}_0(re(\mathsf{c}xv))\ \mathsf{then}\ \mathsf{q}_1(re(\mathsf{c}xv)) \\ & \mathsf{else}\ ex(\mathsf{c}'v(r_F(r(\mathsf{q}_1(re(\mathsf{c}xv)))))). \end{aligned}$$

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

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

Essential is that if $a \in I_1^{A[F]}(f)$, then for $b \in \gamma(a)$, $rb \in I^{\gamma}_1(f)$.

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.

The other statements are easy to check. □

In the above theorem, we needed that F is total to guarantee that in $A[F]$, F will be an effective operation (so it is defined on all total functions in $A[F]$). However, one could also relax the definition of an effective operation to also include partial functionals. It is not hard to see that the above definition also holds for this definition, but for our purposes (in the next section, and section 5.3 and 5.4), we do not bother with it.

Question It is not known to me whether the above theorem holds for higher types, especially type 3. I strongly feel that it does not hold for type 3, but I haven't been able to come up with a counterexample yet.

2.8 A Generalization of Kleene's Computable Functionals

In this section we discuss another application of generalized relative recursion for pcas. We present a way in which it can be used to generalize a notion of *computable functionals* defined by Kleene in [Kle59]. As opposed to ordinary computable functions, computable functionals can take non-computable higher type objects as input. Note that in that way they also differ from effective operations, that only take computable objects as input. Higher-type recursion theory is a difficult subject and has been studied by a lot of recursion theorists. In contrast to ordinary recursion theory, there is no Church's thesis for computable functionals. In fact, various notions of higher-type computability have been studied, and they have been shown to be inequivalent. A great historical survey can be found in [Lon05]. Here we will focus on Kleene's notion, but several connections can be made to other notions, for example the Hereditarily Effective Operations (HEOs).

In Kleene computability, recursive functionals are $\mathbb{N}-$indexed functions that, besides type 0 arguments, also take arguments of higher types. As an example, imagine a "computable" partial functional φ that takes a single type 2 functional as its input. For a functional $F : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$, the value $\varphi(F)$ is determined by a procedure that can plausibly be called "algorithmic". In [Kle59], Kleene puts it as follows:

> "For a function $\varphi(F)$ of a type-2 variable F, it seems natural to image similarly an 'oracle' for F who at any step in the computation can be called upon to supply the value of $F(\beta)$ when a procedure for $\beta(y)$ for any y has arisen. (...) When the oracle for F is used, if the computations of $\beta(y)$ for each $y = 0, 1, \ldots$ are considered to be parts of the computation of $\varphi(F)$, that computation for the given F will not be a finite object (...)."

One might already observe the analogy with the construction of $A[F]$ for an arbitrary pca. We constructed $A[F]$ as $A[g]$ where g satisfies:

$$g(a) = b \text{ if } a \text{ is an index for } f \text{ in } A[g] \text{ and } F(f) = b.$$

In other words: in $A[g]$, we can ask for values of g at any step in the computation (using dialogues), which are values of F for functions f for which a "procedure has arisen", that is: we have an *index* of f in the pca $A[g]$ itself. It is this analogy that we use to generalize Kleene's notion of computability. The philosophy is that every pca defines a notion of what we consider "algorithmic", and the use of higher-type oracles is added on top of that.

Kleene defines his partial recursive functionals inductively in nine clauses that he named S1-S9. The definition below is taken from [Lon05], but essentially the same as in [Kle59]. Every partial recursive function has an index e, and the corresponding functional is denoted $\{e\}$. The arguments of a

partial recursive functional are tuples of functionals of a certain pure type. We write $\mathcal{X}$ for the disjoint union of these tuples. We will denote arbitrary tuples $(x_1,\dots,x_n) \in \mathcal{X}$ by $\boldsymbol{x}$, and for $\boldsymbol{\sigma} = (\sigma_1,\dots,\sigma_n)$ a tuple of pure types we write $\boldsymbol{x}:\boldsymbol{\sigma}$ to denote that $x_i : \sigma_i$ for each $1 \le i \le n$. For $\boldsymbol{\sigma} = (\sigma_1,\dots,\sigma_n)$ a tuple of pure types, we write $\#\boldsymbol{\sigma}$ for the coded sequences $\langle s_1,\dots,s_n\rangle \in \mathbb{N}$, where $\sigma_i = \overline{s_i}$. For $e, f \in \mathbb{N}$, we again write

$$\{e\}(\boldsymbol{x}) \simeq \{f\}(\boldsymbol{x}), \text{ resp. } \{e\}(\boldsymbol{x}) \precsim \{f\}(\boldsymbol{x})$$

to express that the left hand side is defined if and only if, (resp. if) the right hand side is, and in that case they are equal. Lastly, we use $\lambda-$notation to functionals: if y has type $\overline{t-2}$, then $\lambda y : \overline{t-2}.\{h\}(y,\boldsymbol{x})$ denotes the (partial) type $\overline{t-1}$ functional $y \mapsto \{h\}(y,\boldsymbol{x})$, for any $\boldsymbol{x}$. We now have the following definition:

Definition 2.8.1 (Kleene's S1-S9)**.** The partial map $\{-\}(-) : \mathbb{N} \times \mathcal{X} \to \mathbb{N}$ is defined by the following inductive clauses, that only apply when the expression is well-typed:

S1: Successor function: $\{\langle 1, \#\boldsymbol{\sigma}\rangle\}(\boldsymbol{x}:\boldsymbol{\sigma}) = x_1 + 1$

S2: Constant functions: For any $q \in \mathbb{N}$, $\{\langle 2, \#\boldsymbol{\sigma}, q\rangle\}(\boldsymbol{x}:\boldsymbol{\sigma}) = q$

S3: Projection: $\{\langle 3, \#\boldsymbol{\sigma}\rangle\}(\boldsymbol{x}:\boldsymbol{\sigma}) = x_1$

S4: Composition: For $g, h \in \mathbb{N}$:

$$\{\langle 4, \#\boldsymbol{\sigma}, g, h\rangle\}(\boldsymbol{x}:\boldsymbol{\sigma}) \simeq \{g\}(\{h\}(\boldsymbol{x}),\boldsymbol{x})$$

S5: Primitive recursion: For any $g, h \in \mathbb{N}$, if $m = \langle 5, \#\boldsymbol{\sigma}, g, h\rangle$, then for all $n \in \mathbb{N}$:

$$\{m\}(0,\boldsymbol{x}) \simeq \{g\}(\boldsymbol{x})$$
$$\{m\}(n+1,\boldsymbol{x}) \simeq \{h\}(n,\{m\}(n,\boldsymbol{x}),\boldsymbol{x}).$$

S6: Permutation of arguments: For any $g \in \mathbb{N}$ and $1 \le k < n$:

$$\{\langle 6, \#\boldsymbol{\sigma}, k, g\rangle\}(\boldsymbol{x}:\boldsymbol{\sigma}) \simeq \{g\}(x_{k+1}, x_1, \dots, x_k, x_{k+2}, \dots, x_n).$$

S7: Type 1 application: if $\sigma_1 = \overline{1}$ and $\sigma_2 = \overline{0}$:

$$\{\langle 7, \#\boldsymbol{\sigma}\rangle\}(\boldsymbol{x}:\boldsymbol{\sigma}) = x_1(x_2).$$

S8: Higher type application: For $h \in \mathbb{N}$ and $t \ge 2$:

$$\{\langle 8, \#\boldsymbol{\sigma}, t, h\rangle\}(y:\overline{t},\boldsymbol{x}:\boldsymbol{\sigma}) \simeq y(\lambda z : \overline{t-2}.\{h\}(y,z,\boldsymbol{x}))$$

S9: Index vocation:

$$\{\langle 9, \#\boldsymbol{\sigma}, \#\boldsymbol{\tau}\rangle\}(n:\overline{0},\boldsymbol{x}:\boldsymbol{\sigma},\boldsymbol{y}:\boldsymbol{\tau}) \simeq \{n\}(\boldsymbol{x}).$$

Observe that the closure rules S1-S8 yield a set of *total* computable functionals, which contains all primitive recursive functions. It is only due to S9 that partiality comes in.

We will state a couple of basic results on Kleene's functionals that we will need later on. They correspond to XIII-XV in [Kle59].

Theorem 2.8.2 (S^m theorem)**.** *For each $m \geq 1$ there is a primitive recursive function S^m in $m+1$ variables such that whenever z is an index for a recursive functional $\varphi(y_1 : \overline{0}, \ldots, y_n : \overline{0}, \boldsymbol{x} : \boldsymbol{\sigma})$, then*

$$\{S^m(z, y_1, \ldots, y_m)\}(\boldsymbol{x}) \simeq \{z\}(y_1, \ldots, y_m, \boldsymbol{x}) = \varphi(y_1, \ldots, y_n, \boldsymbol{x}).$$

Theorem 2.8.3 (Recursion theorem/Fixed-point theorem)**.** *Suppose $\psi(z : \overline{0}, \boldsymbol{x} : \boldsymbol{\sigma})$ is a partial recursive functional, then the equation*

$$\{z\}(\boldsymbol{x}) \simeq \psi(z, \boldsymbol{x})$$

has a solution for z.

Theorem 2.8.4 (Definition by cases)**.** *If*

$$\psi_0(\boldsymbol{x} : \boldsymbol{\sigma}), \psi_1(\boldsymbol{x} : \boldsymbol{\sigma})$$

are partial recursive functionals and $Q(\boldsymbol{x} : \boldsymbol{\sigma})$ is a partial recursive predicate (it takes values in $\{0,1\}$), then the functional ψ defined by

$$\psi(\boldsymbol{x}) \simeq \begin{cases} \psi_0(\boldsymbol{x}) & \text{if } Q(\boldsymbol{x}) \\ \psi_1(\boldsymbol{x}) & \text{if } \neg Q(\boldsymbol{x}) \end{cases}$$

is partial recursive.

2.8.1 Kleene Computability for Pcas

We now proceed to define a notion of "computable functional" for an arbitrary pca A. The indices of computable functionals are elements of A. In contrast to Kleene's S1-S9, we don't need to define explicit coding for the indices. The applicative structure on A will be enough to show that our functionals are closed under the analogues of S1-S9.

The goal is to define for any $a, b \in A$, and any tuple $(x_1, \ldots, x_k)$ of functionals the application of a to $(b, x_1, \ldots, x_k)$, which we will denote by $a \cdot^{x_1,\ldots,x_k} b$. By convention, we always require a type $\overline{0}$ argument, and insert $\top$ as a dummy whenever no such argument is provided. Again denote the type of each x_i by σ_i, and we work only with pure types, i.e. $\sigma_i = \overline{n_i}$ for some $n_i \in \mathbb{N}$. We write $a \cdot^{x_1,\ldots,x_k} b \downarrow$ to denote that $a \cdot^{x_1,\ldots,x_k} b$ is defined, $a \cdot^{x_1,\ldots,x_k} b = c$ to denote that $a \cdot^{x_1,\ldots,x_k} b$ is defined and has value c.

We start by defining a family of functions $g_{\alpha,\sigma_1,\ldots,\sigma_k} : A_{\sigma_1} \times \ldots A_{\sigma_k} \times A \to A$, where α ranges over the ordinals. We usually omit the subscript $_{\sigma_1,\ldots,\sigma_k}$ since that can be read-off from the arguments, i.e. we assume that

$$g_\alpha(x_1, \ldots, x_k, a) = g_{\alpha,\sigma_1,\ldots,\sigma_k}(x_1, \ldots, x_k, a)$$

in such a way that the expression is well-typed. For $(x_1, \ldots, x_k) \in A_{\sigma_1} \times \ldots A_{\sigma_k}$, $a \in A$, $1 \leq i \leq k$, let:

- $g_0 = \emptyset$
- $g_{\alpha+1}(x_1, \ldots, x_k, \mathsf{p}\overline{i}a) = x_i(a)$ if $\sigma_i = \overline{1}$
- $g_{\alpha+1}(x_1, \ldots, x_k, \mathsf{p}\overline{i}a) = x_i(y)$ if $\sigma_i = \overline{2}$, where $y = \lambda z.a \cdot^{\lambda v. g_\alpha(x_1,\ldots,x_k,v)} z$ is total
- $g_{\alpha+1}(x_1, \ldots, x_k, \mathsf{p}\overline{i}a) = x_i(y)$ if $\sigma_i = \overline{n}$ for some $n \geq 3$, where $y = \lambda z.a \cdot^{\lambda v.(g_\alpha(x_1,\ldots,x_k,z,v))} \top$ is total.
- $g_\lambda = \bigcup_{\alpha<\lambda} g_\alpha$ for λ a limit ordinal.

One should of course check that the above inductive definition actually works, and that for each $\sigma_1, \ldots, \sigma_k$, $\{g_{\alpha,\sigma_1,\ldots,\sigma_k}\}_\alpha$ is a compatible family of functions. It implies that the following is well-defined:

Definition 2.8.5. For all tuples of pure types $\sigma_1,\dots,\sigma_k$, The *associate of type* $\sigma_1,\dots,\sigma_k$ is the function $g_{\sigma_1,\dots,\sigma_k}$ defined by:

$$g_{\sigma_1,\dots,\sigma_k} = \bigcup_\alpha g_{\alpha,\sigma_1,\dots,\sigma_k}.$$

We will often write $g(x_1,\dots,x_k,v)$ instead of

$$g_{\sigma_1,\dots,\sigma_k}(x_1,\dots,x_k),$$

and write $g(x_1,\dots,x_k)$ for $\lambda v.g(x_1,\dots,x_k,v)$.

We now define:

$$a\cdot^{x_1,\dots,x_n} b = a\cdot^{g(x_1,\dots,x_n)} b.$$

Clearly, if $a\cdot^{x_1,\dots,x_n} b\downarrow$, then there is some α such that

$$a\cdot^{\lambda v.g_\alpha(x_1,\dots,x_n)} b\downarrow.$$

Clearly, the set of such α is upward closed.

Definition 2.8.6. For $\boldsymbol{x:\sigma}$ a tuple of functionals, we say that *a halts on b at stage α* if α is the least such that

$$a\cdot^{\lambda v.g_\alpha(x_1,\dots,x_n)} b\downarrow.$$

We will often need proofs by induction over the stage on which a term halts.
The following shows how Kleene's S1-S9 fits within the above model:

S1-S3: Successor function, Constant function, projection: Easy to see. Application in A has an index in $A[g]$, independent of g. In fact we gain something here: the indices of $A-$computable functions are independent of any higher type arguments that are also given.

S4: Composition: it is not hard to see that there is an index $c\in A$ such that for any $g:A\to A$:

$$(cae)\cdot^g b = a\cdot^g (e\cdot^g b).$$

Then cae satisfies for all tuples $(x_1,\dots,x_n)$:

$$cae\cdot^{x_1,\dots,x_n} b = a\cdot^{x_1,\dots,x_n}(e\cdot^{x_1,\dots,x_k} b).$$

S5: Primitive recursion: one should verify that there is $\mathcal{R}\in A$ such that for all $a,R\in A$, if we let $f=\mathcal{R}aR$, then for all $g:A\to A$, $n\in\mathbb{N}$:

$$f\cdot^g \overline{0} = a$$
$$f\cdot^g \overline{n+1} \simeq R\cdot^g \overline{n}\cdot^g (f\cdot^g \overline{n}).$$

This is easy using the proof of 2.3.3 in [vO08] (proposition 1.3.5 in that source). The definition of $\mathcal{R}$ only depends on the elements s,k in $A[g]$, which are independent of g. So $\mathcal{R}$ is defined by some term in which only the constants s,k appear, therefore we can apply proposition 2.6.5 to define $\mathcal{R}$ independent of g.

S6: Permutation of arguments: easy to see

S7: Type 1 application: If x_1 is of type $\overline{1}$, then if we let $a \in A$ be such that for all $b \in A, g : A \to A$:

$$a([b]) = \mathsf{pF}(\mathsf{p}\overline{1}b)$$
$$a([b, u_0]) = \mathsf{pT}u_0$$

Then $a \cdot^g b = g(\mathsf{p}\overline{1}b)$. Hence for such A:

$$a \cdot^{x_1,\ldots,x_k} b = x_1(b).$$

S8: Higher type application: Suppose we want to compute $x_i(\lambda z.e \cdot^{x_1,\ldots,x_k,z} \mathsf{T})$, and we are given that $\lambda z.e \cdot^{x_1,\ldots,x_k,z} \mathsf{T}$ is total. Let $a \in A$ be such that for all $b \in A, g : A \to A$:

$$a([b]) = \mathsf{pF}(\mathsf{p}\overline{i}b)$$
$$a([b, u]) = \mathsf{pT}u$$

Then by the above definitions:

$$a \cdot^{x_1,\ldots,x_k} e = x_i(y) \text{ where } y = \lambda z.e \cdot^{x_1,\ldots,x_k,z} \mathsf{T}$$

S9: Index invocation: easy to see.

The above tells us that any Kleene computable functional is computable with respect to the newly defined notion of "computable functional". The converse is also true:

Theorem 2.8.7. *For $A = \mathcal{K}_1$, the above defined notion of computability coincides with Kleene computability. In other words, for every $a \in A$, there is an index n such that for all $b \in A$ and functionals $(x_1 : \sigma_1, \ldots, x_k : \sigma_k)$:*

$$\{n\#\boldsymbol{\sigma}\}(b, x_1, \ldots, x_k) \precsim a \cdot^{x_1,\ldots,x_k} b.$$

Here $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_k)$.

Proof. We have to check that the partial map

$$(a, b, x_1, \ldots, x_k) \mapsto a \cdot^{x_1,\ldots,x_k} b$$

is computable in Kleene's sense. We therefore have to find an index n such that for any $\boldsymbol{x} : \boldsymbol{\sigma} = (x_1 : \sigma_1, \ldots, x_k : \sigma_n)$:

$$\{n\#\boldsymbol{\sigma}\}(a, b, \boldsymbol{x}) \preceq a \cdot^{x_1,\ldots,x_k} b$$

This will just be a tedious exercise. First, let χ be the following partial recursive function:

$$\chi(e, \#\boldsymbol{\sigma}, i, b, x_1, \ldots, x_k) =$$
$$\begin{cases} x_i(b) & \text{if } \sigma_i = \overline{1} \\ x_i(\lambda z.\{e\}(\#\boldsymbol{\sigma}, b, z, [], x_1, \ldots, x_k)) & \text{if } \sigma_i = \overline{2} \\ x_i(\lambda z.\{e\}(\#(\sigma_1, \ldots, \sigma_k, \sigma_i - 2), b, \mathsf{T}, [], x_1, \ldots, x_k, z)) & \text{if } \sigma_i = \overline{n} \\ & \text{for some } n \geq 3. \end{cases}$$

Convince yourself that χ is partial recursive by S7,S8,S9 and theorem 2.8.4.

Define the partial recursive function ψ as follows:

$$\psi(e,\#\boldsymbol{\sigma},a,b,u,x_1,\ldots,x_k)\simeq$$
$$\begin{cases}\mathsf{p}_1(a([b]*u)) & \text{if } \mathsf{p}_0(a([b]*u))\\ \{e\}(\#\boldsymbol{\sigma},a,b,u*[w],x_1,\ldots,x_k) & \text{else if } (\exists i\le k,v)\text{s.t. } \mathsf{p}_1(a([b]*u))=p\bar{i}v\\ & \text{where } w=\chi(e,\#\boldsymbol{\sigma},i,v,x_1,\ldots,x_k)\end{cases}$$

Again, convince yourself that ψ is indeed partial recursive.

By theorem 2.8.3, there is an index $\mathsf{app}\in\mathbb{N}$ such that:

$$\{\mathsf{app}\}(\#\boldsymbol{\sigma},a,b,u,x_1,\ldots,x_k)\simeq\psi(\mathsf{app},\#\boldsymbol{\sigma},a,b,u,x_1,\ldots,x_k).$$

I'll now prove by induction on α that for all $(x_1:\sigma_1,\ldots,x_k:\sigma_k)$:

$$a\cdot^{g_\alpha(x_1,\ldots,x_k)}b\downarrow\Rightarrow\{\mathsf{app}\}(a,b,[],x_1,\ldots,x_k)=a\cdot^{g_\alpha(x_1,\ldots,x_k)}b.$$

Here we write $g_\alpha(x_1,\ldots,x_k)$ for $\lambda v.g_\alpha(x_1,\ldots,x_k,v)$.

Recall that $a\cdot^{g_0(x_1,\ldots,x_k)}b\downarrow$ implies that $a([b])=\mathsf{p}\mathsf{T}c$ for some c. In that case:

$$\psi(\mathsf{app},\#\boldsymbol{\sigma},a,b,[],x_1,\ldots,x_k)=\psi(\mathsf{app},\#\boldsymbol{\sigma},a,b,[],x_1,\ldots,x_k)=c.$$

Now suppose the statement holds for all $\beta<\alpha$. Then $a\cdot^{g_\alpha(x_1,\ldots,x_k)}b=c$ implies that there is a $g_\alpha(x_1,\ldots,x_k)$-dialogue $u=[u_0,\ldots,u_n]$ between a and b such that $a([b]*u)=\mathsf{p}\mathsf{T}c$. By reverse induction, I'll show that for all $0\le i\le n+1$

$$\psi(\mathsf{app},\#\boldsymbol{\sigma},a,b,u^{<i},x_1,\ldots,x_k)=c$$

For $i=n+1$ this is clear, since $a([b]*u)=\mathsf{p}\mathsf{T}c$. Now suppose $i<n+1$. Then $\mathsf{p}_1(a([b]*u^{<i}))=\mathsf{p}\bar{j}v$ for some $v,1\le j\le k$, and:

$$\psi(\mathsf{app},\#\boldsymbol{\sigma},a,b,u^{<i},x_1,\ldots,x_k)=$$
$$\{\mathsf{app}\}(\#\boldsymbol{\sigma},a,b,u*[w],x_1,\ldots,x_k)\text{ where}$$
$$w=\chi(\mathsf{app},\#\boldsymbol{\sigma},i,v,x_1,\ldots,x_k).$$

We know that $u_i=g_\alpha(x_1,\ldots,x_k,\mathsf{p}\bar{j}v)$. We now have three cases:

- $\sigma_j=\overline{1}$, then $w=x_j(a)=g_\alpha(\mathsf{p}\bar{j}v)=u_i$.
- $\sigma_j=\overline{2}$, then $w=x_j(\lambda z.\{\mathsf{app}\}(\#\boldsymbol{\sigma},v,z,[],x_1,\ldots,x_k))$
- $\sigma_j=\overline{n}$ for some $n\ge 3$, then

$$w=x_j(\lambda z.\{\mathsf{app}\}(\#(\sigma_1,\ldots,\sigma_k,\sigma_{j-2}),v,\mathsf{T},[],x_1,\ldots,z,\ldots,x_k)).$$

In the third case, we know that $g_\alpha(x_1,\ldots,x_k,\mathsf{p}\bar{j}v)$ is defined, whence we deduce that $\lambda z.v\cdot^{g_\beta(x_1,\ldots,x_k,z)}\mathsf{T}$ is total for some $\beta<\alpha$. By induction hypothesis, we have that for all z:

$$v\cdot^{g_\beta(x_1,\ldots,x_k,z)}\mathsf{T}=\{\mathsf{app}\}(\#(\sigma_1,\ldots,\sigma_k,\sigma_{j-2}),v,\mathsf{T},[],x_1,\ldots,z,\ldots,x_k)).$$

So $w=g_\alpha(x_1,\ldots,x_k,\mathsf{p}\bar{j}v)=u_i$, hence

$$\psi(\mathsf{app},a,b,u^{<i},x_1,\ldots,x_k)=\{\mathsf{app}\}(\#\boldsymbol{\sigma},a,b,u*[u_i],x_1,\ldots,x_k)=a\cdot^{g_\alpha(x_1,\ldots,x_k)}b$$

by the reverse induction hypothesis.

For the second case we have roughly the same argument and the first case is clear. Hence it follows that

$$\{\mathsf{app}\}(\#\boldsymbol{\sigma}, a, b, [], x_1, \ldots, x_k) = a \cdot^{g_\alpha(x_1,\ldots,x_k)} b$$

whenever the right hand side is defined. This finishes the inductive step.

It then follows that for all $(x_1 : \sigma_1, \ldots, x_k : \sigma_k)$:

$$\{\mathsf{app}\}(\#\boldsymbol{\sigma}, a, b, [], x_1, \ldots, x_k) \precsim a \cdot^{x_1,\ldots,x_k} b.$$

By theorem 2.8.2 we can find for every a an index n such that

$$\{n\#\boldsymbol{\sigma}\}(b, x_1, \ldots, x_k) \precsim a \cdot^{x_1,\ldots,x_k} b.$$

□

One may wonder whether we can obtain the above result with "$\simeq$" instead of "$\precsim$". This is a very subtle issue, and to be able to prove such a thing we would have to look at the definition of Kleene's functionals a bit more carefully. However, it is not very important for the rest of this thesis. What is important here is that either system can simulate the other, and that therefore the set of total computable functionals is the same in either case.

2.8.2 Relative Recursion and Hierarchies

In Kleene computability, a function $f : \mathbb{N} \to \mathbb{N}$ is computable with respect to the functionals $\boldsymbol{x} : \boldsymbol{\sigma}$ if there exists $e \in \mathbb{N}$ such that:

$$(\forall n)\{e\}(n, \boldsymbol{x}) = f(n).$$

Sets of functions computable from certain functionals are also known as (recursive) *hierarchies*. Using the previous section, we can define hierarchies for any partial combinatory algebra A. In fact, if we denote by $A[x_1, \ldots, x_n]$ the partial combinatory algebra with applicative map $\cdot^{x_1,\ldots,x_n}$, then the computable functions in $A[x_1, \ldots, x_n]$ are precisely the functions computable from $x_1, \ldots, x_n$. We have immediately obtained the additional result that *all hierarchies are pcas.* In the rest of this section we will look closely at a specific hierarchy that we will need later on.

The first was already described by Kleene in [Kle59]. Define the type 2 functional $E : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ as follows:

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

Definition 2.8.8. A total functional $\Phi : \mathbb{N} \times \mathbb{N}_{\sigma_1} \times \mathbb{N}_{\sigma_2} \times \ldots \times \mathbb{N}_{\sigma_k} \times \mathbb{N} \to \mathbb{N}$ is a *predicate* if it takes values in $\{0, 1\}$. Here we take 0 to represent the truth value "true", and 1 "false".

A *recursive predicate* is a predicate $\Phi : \mathbb{N}_{\sigma_1} \times \mathbb{N}_{\sigma_2} \times \ldots \times \mathbb{N}_{\sigma_k} \times \mathbb{N} \to \mathbb{N}$ for which there exists an index $e \in \mathbb{N}$, so that for all $\boldsymbol{x} : \boldsymbol{\sigma}$:

$$e \cdot^{x_1,\ldots,x_n} v = \Phi(x_1, \ldots, x_n, v).$$

With the above definition we might as well define E as:

$$E(f) = \big((\exists m) f(m) = 0\big).$$

Kleene has shown that the total functions computable from E are precisely the Δ^1_1 functions. We define this set of functions as part of the *Analytical Hierarchy*:

Definition 2.8.9 (Analytical Hierarchy)**.** Let $k \geq 0$. Let $\mathcal{F} : \mathbb{N}_{\sigma_1} \times \ldots \mathbb{N}_{\sigma_k} \times \mathbb{N} \to \{0,1\}$ be predicate. Then we define inductively for all $n \geq 1$:

- We say that $\mathcal{F}$ is Π^1_1 if there exists $e \in \mathbb{N}$ such that e is an index for a recursive predicate and
$$\mathcal{F}(n, x_1, \ldots, x_k) = (\forall g : \mathbb{N}^{\mathbb{N}})(\exists y) e^{\cdot^{g, x_1, \ldots, x_n}}[y, n].$$
- Similarly, $\mathcal{F}$ is called Σ^1_1 if there exists $e \in \mathbb{N}$ such that e is an index for a recursive predicate and
$$\mathcal{F}(n, x_1, \ldots, x_k) = (\exists g : \mathbb{N}^{\mathbb{N}})(\forall y) e^{\cdot^{g, x_1, \ldots, x_k}}[y, n].$$
- $\mathcal{F}$ is Π^1_{n+1} if there exists a Σ^1_n predicate $\mathcal{G} : \mathbb{N}^{\mathbb{N}} \times \mathbb{N}_{\sigma_1} \times \ldots \mathbb{N}_{\sigma_k} \times \mathbb{N} \to \mathbb{N}$ such that
$$\mathcal{F}(n, x_1, \ldots, x_k) = (\forall g : \mathbb{N}^{\mathbb{N}}) \mathcal{G}(n, g, x_1, \ldots, x_k).$$
- $\mathcal{F}$ is Σ^1_{n+1} if there exists a Π^1_n predicate $\mathcal{G} : \mathbb{N}^{\mathbb{N}} \times \mathbb{N}_{\sigma_1} \times \ldots \mathbb{N}_{\sigma_k} \times \mathbb{N} \to \mathbb{N}$ such that
$$\mathcal{F}(n, x_1, \ldots, x_k) = (\exists g : \mathbb{N}^{\mathbb{N}}) G(n, g, x_1, \ldots, x_k).$$
- $\mathcal{F}$ is Δ^1_n if it is both Π^1_n and Σ^1_n.

Call a subset $A \subseteq \mathbb{N}$ Π^1_n, Σ^1_n resp. Δ^1_n if its characteristic function is given by a Π^1_n, Σ^1_n resp. Δ^1_n predicate. We call a partial function $f : \mathbb{N} \to \mathbb{N}$ Π^1_n, Σ^1_n resp. Δ^1_n if its graph, coded as a subset of $\mathbb{N}$, is $\Pi^1_n, \Sigma^1_n, \Delta^1_n$.

We call a set, or function *analytical* if it is Π^1_n or Σ^1_n for some n.

Remark It is an easy exercise to show that a total function $f : \mathbb{N} \to \mathbb{N}$ is Π^1_n (or Σ^1_n) if and only if it is Δ^1_n.

Theorem 2.8.10. *The total functions in $\mathcal{K}_1[E]$ are precisely the total Δ^1_1 functions.*

Proof. See [Kle59], XLVIII (and theorem 2.8.7). An alternative approach is given in [Hin78], theorem VI.1.8. □

In [Hin78], Peter Hinman proves a stronger result than the above theorem. Namely, he shows that the domains of the partial functions recursive in E (in his sense, but they are essentially the same), are precisely the Π^1_1 sets. I will state his result here, formulated in way that I consider more appropriate for this context. Recall the following definition from ordinary recursion theory:

Definition 2.8.11. A set $C \subseteq \mathbb{N}$ is Π^1_1-*complete* if it is Π^1_1 and for every Π^1_1 set $X \subseteq \mathbb{N}$, there is a total computable function $F : \mathbb{N} \to \mathbb{N}$ such that
$$x \in X \iff F(x) \in C.$$

An example of a Π^1_1 complete set is the following:
$$W = \{z \mid \varphi_z \text{ codes a well-ordering on } \mathbb{N}\}. \tag{2.4}$$

Where φ_z denotes the partial computable function with index z for some model of computation, and a total function $f : \mathbb{N} \to \mathbb{N}$ *codes a well-ordering* if the ordering
$$n \leq m \iff f(\langle n, m \rangle) = 0$$
is a well-ordering on $S = \{n \in \mathbb{N} \mid (\exists m) f(\langle n, m \rangle) = 0 \vee f(\langle m, n \rangle) = 0\}$.

I now wrap up Hinman's result in the following theorem:

Theorem 2.8.12. *The partial function $(a,b) \mapsto a \cdot^E b$ (the application in $\mathcal{K}_1[E]$) is Π^1_1-complete.*

Proof. I'll 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 $\mathcal{K}_1[f]$, for any f. Recall that $\mathcal{K}_1[E] = \mathcal{K}_1[g]$, where we defined g in stages. I'll show that g is Π^1_1. Let e be an index such that whenever F is the characteristic function of the graph of a partial function $f : A \to A$, then for all $b, x \in \mathcal{K}_1$:

$$eb \cdot^F x \simeq b \cdot^f x.$$

Convince yourself 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:

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

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 I claim:

$$G([b,i]) \iff (\forall X)(\exists p,j)(\Phi([p,j],X) \Rightarrow X([p,j])) \Rightarrow X([b,i]). \tag{2.5}$$

Proof of claim. First, observe:

$$\begin{aligned}\Phi([b,i],G) &\Leftrightarrow (\forall s)(\exists t \forall y) eb \cdot^G s\downarrow \wedge (eb \cdot^G t = 0 \vee i = 1) \wedge (eb \cdot^G y \neq 0 \vee i = 0)\\ &\Leftrightarrow (\forall s)(\exists t \forall y) b \cdot^g s\downarrow \wedge (b \cdot^g t = 0 \vee i = 1) \wedge (b \cdot^g y \neq 0 \vee i = 0)\\ &\Leftrightarrow G([b,i]).\end{aligned}$$

This proves the "$\Leftarrow$" direction of (2.5) (we substitute G for X).

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

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

Suppose we have X such that

$$(\forall p,j) : \Phi([p,j],X) \Rightarrow X([p,j]).$$

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

$$(\forall p,j) : G([p,j]) \Rightarrow X([p,j]). \tag{2.6}$$

This is done by induction. Recall that g is constructed as $g = \bigcup_\alpha g_\alpha$. Denote the corresponding graphs by G_α. Then for $\alpha = 0$,

$$(\forall p,j) : G([p,j]) \Rightarrow X([p,j])$$

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

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

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

Hence we have shown (2.6), from which $X[b,i]$ follows by assumption. □

Since the right-hand side of (2.5) is Π^1_1, we are done.

To see that the application is Π^1_1 complete, see [Hin78], section VI.1. There one can find a proof in which the set W from (2.4) is reduced to a *semi-recursive* set computable from E in Hinman's sense. This proof is easily adopted to a proof of Π^1_1-completeness for our case. □

Another hierarchy is given by E's bigger brother of type 3:

$$\mathbb{E}(F) = \begin{cases} 0 & \text{if } (\exists \alpha) F(\alpha) = 0 \\ 1 & \text{else .} \end{cases}$$

Hinman studies this functional in more detail. For now I'll leave it at the following proposition:

Proposition 2.8.13 (Hinman)**.** $\mathbb{E}$ *forces the analytical hierarchy. That is, every analytical set is computable in* $\mathcal{K}_1[\mathbb{E}]$.

Proof. First, $\mathbb{E}$ computes E since for any function f, we can define the functional

$$F : g \mapsto f(g(0)).$$

computable from f. So then

$$E(f) = \mathbb{E}(F).$$

It follows that E is an effective operation in $\mathcal{K}_1[\mathbb{E}]$. By theorem 2.7.4, $n \mapsto \{n\}$ is a decidable applicative morphism

$$\mathcal{K}_1[E] \to \mathcal{K}_1[\mathbb{E}]$$

hence $\mathcal{K}_1[\mathbb{E}]$ computes all Δ^1_1 sets.

Now suppose e is an index of a recursive predicate $\mathbb{N}^{\mathbb{N}} \to \mathbb{N}$. Then

$$(\forall y) e \cdot^g [y, n]$$

is Δ^1_1 so there is f such that

$$f n \cdot^{\mathbb{E},g} \mathsf{T} = (\forall y) e \cdot^{g,\mathbb{E}} [y, n]$$

But then

$$(\exists g)(\forall y) e \cdot^g [y, n] \iff r_{\mathbb{E}} \cdot^{\mathbb{E}} (f n)$$

where $r_{\mathbb{E}}$ represents $\mathbb{E}$ in $\mathcal{K}_1[\mathbb{E}]$. It follows that Σ^1_1 predicates are computable in $\mathcal{K}_1[E]$, and, by complementation, Π^1_1 predicates as well.

The inductive step is similar. □

2.9 Lifting Applicative Morphisms

In this section I'll describe a procedure that I call *lifting*. For the remainder of this section, $\gamma : A \to B$ is a computationally dense, discrete applicative morphism $A \to B$. Note that if A is decidable, any computationally dense applicative morphism $\gamma : A \to B$ is discrete. By theorem 2.5.16, there exists $\delta : B \to A$ such that $\gamma\delta \preceq \iota_B$ (so by theorem 2.5.15, δ is also discrete).

Now define a partial function $f_\delta : A \to A$ by:

$$f_\delta(a) = a' \iff a \in \delta\gamma(a').$$

Since γ is discrete, f_δ is well-defined. Since $\gamma\delta \preceq \iota_B$, f_δ is representable with respect to γ. Since γ is decidable (proposition 2.5.12), γ is realized as applicative morphism $A[f_\delta] \to B$. By composition with $\iota_f : A \to A[f_\delta]$, we can view δ as an applicative morphism $B \to A[f_\delta]$.

Suppose f_δ is represented by $r \in A[f_\delta]$. Then r realizes $\delta\gamma \preceq \iota_{A[f_\delta]}$. But then by theorem 2.5.16, δ is computationally dense!

Definition 2.9.1. Define the *lift* of γ with respect to δ to be γ viewed as morphism $A[f_\delta] \to B$.

In [Lon95], an applicative morphism $\gamma : A \to B$ is an *inclusion* if there exists $\delta : B \to A$ such that $\gamma \dashv \delta$ and $\gamma\delta \simeq \iota_B$. Similarly, $\gamma : A \to B$ is a *retraction* if $\gamma \dashv \delta$ and $\delta\gamma \simeq \iota_A$.

The theorem below tells us that we can lift a discrete inclusion to an equivalence.

Theorem 2.9.2. *Suppose $\gamma : A \to B$ is discrete, and $\delta : B \to A$ is such that $\gamma \dashv \delta$ and $\gamma\delta \simeq \iota_B$. Then lift of γ with respect to δ is an equivalence.*

Proof. Observe that we get $\gamma : A[f_\delta] \to B$, $\gamma\delta \simeq \iota_B$ still holds, and $\iota_{A[f_\delta]} \preceq \delta\gamma$ also still holds since this had a realizer in A. If r represents f_δ in $A[f_\delta]$, then r realizes $\delta\gamma \preceq \iota_{A[f_\delta]}$.

Hence this lift is an equivalence. □

Effectively quasi-surjective applicative morphisms (corollary 2.5.17) lift to retractions (if they are discrete):

Theorem 2.9.3. *Suppose $\gamma : A \to B$ is a discrete applicative morphism and $\delta : B \to A$ is such that $\gamma\delta \simeq \iota_B$. Then the lift of γ with respect to δ yields a discrete retraction $\delta : B \to A[f_\delta]$.*

Proof. For the lift $\gamma : A[f_\delta] \to B$, $\delta\gamma \preceq \iota_A$, so δ is computationally dense and $\gamma\delta \simeq \iota_B$ still holds. Therefore $\delta \dashv \gamma$, so δ is a discrete retraction. □

Chapter 3

Assemblies

In the introduction, we have informally described a category of assemblies. In this chapter, we will give a definition and exhibit the structure of such a category. It turns out that they are very rich categories with interesting properties. Assemblies are also studied in computer science, for example in domain theory.

3.1 The Category of Assemblies

Every partial combinatory algebra gives rise to a category of *assemblies*. Informally, it looks like the category of sets, but we require the morphisms to be representable functions of A in a certain sense. We start with the definition of an assembly:

Definition 3.1.1. An *assembly* on a pca A is a pair (X,E), where X is a set and $E : X \to A$ is a total relation. We write $E(x)$ for the set $\{a \in A \mid (x,a) \in E\}$.

A morphism between assemblies $(X,E) \to (Y,F)$ is given by a map $f : X \to Y$ for which there is an $a \in A$ such that:

$$\forall x\colon\ \forall b \in E(x), ab\downarrow \text{ and } ab \in F(f(x)).$$

For an assembly (X,E) and $x \in X$, we can view $E(x)$ as the set of representatives of the element x in A. Note that an assembly is basically just a map $E : X \to \mathcal{P}(A)$. We therefore refer to the underlying set X of an assembly (X,E) as the *domain* of (X,E).

A morphism of assemblies $(X,E) \to (Y,F)$ given by a map $f : X \to Y$ is then a representable function of A that maps every representative of x to a representative of $f(x)$. In other words, there is an element $a \in A$ such that for all $a' \in E(x)$, $aa' \in F(f(x))$. The element a is also said to *track* f.

It is easy to check that morphisms between assemblies are closed under composition. Therefore assemblies on a pca A form a category, denoted $\mathrm{Ass}(A)$. From the proposition below we can already see that the category $\mathrm{Ass}(A)$ has a lot of nice properties.

Definition 3.1.2. A category $\mathcal{C}$ is called *regular* if the following conditions hold:

1. $\mathcal{C}$ is finitely complete
2. We have coequalizers of kernel pairs, i.e. for any arrow $f : C \to D$ whenever we have a pullback diagram

$$\begin{array}{ccc} E \times_D E & \xrightarrow{e_0} & C \\ \downarrow{\scriptstyle e_1} & & \downarrow{\scriptstyle f} \\ C & \xrightarrow{f} & D \end{array}$$

then the coequalizer of $\langle e_0, e_1\rangle : E \times_D E \to C \times C$ exists.

3. Regular epimorphisms (i.e. epimorphism that are coequalizers) are stable under pullback. That is, whenever $e : C \to D$ is a regular epimorphism and

$$\begin{array}{ccc} E \times_D C & \longrightarrow & C \\ \downarrow{\scriptstyle e'} & & \downarrow{\scriptstyle e} \\ E & \xrightarrow{\ f\ } & D \end{array}$$

is a pullback diagram, e' is regular epi.

Regular categories are very important in logic. In such categories, the collection of subobjects has a well-behaved structure that allows for an interpretation of a fragment of first-order logic (called *regular logic*), and often more than that. One important property of regular categories is that every arrow $f : C \to D$ has a regular epi-mono decomposition. This means that every such f decomposes as:

$$f = me$$

with e regular epi, and m mono. One can show that such a decomposition is unique (up to isomorphism), and we call the subobject associated to m the *image* of f. In what follows, we will make extensive use of regularity of certain categories. For more on regular categories, see [BGO71].

Next, we recall the definition of a cartesian closed category.

Definition 3.1.3. A category $\mathcal{C}$ is *cartesian closed* if it has binary products and for every $X \in \mathcal{C}$, the product functor $(-) \times X : \mathcal{C} \to \mathcal{C}$ has a right adjoint $(-)^X : \mathcal{C} \to \mathcal{C}$.

For any Y, the object Y^X behaves like the object of "morphisms from X to Y". In what follows, we will often explicitly compute these objects, that are also called *exponentials*.

The following proposition summarizes some properties of a category of assemblies that are important for the rest of this chapter. For a proof of the following I refer to [vO08] or [Lon95].

Proposition 3.1.4. *For a pca A, the category $Ass(A)$ is regular, cartesian closed and has finite colimits.*

We will see in section 3.1.1 that $\mathrm{Ass}(A)$ is also equipped with a *natural numbers object*. For more categorical properties of $\mathrm{Ass}(A)$, see section 4.1.

Instead of proving the above proposition, I'll quickly sketch some constructions that we will need later. For $(X, E), (Y, F) \in \mathrm{Ass}(A)$, we can form the product by the assembly $(X \times Y, E \wedge F)$ where we denote by $E \wedge F$ the map

$$(x, y) \mapsto \{\mathsf{p}ab \mid a \in E(x), b \in F(y)\}.$$

It is easy to see that any singleton set $\{*\}$, together with any nonempty subset of A yields a terminal object.

The coproduct of $(X, E), (Y, F)$ is given by $(\{0\} \times X \cup \{1\} \times Y, E \vee F)$ where

$$E \vee F(0, x) = \{\mathsf{p}\mathsf{T}a \mid a \in E(x)\} \text{ and } E \vee F(1, x) = \{\mathsf{p}\mathsf{F}b \mid b \in F(y)\}$$

Note that the use of T and F is purely to be able to tell what kind of element the representative represents, so an element from X or from Y. We can use any pair of elements here that we can distinguish in A.

Exponentials can be defined as follows. For assemblies (X, E), (Y, F), the exponential $(Z, G) = (Y, F)^{(X,E)}$ has as domain Z the set of morphisms $f : X \to Y$ in $\mathrm{Ass}(A)$, and $G : Z \to A$ is defined by:

$$G(f) = \{a \mid a \text{ tracks } f\}.$$

For $P, Q \subseteq A$, we define a set $P \to Q$ by:

$$P \to Q = \{a \in A \mid (\forall p \in P)\, ap \downarrow \wedge ap \in Q\}.$$

For instance, the function G can be written as:

$$G(f) = \bigcap_{x \in X} E(x) \to F(f(x)).$$

Similarly, we define analogously to the above:

$$P \wedge Q = \{\mathsf{p}pq \mid p \in P, q \in Q\}$$

and:

$$P \vee Q = \{\mathsf{p}\mathsf{T}p \mid p \in P\} \cup \{\mathsf{p}\mathsf{F}q \mid q \in Q\}.$$

The following two propositions are useful to keep in mind and easy to prove.

Proposition 3.1.5. *A morphism $m : (X, E) \to (Y, F)$ is a monomorphism if and only if it is injective as a function $X \to Y$.*

Proposition 3.1.6. *A morphism $e : (X, E) \to (Y, F)$ is an epimorphism if and only if it is surjective as a function $X \to Y$.*

3.1.1 Natural Numbers Object

We recall the definition of a natural numbers object (nno) in a category with a terminal object:

Definition 3.1.7. Let $\mathcal{C}$ be a category with terminal object 1. A *natural numbers object* in $\mathcal{C}$ consists of an object $N \in \mathcal{C}$, and arrows $1 \xrightarrow{0} N \xrightarrow{S} N$ such that whenever we have arrows $1 \xrightarrow{x_0} X \xrightarrow{f} X$, there exists a unique $\varphi : N \to X$ that makes the following diagram commute:

$$\begin{array}{ccccc} 1 & \xrightarrow{0} & N & \xrightarrow{S} & N \\ & \searrow^{x_0} & \downarrow{\scriptstyle \varphi} & & \downarrow{\scriptstyle \varphi} \\ & & X & \xrightarrow{f} & X \end{array}.$$

In the above definition, one can view $\varphi : N \to X$ as being defined by induction. It is easy to check that a natural numbers object is unique up to isomorphism.

For a non-trivial pca A, we can define the assembly $(\mathbb{N}, N)$ by $N(n) = \{\overline{n}\}$, where $\overline{n}$ is defined as in definition 2.3.1. The map $S : \mathbb{N} \to \mathbb{N}$ given by $S(n) = n+1$ can be realized as a morphism $(\mathbb{N}, N) \to (\mathbb{N}, N)$ by proposition 2.3.5.

Proposition 3.1.8. *$(\mathbb{N}, N)$, together with the morphisms $0 : 1 \to (\mathbb{N}, N)$ given by $0(*) = 0$ and S, is a natural numbers object.*

Proof. Let (X, E) be an assembly with maps $x_0 : 1 \to (X, E)$ and $f : (X, E) \to (X, E)$. By induction, there is a map $\varphi : \mathbb{N} \to X$ such that

$$\varphi \circ S(n) = \varphi(n+1) = f(\varphi(n)).$$

Let $a_0 \in E(x_0(*))$. Suppose f is realized by $e \in A$. Now if we let $R \in A$ be such that

$$Rxy = ey,$$

then $g := \mathcal{R}a_0R$ for $\mathcal{R}$ the recursion operator from 2.3.3 satisfies:

$$g\overline{0} = a_0$$
$$g\overline{n+1} = e(g\overline{n}).$$

Therefore g realizes φ as a morphism $(\mathbb{N},N) \to (X,E)$. Uniqueness of φ is clear, therefore $(\mathbb{N},N)$ is a natural numbers object. □

Proposition 3.1.9. *There are at most $|A|$ many arrows $(X,E) \to (\mathbb{N},N)$ for any $(X,E) \in Ass(A)$. Therefore, if A is countable, the $\mathbb{N}-$fold coproduct of 1 does not exist. In general, the $|A|$-fold coproduct of 1 does not exist.*

Proof. A morphism $f : (X,E) \to (\mathbb{N},N)$ is fully determined by its realizer $e \in A$, since $f(x) = n \iff (\forall a \in E(x))ea = \overline{n}$. Therefore there are at most $|A|$ many such arrows.

Now suppose that A is countable. If the $\mathbb{N}-$fold coproduct of 1 would exist, then for every map $f : \mathbb{N} \to \mathbb{N}$ there is a unique morphism

$$\coprod_{n\in\mathbb{N}} 1 \to (\mathbb{N},N)$$

sending the $i-$th summand to $f(i)$. In that case there would be uncountably many arrows $\coprod_{n\in\mathbb{N}} 1 \to (\mathbb{N},N)$. The general statement is similar. □

3.2 The Constant Objects Functor

The following definitions turn out to be useful later on.

Definition 3.2.1 ($\nabla-$category)**.** A ∇-*category* is is a category $\mathcal{C}$ together with a regular functor

$$\nabla : \mathrm{Set} \to \mathcal{C}.$$

A ∇-functor between ∇-categories $(\mathcal{C},\nabla_C)$ and $(\mathcal{D},\nabla_D)$ is a functor $F : \mathcal{C} \to \mathcal{D}$ such that there is an isomorphism

$$F \circ \nabla_C \cong \nabla_D.$$

We write ∇Cat for the sub 2-category of Cat consisting of $\nabla-$categories, ∇-functors and natural transformations. We write ∇Reg for the sub 2-category of Cat consisting of $\nabla-$categories, regular ∇-functors and natural transformations.

Definition 3.2.2 (Γ-category)**.** A Γ-*category* is is a category $\mathcal{C}$ together with a regular functor

$$\Gamma : \mathrm{Set} \to \mathcal{C}.$$

A Γ-functor between Γ-categories $(\mathcal{C},\Gamma_C)$ and $(\mathcal{D},\Gamma_D)$ is a functor $F : \mathcal{C} \to \mathcal{D}$ such that there is an isomorphism

$$F \circ \Gamma_C \cong \Gamma_D.$$

We write ΓCat for the sub 2-category of Cat consisting of $\Gamma-$categories,Γ-functors and natural transformations. We write ΓReg for the sub 2-category of Cat consisting of $\Gamma-$categories, regular Γ-functors and natural transformations.

Definition 3.2.3 ($\Gamma\nabla-$category)**.** A $\Gamma\nabla$-*category* is is a category $\mathcal{C}$ together with an adjoint pair of functors $\Gamma \dashv \nabla$ such that $(\mathcal{C}, \nabla)$ is a $\nabla-$category, $(\mathcal{C}, \Gamma)$ is a Γ-category and

$$\Gamma \circ \nabla \cong \mathrm{Id}.$$

A $\Gamma\nabla$-functor F between $\Gamma\nabla$-categories $(\mathcal{C}, \nabla_C, \Gamma_C)$ and $(\mathcal{D}, \nabla_D, \Gamma_D)$ is both a ∇-functor $F : (\mathcal{C}, \nabla_C) \to (\mathcal{D}, \nabla_D)$ and a Γ-functor $F : (\mathcal{C}, \Gamma_C) \to (\mathcal{D}, \Gamma_D)$.

We write $\Gamma\nabla$Cat for the sub 2-category of Cat consisting of $\Gamma\nabla-$categories, $\Gamma\nabla$-functors and natural transformations. We write $\Gamma\nabla$Reg for the sub 2-category of Cat consisting of $\Gamma\nabla-$categories, regular $\Gamma\nabla$-functors and natural transformations.

There exists an embedding Set $\to$ Ass(A): the *constant objects functor*. The proof of the following proposition is easy.

Proposition 3.2.4 (Constant objects functor)**.** *Let $\nabla : Set \to Ass(A)$ be the map such that for $X \in Set$,*

$$\nabla(X) = (X, E)$$

where $E(x) = A$ for all $x \in X$.

Then for $f : X \to Y$ a map of sets, f is realized as a morphism $\nabla(X) \to \nabla(Y)$. Therefore ∇ is a full and faithful functor between categories. Moreover, ∇ is regular.

Therefore every category of assemblies is an ∇-category in a canonical way.

Proposition 3.2.5. *The functor $\Gamma : Ass(A) \to Set$, given by $\Gamma(X, E) = X$, is a regular functor and left adjoint to ∇. Hence every category of assemblies is a $\Gamma\nabla$-category in a canonical way.*

Proof. Let $(X, E) \in \mathrm{Ass}(A)$. Every function $f : X \to Y$ is realized as a morphism $(X, E) \to \nabla(Y)$, by any total element of A. Naturality is easy to check. □

Remark Whenever there is a chance of confusion, we denote $\nabla : \mathrm{Set} \to \mathrm{Ass}(A)$, $\Gamma : \mathrm{Ass}(A) \to \mathrm{Set}$ by ∇_A, Γ_A.

3.3 Morphisms Between Categories of Assemblies

In this section, we study the relation between the category of partial combinatory algebras and the full subcategory of ΓCat (definition 3.2.2) consisting of the assemblies. The idea is that on the level of sets, a Γ-functor is equivalent to the identity. Therefore a Γ-functor expresses a relation between the underlying pcas, rather than the sets represented in them. It turns out that it is exactly that what characterises Γ-functors between categories of assemblies; we will see below that up to isomorphism, they arise from applicative morphisms.

The following lemma by Jaap van Oosten ([vO08], proposition 1.6.1) makes this even more explicit. We start with a definition.

Definition 3.3.1. An functor $F : \mathrm{Ass}(A) \to \mathrm{Ass}(B)$ is called an $S-$functor if it is the identity on the level of sets, i.e.

$$\Gamma_B \circ F = \Gamma_A.$$

The following lemma shows that up to natural isomorphism, we can work with only $S-$functors.

Lemma 3.3.2. *(i) Any $\Gamma-$functor $F : Ass(A) \to Ass(B)$ is naturally isomorphic to an $S-$functor F'.*

(ii) Suppose $F, G : Ass(A) \to Ass(B)$ are $S-$functors, and $\mu : F \to G$ is a natural transformation. Then μ is the identity on the level of sets.

Proof. (i) Let $\mu : \Gamma_B \circ F \to \Gamma_A$, $\nu : \Gamma_A \to \Gamma_B \circ F$ be natural transformations such that $\mu \circ \nu = 1$, $\nu \circ \mu = 1$.

Let $(X,E) \in \mathrm{Ass}(A)$ be arbitrary. By definition $F(X,E) = (\Gamma_B \circ F(X,E), E')$ for some E'. Now $\mu_{(X,E)}$ is a morphism $(\Gamma_B \circ F(X,E), E') \to (X, E' \circ \nu_{(X,E)})$ realized by the identity, and similarly, $\nu_{(X,E)}$ is a morphism $(X, E' \circ \nu_{X,E}) \to (\Gamma_B \circ F(X,E), E')$. So we have an isomorphism $(X, E' \circ \nu_{(X,E)}) \cong F(X,E)$.

Now define $F'(X,E) = (X, E' \circ \nu_{(X,E)})$ for all $(X,E) \in \mathrm{Ass}(A)$ (where $F(X,E) = (\Gamma_B \circ F(X,E), E')$), and for $f : (X,E) \to (Y,F)$ a morphism, define $F'(f) = f$.

Since $f = \mu_{(Y,F)} \circ F(f) \circ \nu_{(X,E)}$, it is realized as a morphism $F'(X,E) \to F'(Y,F)$. Therefore, F' is a functor that satisfies the statement.

(ii) Recall that $1 = (\{*\}, E)$, with $E(*)$ any nonempty set of A is the terminal object of $\mathrm{Ass}(A)$. Let $(X,E) \in \mathrm{Ass}(A)$ with $x_0 \in X$ be arbitrary. I'll prove that $\mu_{(X,E)}(x_0) = x_0$.

Define the arrow $\hat{x_0} : 1 \to (X,E)$ by $\hat{x_0}(*) = x_0$. It is easily seen that this is a morphism in $\mathrm{Ass}(A)$.

Since F, G are $S-$functors, they preserve the terminal object, i.e. $F(1) = (\{*\}, T) = 1$ and $G(1) = (\{*\}, T') = 1$. Hence $\mu_1 : \{*\} \to \{*\}$ is the identity function. Also, $F(\hat{x_0}) = \Gamma_B \circ F(\hat{x_0}) = \Gamma_A(\hat{x_0}) = \hat{x_0}$.

By naturality, we now have

$$\mu_{(X,E)}(x_0) = \mu_{(X,E)} \circ \hat{x_0}(*) = F(\hat{x_0})(*) = x_0.$$

Since $x_0 \in X$ was arbitrary, this means that μ is the identity on the level of sets.

□

The following lemma corresponds to lemma 1.6.3 in [vO08], the proof can be found there.

Lemma 3.3.3. *Every $S-$functor $F : Ass(A) \to Ass(B)$ is a ∇-functor.*

Working with $S-$functors makes the proofs of the following propositions a little easier. The first proposition is a new result that generalizes the next, which is due to Longley in [Lon95]. For $\Gamma : A \to B$ a proto-applicative morphism (see definition 2.5.19), define an $S-$functor $\Gamma^* : \mathrm{Ass}(A) \to \mathrm{Ass}(B)$ by:

$$\Gamma^*(X,E) = (X, \Gamma \circ E).$$

For an $S-$functor $F : A \to B$, we define a proto-applicative morphism $\widetilde{F}$ as the unique $\widetilde{F} : \mathcal{P}(A) \to \mathcal{P}(B)$ that satisfies:

$$F(\mathcal{P}(A), 1_{\mathcal{P}(A)}) = (\mathcal{P}(A), \widetilde{F}).$$

Of course we have to check that Γ^* is a functor and $\widetilde{F}$ is an applicative morphism. We now have the following proposition.

Proposition 3.3.4. *Let A, B be partial combinatory algebras.*

(i) Every proto-applicative morphism $\Gamma : A \to B$ gives rise to a left-exact $S-$functor $\Gamma^* : Ass(A) \to Ass(B)$. Moreover, for Δ another proto-applicative morphism, $\Gamma \preceq \Delta$ implies that there is a unique natural transformation $\Gamma^* \Rightarrow \Delta^*$.

(ii) Every left-exact $S-$functor $F : Ass(A) \to Ass(B)$ defines an applicative morphism $\widetilde{F} : A \to B$. If $G : Ass(A) \to Ass(B)$ is another left-exact S-functor and there is a natural transformation $F \Rightarrow G$, then $\widetilde{F} \preceq \widetilde{G}$.

(iii) For every Γ, F as above, $\widetilde{(\Gamma^*)} = \Gamma$ and $(\widetilde{F})^* \simeq F$. Therefore, there is a biequivalence between the 2-categories PCA_{proto} and the subcategory of Cat with categories of assemblies as objects and left-exact $S-$functors as morphisms.

Proof. (i) We will first check that Γ^* is a functor. Suppose Γ is realized by $r \in B$. Let $(X,E),(Y,E') \in \mathrm{Ass}(A)$. Suppose $f:(X,E)\to(Y,E')$ is tracked by $t \in A$. Then

$$r\Gamma(\{t\})(\Gamma \circ E(x)) \subseteq \Gamma(tE(x)) \subseteq \Gamma(E'(f(x)))$$

So $f:(X,E)\to(Y,E')$ is tracked by some element of $r\Gamma\{t\}$. Hence Γ^* is an $S-$functor. It remains to show that Γ^* preserves the terminal object, binary products and equalizers.

It is easy to see that Γ^* preserves the terminal object. For $(X,E),(Y,E') \in \mathrm{Ass}(A)$, their product is given by $(X\times Y, E\wedge E')$ where

$$E\wedge E'(x,y) = \mathsf{p}E(x)E(y)$$

We claim that the identity map yields an isomorphism

$$(X\times Y, \Gamma\circ(E\wedge E')) \cong (X\times Y, \Gamma\circ E \wedge \Gamma\circ Y).$$

Since the RHS is a binary product (and we have projection maps),

$$1_{X\times Y} : (X\times Y, \Gamma\circ(E\wedge E')) \to (X\times Y, \Gamma\circ E\wedge\Gamma\circ Y)$$

is a morphism in $\mathrm{Ass}(B)$. For the converse, observe that for any $z \in \Gamma(\{\mathsf{p}\})$:

$$r(rz\Gamma(E(x)))\Gamma(E(y)) \subseteq \Gamma(\mathsf{p}E(x)E(y))$$

so $\langle x\rangle r(rz(\mathsf{p}_0 x))(\mathsf{p}_1 x) \in B$ tracks $1_{X\times Y}$ in the other direction (here $\mathsf{p}_0,\mathsf{p}_1$ are the projections for the pairing combinator in B). So we have an isomorphism.

Any regular monomorphism in $\mathrm{Ass}(A)$ is a strong subobject and these are all of the form

$$m:(X', E\restriction X') \rightarrowtail (X,E)$$

where $X'\subseteq X$ (see proposition 4.1.5). It is now easy to see that the corresponding equalizer diagram is preserved (e.g. use the strong subobject classifier in $\mathrm{Ass}(A)$, see the proof of proposition 4.1.8). Therefore Γ^* preserves finite limits.

Now suppose $\Gamma \le \Delta$ is realized by $t \in B$. Then for $(X,E)\in\mathrm{Ass}(A)$, the identity function on X yields a morphism

$$(X,\Gamma\circ E)\to(X,\Delta\circ E)$$

tracked by t. This gives a natural transformation $\Gamma^* \Rightarrow \Delta^*$.

(ii) We first show that $\widetilde{F}$ is a proto-applicative morphism. Let (P,E) be a subobject of $(\mathcal{P}(A),1_{\mathcal{P}(A)})\times(\mathcal{P}(A),1_{\mathcal{P}(A)})$ defined by:

$$P = \{(\alpha,\alpha')\in\mathcal{P}(A)\times\mathcal{P}(A)\mid \alpha\alpha'\downarrow\}$$

with

$$E(\alpha,\alpha') = \mathsf{p}\alpha\alpha' = 1_{\mathcal{P}(A)}\wedge 1_{\mathcal{P}(A)}(\alpha,\alpha')$$

where p is the pairing combinator in A. Then

$$(P,E)\rightarrowtail(\mathcal{P}(A),1_{\mathcal{P}(A)})\times(\mathcal{P}(A),1_{\mathcal{P}(A)})$$

is a regular mono, so since F is left-exact and by the characterisation of regular monos in $\mathrm{Ass}(B)$, we have that

$$F(P,E)\cong(P,\widetilde{F}\times\widetilde{F}\restriction P).$$

Now the application map $\mathrm{app} : (P,E) \to (\mathcal{P}(A), 1_{\mathcal{P}(A)})$ sending (α,α') to $\alpha\alpha'$ yields a morphism

$$F(\mathrm{app}) : F(P,E) \cong (P, \widetilde{F} \times \widetilde{F} \restriction P) \to (\mathcal{P}(A), \widetilde{F})$$

realized by some r in B. This r then realizes $\widetilde{F}$ as proto-applicative morphism $A \to B$.

For a natural transformation $\alpha : \mathcal{F} \Rightarrow \mathcal{G}$, α is the identity on the level of sets (lemma 3.3.2(ii)). Therefore, the identity map on $\mathcal{P}(A)$ gives a morphism

$$(\mathcal{P}(A), \widetilde{F}) \to (\mathcal{P}(A), \widetilde{G})$$

hence $\widetilde{F} \leq \widetilde{G}$.

(iii) It is easy to see that $\widetilde{(\Gamma^*)} = \Gamma$. Also $(\Gamma \circ \Delta)^* = \Gamma^* \circ \Delta^*$ for any $\Delta : C \to A$, and $(1_{\mathcal{P}(A)})^* = 1_{\mathrm{Ass}(A)}$. So if we define $\mathrm{Ass}(\Gamma) = \Gamma^*$, then we have a 2-functor

$$\mathrm{Ass}(-) : \mathrm{PCA}_{\mathrm{proto}} \to \mathrm{Cat}.$$

We still have to show that for any $F : \mathrm{Ass}(A) \to \mathrm{Ass}(B)$:

$$(\widetilde{F})^* \cong F.$$

Let $\eta : 1 \Rightarrow \nabla_A \Gamma_A$ be the unit of the adjunction $\Gamma_A \dashv \nabla_A$. Then by uniqueness of the natural transformation of $S-$functors $F \Rightarrow F\nabla_A\Gamma_A$ (they are the identity on the level of sets,), we have that $F(\eta_{(X,E)}) = \eta_{F(X,E)}$ for all $(X,E) \in \mathrm{Ass}(A)$.

Now for (X,E) an assembly, E is a morphism $(X,E) \to (\mathcal{P}(A), 1_{\mathcal{P}(A)})$. Moreover, we have a pullback diagram:

$$\begin{array}{ccc} (X,E) & \xrightarrow{E} & (\mathcal{P}(A), 1_{\mathcal{P}(A)}) \\ \downarrow{\scriptstyle \eta} & & \downarrow{\scriptstyle \eta} \\ \nabla_A(X) & \xrightarrow{\nabla E} & \nabla_A(\mathcal{P}(A)) \end{array}$$

The above observations and the fact that F is left-exact yield a pullback diagram:

$$\begin{array}{ccc} F(X,E) & \xrightarrow{F(E)=E} & F(\mathcal{P}(A), 1_{\mathcal{P}(A)}) \\ \downarrow{\scriptstyle \eta} & & \downarrow{\scriptstyle \eta} \\ \nabla_B(X) & \xrightarrow{\nabla E} & \nabla_B(\mathcal{P}(A)) \end{array} .$$

But we have that $F(\mathcal{P}(A), 1_{\mathcal{P}(A)}) = (\mathcal{P}(A), \widetilde{F})$. It is also easy to see that the following is pullback:

$$\begin{array}{ccc} (X, \widetilde{F} \circ E) & \xrightarrow{E} & (\mathcal{P}(A), \widetilde{F}) \\ \downarrow{\scriptstyle \eta} & & \downarrow{\scriptstyle \eta} \\ \nabla_B(X) & \xrightarrow{\nabla E} & \nabla_B(\mathcal{P}(A)) \end{array} .$$

Therefore $(X, \widetilde{F} \circ E) \cong F(X,E)$ and we have obtained the equivalence $F \cong (\widetilde{F})^*$.

We have thus obtained a biequivalence $\mathrm{Ass}((-))$ between the categories $\mathrm{PCA}_{\mathrm{proto}}$ and the subcategory of Cat consisting of categories of assemblies and left-exact $S-$functors.

□

The following corollary is immediate from lemma 3.3.2

Corollary 3.3.5. *For all pcas A, B, we have a natural equivalence of categories:*

$$PCA_{proto}(A,B) \cong \Gamma\nabla Cat(Ass(A), Ass(B)).$$

For an applicative morphism $\gamma : A \to B$, the proto-applicative morphism $\mathcal{P}(\gamma)$ gives rise to a left-exact functor $\mathcal{P}(\gamma)^* : \mathrm{Ass}(A) \to \mathrm{Ass}(B)$. John Longley had already shown in [Lon95] that the functors corresponding to applicative morphisms in this way are precisely the regular Γ-functors $\mathrm{Ass}(A) \to \mathrm{Ass}(B)$. We therefore have the below result. A proof using $S-$functors can be found in [vO08].

Proposition 3.3.6 (Longley)**.** *Let A, B be partial combinatory algebras.*

(i) Every applicative morphism $\gamma : A \to B$ gives rise to a regular $\Gamma-$functor $\mathcal{P}(\gamma)^* : Ass(A) \to Ass(B)$. For $\delta : A \to B$ another applicative morphism such that $\gamma \preceq \delta$, there is a (necessarily) unique natural transformation $\mathcal{P}(\gamma)^* \Rightarrow \mathcal{P}(\delta)^*$.

(ii) Every regular $\Gamma-$functor $F : Ass(A) \to Ass(B)$ defines an applicative morphism $\widetilde{F} : A \to B$. If there is a natural transformation $F \Rightarrow G$, then $\widetilde{F} \preceq \widetilde{G}$.

(iii) We have that $\mathcal{P}(\widetilde{F})^* \cong F$ and $\widetilde{\mathcal{P}(\gamma)} \simeq \gamma$. Therefore there is a natural equivalence of categories

$$PCA(A,B) \cong \Gamma\nabla Reg(Ass(A), Ass(B)).$$

Corollary 3.3.7. *The map $\gamma \mapsto \mathcal{P}(\gamma)$ restricts to a 1-1 correspondence between computationally dense applicative morphisms and proto-applicative morphisms that have a right adjoint.*

Proof. Suppose $\Gamma \dashv \Delta : B \to A$ is an adjoint pair of proto-applicative morphisms. Then $\Gamma^* \dashv \Delta^* : \mathrm{Ass}(B) \to \mathrm{Ass}(A)$ is an adjoint pair of left-exact functors. So $\Gamma^* : A \to B$ preserves finite limits, but it preserves colimits since it is a left adjoint. Therefore Γ^* is regular, whence we see from proposition 3.3.6 that $\Gamma \cong \mathcal{P}(\gamma)$ for some applicative morphism $\gamma : A \to B$. This γ is computationally dense by theorem 2.5.20.

The other directions follows directly from theorem 2.5.20. □

The following is a nice property of regular functors between categories of assemblies to keep in mind. Again, for a proof I refer to [Lon95], propositions 2.3.2/2.3.3, or [vO08]. Note that one part is implied by lemma 3.3.3.

Proposition 3.3.8. *For every regular functor $F : Ass(A) \to Ass(B)$, F is a ∇-functor or Γ-functor if and only if F is a $\Gamma\nabla$-functor.*

Chapter 4

Realizability toposes

One of the greatest papers in the field of realizability is [Hyl82], in which the effective topos is constructed. The Effective Topos is the archetype of a *realizability topos*. Its internal logic is the same as Kleene's original notion of *realizability*.

In this chapter, we will describe the construction of a realizability topos for arbitrary partial combinatory algebras. It will arise as a universal completion of the category of assemblies on a pca. The fact that this gives a topos is not immediate! After that, we will look at subobjects and the constant objects functor, the latter will turn out to be a geometric inclusion of the category of sets.

4.1 How Much of a Topos is Ass(A)?

In this section we will show to what extent the category Ass(A) is already a topos. We start by defining what a topos is.

Definition 4.1.1. In a category $\mathcal{C}$ with finite limits, a *subobject classifier* is an arrow $t : 1 \to \Omega$ such that for every subobject $U \rightarrowtail X$, there exists a unique map $\chi : X \to \Omega$ such that the following is a pullback square:

$$\begin{array}{ccc} U & \longrightarrow & 1 \\ \downarrowtail & & \downarrow t \\ X & \xrightarrow{\chi} & \Omega \end{array}.$$

Here 1 is the terminal object and Ω is some object, sometimes called an object of *truth values*. We say that the map χ *characterises* $U \rightarrowtail X$.

We can now give the definition of a *topos*:

Definition 4.1.2. A category $\mathcal{E}$ is called a *topos* if it is finitely complete, cartesian closed and has a subobject classifier $t : 1 \to \Omega$.

One of the motivations of studying toposes is that "essentially any categorical construction that can be done in Set can be done in an arbitrary topos". If you want to learn more about what this exactly means, [MLM94] is an excellent introduction. Otherwise, one could also step into the monumental [Joh02a] which covers everything in great detail. For now I will wrap up a couple of results in a proposition.

Proposition 4.1.3. *Let $\mathcal{E}$ be a topos. Then $\mathcal{E}$ is regular, balanced and finitely cocomplete. Moreover, for any object $X \in \mathcal{E}$, the lattice of subobjects Sub(X) has the structure of a Heyting algebra, and for*

$f : Y \to X$ an arrow, pullback of subobjects: $f^* : Sub(X) \to Sub(Y)$ is a functor that preserves the Heyting structure and has both a left- and right adjoint.

From proposition 3.1.4, we see that $\mathrm{Ass}(A)$ has a lot of the properties of a topos. However, it is not a topos. For instance, it is not balanced: If we look at the map $(\mathbb{N}, N) \to \nabla(\mathbb{N})$ given by the identity, then this both epi and mono (see propositions 3.1.5, 3.1.6), but clearly not an isomorphism.

It turns out that it does have a little more topos-like structure than proposition 3.1.4 tells us. We start by defining *strong subobjects*.

Definition 4.1.4. In a category $\mathcal{C}$, a subobject $m : U \rightarrowtail X$ is called *strong* if for every commutative square

$$\begin{array}{ccc} V & \twoheadrightarrow & Y \\ \downarrow & \overset{\exists}{\swarrow} & \downarrow \\ U & \underset{m}{\rightarrowtail} & X \end{array}$$

where $V \twoheadrightarrow Y$ is an epimorphism, there exists an arrow $Y \to U$ such that the diagram commutes.

The following characterises the strong subobjects of $\mathrm{Ass}(A)$:

Proposition 4.1.5. *A subobject $(U,F) \rightarrowtail (X,E)$ in $Ass(A)$ is* strong *if and only if it is isomorphic to a subobject of the form $(X', E \restriction X')$ where $X' \subseteq X$, with the inclusion map.*

Proof. "⇒". Let $X' = m(U)$. Since (U,E) is a strong subobject, there exists $h : (X', E \restriction X') \to (U,F)$ such that the following diagram commutes:

$$\begin{array}{ccc} (U,F) & \overset{e}{\twoheadrightarrow} & (X', E \restriction X') \\ \downarrow{\scriptstyle 1_{(U,F)}} & \swarrow_{h} & \downarrow{\scriptstyle i} \\ (U,F) & \rightarrowtail & (X,E) \end{array}$$

Therefore $h \circ e = 1_{(U,F)}$ and $e \circ h \circ e = e$, hence $e \circ h = 1_{(X', E\restriction X')}$, so e is an isomorphism.

"⇐": For any commutative square as in definition 4.1.4 for the subobject $(X', E \restriction X') \to (X,E)$, it is easy to see that the morphism $V \to (X,E)$ is in fact also a morphism $V \to (X', E \restriction X')$. □

A weaker notion than that of a subobject classifier is the notion of a *strong subobject classifier*.

Definition 4.1.6. In a category $\mathcal{C}$ finite limits, an arrow $t : 1 \to \Omega$ is a strong subobject classifier if we have for every strong subobject $U \rightarrowtail X$ a unique map $\chi : X \to \Omega$ such that the following is a pullback square:

$$\begin{array}{ccc} U & \to & 1 \\ \downarrow & & \downarrow{\scriptstyle t} \\ X & \underset{\chi}{\to} & \Omega \end{array} .$$

We say that the map χ *characterises* $U \rightarrowtail X$.

Definition 4.1.7. A *quasi-topos* is a category with finite limits, and finite colimits that is locally cartesian closed and admits a strong subobject classifier.

Proposition 4.1.8. *For any pca A, $Ass(A)$ is a quasi-topos.*

Proof. We have to prove that $\mathrm{Ass}(A)$ admits a strong subobject classifier. Define $\Omega = \nabla\{0,1\}$ and define $t : 1 \to \nabla\{0,1\}$ by $t(*) = 0$.

Let $(X,E) \in Ass(A)$ with $X' \subseteq X$ be arbitrary. Then it is easy to see that $(X', E \restriction X') \rightarrowtail (X,E)$ is characterized by $\chi : (X,E) \to \Omega$ defined as:

$$\chi(x) = \begin{cases} 0 & \text{if } x \in X' \\ 1 & \text{if } x \notin X'. \end{cases}$$

By proposition 4.1.5, we are done. □

There are many other examples of quasi-toposes. A brief motivation for the definition of a quasi-topos can be found in [Joh02a], and [Men00] provides a lot of examples. In section 4.3, we see how a category of assemblies $\mathrm{Ass}(A)$ can be "completed" to a topos: the *realizability topos* $\mathrm{RT}(A)$.

4.2 Relations in a Regular Category

Preliminary to our construction of a realizability topos, we need to discuss relations in regular categories. Relations can be used to generalize a lot of constructions from set theory, for example where one takes the quotient of a set with respect to some equivalence relation.

For $\mathcal{E}$ a category with finite limits, we call a subobject of a product $X \times Y$ a *relation* from X to Y. If $\mathcal{E}$ is regular, we can define the *composition* $RS \rightarrowtail X \times Z$ of two relations $R \overset{(\partial_X,\partial_Y)}{\rightarrowtail} X \times Y$, $S \overset{(\epsilon_Y,\epsilon_Z)}{\rightarrowtail} X \times Z$ by the image of $R * S \overset{(\partial_X \circ \pi_R, \epsilon_Z \circ \pi_S)}{\to} X \times Z$ in the following diagram:

$$\begin{array}{ccccc} R*S & \xrightarrow{\pi_S} & S & \xrightarrow{\epsilon_Z} & Z \\ \downarrow{\scriptstyle \pi_R} & & \downarrow{\scriptstyle \epsilon_Y} & & \\ R & \xrightarrow{\partial_Y} & Y & & \\ \downarrow{\scriptstyle \partial_X} & & & & \\ X & & & & \end{array}$$

where the square is pullback. One can check that composition of relations is associative.

Definition 4.2.1. Let $\mathcal{E}$ be a category with finite limits, and $X \in \mathcal{E}$ an object. A subobject $R \overset{(\partial_0,\partial_1)}{\rightarrowtail} X \times X$ is called an *equivalence relation*, if the following axioms hold:

1. (reflexivity) The diagonal $\Delta : X \to X \times X$ factors through (∂_0, ∂_1)
2. (symmetry) The map (∂_1, ∂_0) factors through (∂_0, ∂_1).
3. (transitivity) If the following diagram is pullback:

$$\begin{array}{ccc} R*R & \xrightarrow{\pi_2} & R \\ \downarrow{\scriptstyle \pi_1} & & \downarrow{\scriptstyle \partial_0} \\ R & \xrightarrow{\partial_1} & X \end{array}$$

then $(\partial_0\pi_1, \partial_1\pi_2)$ factors through (∂_0, ∂_1).

I adopted the notation from [MLM94]. Observe that the transitivity condition just means that the composition of R with itself is R.

Every equivalence relation $R \overset{(\partial_0,\partial_1)}{\rightarrowtail} X \times X$ on an object $X \in \mathcal{E}$ induces a relation R' on the sets $\mathcal{E}(Y,X)$ for every $Y \in \mathcal{E}$ by:

$$R' = \{(\partial_0 \circ f, \partial_1 \circ f) \mid f : Y \to R\}.$$

One can check that this induced relation is an equivalence relation in the usual sense, i.e. in Set. Conversely, if such an induced relation for a subobject $(\partial_0,\partial_1) : R \rightarrowtail X \times X$ is an equivalence relation, it is an equivalence relation in the sense of definition 4.2.1.

Definition 4.2.2. An equivalence relation $R \overset{(\partial_0,\partial_1)}{\rightarrowtail} X \times X$ is called *effective* if it arises as a kernel pair of some arrow $f : X \to Y$.

It is easy to see that any kernel pair is an equivalence relation. We can now define what an *exact* category is:

Definition 4.2.3. A category $\mathcal{E}$ is called *exact* if it is regular and if every equivalence relation is effective.

If $\mathcal{E}$ is regular, we can take the *quotient* of an equivalence relation as their coequalizer. It is the categorical analogue of the "equivalence classes modulo R". Since in a regular category, every coequalizer is the coequalizer of its kernel pair, each equivalence relation factors through an effective equivalence relation, and their quotients are the same.

If a regular category $\mathcal{E}$ is not exact, this means that there are equivalence relations for which either the quotient does not exist, or the quotient is the quotient of a bigger effective equivalence relation. In other words, in taking the quotient, we have to divide out a little bit too much (or we cannot do it at all).

For a partial combinatory algebra A, the regular category $\mathrm{Ass}(A)$ is not in general exact. Therefore, not all equivalence relations have "exact" quotients. Note that they always have quotients, since $\mathrm{Ass}(A)$ has all finite colomits. In the following, we present a construction, called the *exact/regular completion* of a regular category, which extends a regular category by adding "exact" quotients. In the case of $\mathrm{Ass}(A)$, this completion not only yields an exact category, but even a topos. For more on completions, an accessible treatment is given in [Men00].

4.2.1 Functional Relations

Suppose we have $X, Y \in \mathcal{E}$, with equivalence relations $R \rightarrowtail X \times X$, $S \rightarrowtail Y \times Y$. We can distinguish a specific relation $F \rightarrowtail X \times Y$ that we call a *functional relation*. Intuitively, one can view it as a function from the equivalence classes of X modulo R to the equivalence classes of Y modulo S. For a relation $F \overset{(\partial_X,\partial_Y)}{\rightarrowtail} X \times Y$, we define the relation $F^{\mathrm{tw}} \rightarrowtail Y \times X$ by the composition of (∂_X,∂_Y) with the twist map $\mathrm{tw} : X \times Y \to Y \times X$.

Definition 4.2.4. For $R \rightarrowtail X \times X$, $S \rightarrowtail Y \times Y$ equivalence relations, a relation $F \rightarrowtail X \times Y$ is a functional relation from R to S if the following conditions hold:

(i) (relational) $RF \leq F$ and $FS \leq F$

(ii) (single-valued) $F^{\mathrm{tw}}F \leq S$

(iii) (total) $R \leq FF^{\mathrm{tw}}$.

One can show that the composition of two functional relations is again a functional relation. Since composition is also associative, this establishes the resemblance between functional relations and functions.

4.3 The Exact/Regular Completion of Ass(A)

For a regular category $\mathcal{E}$, the *exact/regular* completion $\mathcal{E}_{\text{ex/reg}}$ is the category that has the equivalence relations of E as objects, and functional relations between them as morphisms. We define $y : \mathcal{E} \to \mathcal{E}_{\text{ex/reg}}$ by $y(X) = \Delta_X$, where Δ_X is the diagonal $\Delta_X \rightarrowtail X \times X$, and morphisms $f : X \to Y$ are mapped to the functional relation $y(f)$ between $y(X)$ and $y(Y)$ given by $(1_X, f)$.

We have the following theorem:

Theorem 4.3.1. *$\mathcal{E}_{ex/reg}$ is exact, the functor $y : \mathcal{E} \to \mathcal{E}_{ex/reg}$ is regular, full and faithful, and surjective on subobjects, and every object of $\mathcal{E}_{ex/reg}$ is a quotient of some object in the image of y.*

The following theorem characterizes the exact/regular completion of $\mathcal{E}$.

Theorem 4.3.2. *Let $\Phi : \mathcal{E} \to \mathcal{D}$ be a regular functor from a regular category $\mathcal{E}$ to an exact category $\mathcal{D}$. Then the following are equivalent:*

(i) There is an equivalence of categories $\mathcal{K} : \mathcal{E}_{ex/reg} \to \mathcal{D}$ that makes the following diagram commute:

$$\begin{array}{ccc} \mathcal{E} & \xrightarrow{y} & \mathcal{E}_{ex/reg} \\ & \searrow^{\Phi} & \downarrow{\scriptstyle \mathcal{K}} \\ & & \mathcal{D} \end{array}$$

(ii) Φ is full, faithful, surjective on subobjects and every object in $\mathcal{D}$ is a quotient of an object in the image of Φ.

For a proof, see [vO08], lemma 2.4.4.

A similar result is the following:

Proposition 4.3.3. *For $\Phi : \mathcal{E} \to \mathcal{D}$ a regular functor from $\mathcal{E}$ to an exact category $\mathcal{D}$, there exists a unique (up to natural isomorphism) regular functor $\Psi : \mathcal{E}_{ex/reg} \to \mathcal{D}$ such that the following diagram commutes:*

$$\begin{array}{ccc} \mathcal{E} & \xrightarrow{\Phi} & \mathcal{D} \\ \downarrow{\scriptstyle y} & \nearrow_{\Psi} & \\ \mathcal{E}_{ex/reg} & & \end{array}$$

A proof can be found in [FS90], 2.169, but the notation they use is quite different.

We will now proceed to give an explicit description of $\text{Ass}(A)_{\text{ex/reg}}$. Let (X, D) be an assembly. We will denote an equivalence relation on X as $(R, \sim) \rightarrowtail (X, D) \times (X, D)$, where we write $[x \sim y]$ instead of $\sim (x, y)$. Up to isomorphism, $R \subseteq X \times X$ and the diagonal map $X \to X \times X$ factors through $(R, \sim)$, by reflexivity. A witness for symmetry is an element $\sigma \in A$ such that:

$$\sigma \in \bigcap_{x,y \in X} [x \sim y] \to [y \sim x]$$

(recall the notation introduced in the beginning of chapter 3). Also, we use the convention that for $(x, y) \notin R$, $[x \sim y] = \emptyset$. Similarly, transitivity is witnessed by some $\tau \in A$ such that:

$$\tau \in \bigcap_{x,y,z \in X} \big([x \sim y] \wedge [y \sim z]\big) \to [x \sim z].$$

So $(R, \sim)$ is an equivalence relation if $R \subseteq X \times X$, the diagonal map factors through $(R, \sim)$,and there are σ, τ as above.

Now suppose $(R,\sim)$ and $(S,\sim)$ are equivalence relations on assemblies (X,D), (Y,E) respectively. Then a functional relation from $(R,\sim)$ to $(S,\sim)$ is a map $F : X \times Y \to \mathcal{P}(A)$, together with witnesses $\mathsf{rl},\mathsf{sv},\mathsf{tl} \in A$ such that:

(i) $\mathsf{rl} \in \bigcap_{x,x' \in X, y,y' \in Y} \big([x \sim x'] \wedge [y \sim y'] \wedge F(x,y)\big) \to F(x',y')$

(ii) $\mathsf{sv} \in \bigcap_{x \in X, y,y' \in Y} \big(F(x,y) \wedge F(x,y')\big) \to [y \sim y']$

(iii) $\mathsf{tl} \in \bigcap_{x \in X} \big([x \sim x] \to \bigcup_{y \in Y} F(x,y)\big)$

where one should observe that (i)-(iii) correspond to (i)-(iii) in definition 4.2.4.

Since in $\mathrm{Ass}(A)_{\mathrm{ex/reg}}$ the objects are equivalence relations in $\mathrm{Ass}(A)$, and the morphisms are (equivalence classes of) functional relations, the above gives an explicit description of $\mathrm{Ass}(A)_{\mathrm{ex/reg}}$. We have the following theorem:

Theorem 4.3.4. *$Ass(A)_{ex/reg}$ is a topos. We denote this topos by $RT(A)$, the* realizability topos *on A. The inclusion $I : Ass(A) \to RT(A)$ sends assemblies (X,E) to $(X, \exists_\delta E)$ where $\delta : X \to X \times X$ is the diagonal, and the equivalence relation $\exists_\delta E$ is given by:*

$$(\exists_\delta E)(x,x) = E(x).$$

I is regular, full and preserves exponentials, finite sums and the natural numbers object.

Most proofs are given for $\mathcal{K}_1$ and are generalized in a straightforward way. See for example [Hyl82].

In the following, we denote objects of $\mathrm{RT}(A)$ by $(X,\sim)$, where X is a set and $\sim$ is an equivalence relation on some assembly with domain X. Note that since the diagonal factors through this equivalence relation, that assembly is isomorphic to (X,D) where $D(x) = [x \sim x]$. Therefore $(X,\sim)$ determines a unique equivalence relation on some unique assembly in $\mathrm{Ass}(A)$. We will treat $\sim$ as a map $X \times X \to \mathcal{P}(A)$, where the image of (x,y) is denoted $[x \sim y]$.

Sometimes it will be convenient to denote an object $(X',\sim)$ by $(X,\sim)$ where $X' \subseteq X$ and

$$X' = \{x \in X \mid [x \sim x] \neq \emptyset\}.$$

So $[x \sim x]$ need not be non-empty on X. This notation will be useful when treating subobjects, for instance we will look at subobjects of the form $(X,\sim_K) \rightarrowtail (X,\sim)$, where $\sim_K$ really defines the subobject, but denoting X as its underlying set makes it clear that we are dealing with elements of "type" X. It also improves readability since we don't have to restrict the domain all the time [1].

For morphisms $f : (X,\sim) \to (Y,\sim)$, we will treat the functional relation that represents f as a map $F : X \times Y \to \mathcal{P}(A)$.

4.4 The Lattice of Subobjects in RT(A)

We wish to develop some theory on the structure of $\mathrm{RT}(A)$. Of great importance is understanding subobjects, especially when interested in the logic of a topos. In the following, we will describe a semantics for talking about subobjects and relations in $\mathrm{RT}(A)$.

For an object $(X,\sim) \in \mathrm{RT}(A)$, we view $\sim : X \times X \to \mathcal{P}(A)$ as a 2-ary predicate on X, where $\mathcal{P}(A)$ is a set of *truth values*. In general, we view the set $\mathcal{P}(A)^X$ as a set of predicates on the set X. We endow $\mathcal{P}(A)^X$ with a pre-order as follows: for $K,L \in \mathcal{P}(A)^X$, let

$$K \leq L \iff (\exists a \in A)(\forall x \in X)\, a \in K(x) \to L(x)$$

[1] In most treatments of realizability toposes, the objects $(X,\sim)$, where $\sim$ is an equivalence relation on X' for some $X' \subseteq X$ are in fact all objects of the topos, which is not the case here. Actually, it suffices for $\sim$ to be just symmetric and transitive, so it need not be reflexive. However, one should observe that $(X,\sim)$ is just notation and the only real objects are actual equivalence relations, so we will also just consider those $(X,\sim)$ elements of $\mathrm{RT}(A)$.

where we recall that for $P, Q \subseteq A$,

$$P \to Q = \{a \in A \mid (\forall b) b \in P \Rightarrow (ab\downarrow \wedge ab \in Q)\}.$$

The poset reflection of this preorder turns out to be a complete Heyting Algebra. We can define operations $\wedge, \vee$ and elements $\top, \bot$ in $\mathcal{P}(A)^X$ by:

$$\begin{aligned} K \wedge L(x) &= \{\mathsf{p}ab \mid a \in K(x), b \in L(x)\} \\ K \vee L(x) &= \{\mathsf{p}\mathsf{T}a \mid a \in K(x)\} \cup \{\mathsf{p}\mathsf{F}a \mid a \in L(x)\} \\ \top(x) &= A \\ \bot(x) &= \emptyset. \end{aligned}$$

For $f : Y \to X$ a map between sets, we have a pullback operation:

$$f^* : \mathcal{P}(A)^X \to \mathcal{P}(A)^Y$$

defined by precomposition with f. It turns out that f^* has a left- and right adjoint with respect to the preorder structure, denoted by $\exists_f$ and $\forall_f$ respectively. For every $K \in \mathcal{P}(A)^Y$, they are given by:

$$\begin{aligned} \exists_f(K)(x) &= \{a \in A \mid (\exists y) f(y) = x \wedge a \in K(y)\} \\ \forall_f(K)(x) &= \{a \in A \mid (\forall y) f(y) = x \Rightarrow a \in K(y)\}. \end{aligned}$$

For $\pi : X \times Y \to X$ a projection, we will denote $\exists_\pi$ by $\exists y$ and $\forall_\pi$ by $\forall y$. Now for $K \in \mathcal{P}(A)^X$, we write

$$a \Vdash K(x) \iff a \in K(x) \text{ for every } x.$$

and

$$\Vdash K(x) \text{ if } a \Vdash K(x) \text{ for some } a.$$

Now one can verify that $\sim : X \times X \to \mathcal{P}(A)$ expresses an equivalence relation if and only if

$$\begin{aligned} &\Vdash \forall x[x \sim x] \\ &\Vdash \forall x \forall y[x \sim y] \to [y \sim x] \\ &\Vdash \forall x \forall y \forall z \left([x \sim y] \wedge [y \sim z]\right) \to [x \sim z] \end{aligned}$$

where $\forall x[x \sim x]$ should be interpreted as $(\forall x)\delta^* \sim (x)$, for $\delta : X \to X \times X$ the diagonal. Using similar formulas, we can express that $F : X \times Y \to \mathcal{P}(A)$ is a functional relation. For $f : (X, \sim) \to (Y, \sim)$ and $g : (Y, \sim) \to (Z, \sim)$ morphisms represented by F and G, the composition $g \circ f$ is given by the functional relation FG, which satisfies:

$$\Vdash \forall x \forall z FG(x, z) \leftrightarrow \left(\exists y F(x, y) \wedge G(y, z)\right).$$

This yields a convenient way to represent compositions of morphisms.

Definition 4.4.1. A map $K : X \to \mathcal{P}(A)$ is called a *strict relation* for an object $(X, \sim) \in \mathrm{RT}(A)$ if:

$$\begin{aligned} \Vdash \forall x(K(x) \to [x \sim x]) \\ \Vdash \forall x \forall y(K(x) \wedge [x \sim y]) \to K(y) \end{aligned}$$

The first formula expresses that K is *strict*, the second that K is *relational*.

If $K : X \to \mathcal{P}(A)$ is a strict relation, let $X' = \{x \in X \mid K(x) \neq 0\}$. Define $\sim_K : X' \times X' \to \mathcal{P}(A)$ by the predicate

$$[x \sim_K x'] \leftrightarrow K(x) \wedge x \sim x'.$$

Then the following proposition is easy:

Proposition 4.4.2. *$(X', \sim_K)$ is a well defined object of RT(A). Moreover, $\sim_K$ represents a monomorphism $m : (X', \sim_K) \rightarrowtail (X, \sim)$.*

Using the following proposition, it is readily verified that for $K \in \mathcal{P}(A)^X$ a strict relation, $\sim_K : (X', \sim_K) \to (X, \sim)$ is a monomorphism:

Proposition 4.4.3. *A morphism $f : (X, \sim) \to (Y, \sim)$ in RT(A) represented by $F : X \times Y \to \mathcal{P}(A)$ is a monomorphism if and only if*

$$\Vdash \forall x \forall x' \forall y \big(F(x, y) \wedge F(x', y)\big) \to [x \sim x'].$$

The proof is a nice exercise.

Following the discussion above, we will denote $(X', \sim_K)$ by $(X, \sim_K)$.

We call monomorphisms given by

$$\sim_K : (X, \sim_K) \to (X, \sim)$$

for some strict relation K *standard monomorphisms* . The following propositions tells us that these are all:

Proposition 4.4.4. *Every monomorphism $f : (X, \sim) \to (Y, \sim)$ is isomorphic to a standard monomorphism.*

Sketch of proof. It comes down to the fact that if f is represented by $F : X \times Y \to \mathcal{P}(A)$, then

$$(X, \sim) \cong (Y', \sim_F)$$

where $Y' = \{y \in Y \mid (\exists x) F(x, y) \neq \emptyset\}$ and

$$y \sim_F y' \leftrightarrow y \sim y' \wedge (\exists x F(x, y)).$$

It is then clear that $\sim_F$ represents a standard mono $(Y', \sim_F) \rightarrowtail (Y, \sim)$. □

4.4.1 The Subobject Classifier of RT(A)

By the above, the lattice of subobjects of an object $(X, \sim)$ in RT(A) is isomorphic to the lattice of (equivalence classes of) strict relations in $\mathcal{P}(A)^X$.

Define $(\Omega, \Leftrightarrow) \in \mathrm{RT}(A)$ by $\Omega = \mathcal{P}(A)$, and:

$$A \Leftrightarrow B := (A \to B) \wedge (B \to A).$$

We can define a map $\top : 1 \to \Omega$ by the functional relation $1_{\mathcal{P}(A)} \in \mathcal{P}(A)^{1 \times \mathcal{P}(A)} \cong \mathcal{P}(A)^{\mathcal{P}(A)}$. This map is clearly a monomorphism.

For every strict relation K on an object $(X, \sim)$, we have a map $\chi_K : (X, \sim) \to (\Omega, \Leftrightarrow)$ represented by the functional relation

$$[x \sim x] \wedge K(x) \Leftrightarrow A.$$

Proposition 4.4.5. *For every strict relation K, χ_K is the unique morphism such that we have a pullback square*

$$\begin{array}{ccc} (X,\sim_K) & \longrightarrow & 1 \\ \downarrow & & \downarrow \top \\ (X,\sim) & \xrightarrow{\chi_K} & (\Omega,\Leftrightarrow) \end{array} .$$

Therefore $\top : 1 \to (\Omega,\Leftrightarrow)$ is a subobject classifier in RT(A).

The equivalence classes of strict relations on an object $(X,\sim)$ form a Heyting algebra. However, one should note that one cannot just take the Heyting structure from $\mathcal{P}(A)^X$, since the strict relations need not be closed under those binary operations. Without proof, I'll just give the operations for the strict relations. Whenever I refer to the Heyting structure of the sublattice of strict relations, I enlose the expression by double brackets $[\![\ldots]\!]$ (as a reference to the subobject it represents). For $K, L \in \mathcal{P}(A)^X$ strict relations, we have:

$$\begin{aligned} [\![K \wedge L(x)]\!] &= K(x) \wedge L(x) \\ [\![K \vee L(x)]\!] &= K(x) \vee L(x) \\ [\![(K \to L)(x)]\!] &= [x \sim x] \wedge (K(x) \to L(x)) \end{aligned}$$

So in fact, only in the case of the Heyting implication we had to do something more to guarantee that the resulting relation is strict. We have top and bottom elements $\top, \bot$, given by $\top(x) = A$, $\bot(x) = \emptyset$ for all $x \in X$. Negation is, as always, defined by:

$$[\![\neg K(x)]\!] = [\![(K \to \bot)(x)]\!] = x \sim x \wedge (K(x) \to \emptyset)$$

4.4.2 Some More Objects in RT(A)

We will give a quick description of some more objects in RT(A).

Powerobject There is a convenient representation of the power object of an object $(X,\sim)$ in terms of strict relations on X. Define the following relation on $\mathcal{P}(A)^X$:

$$K \sim L := \forall x \forall y \underbrace{(K(x) \to [x \sim x]) \wedge \big((K(x) \wedge [x \sim y]) \to K(y)\big)}_{K \text{ is } \textit{strict} \text{ and } \textit{relational}} \wedge [x \sim x] \to (K(x) \leftrightarrow L(x)) .$$

Then one can verify that $(\mathcal{P}(A)^X, \sim)$ is a powerobject.

Pullback of Subobjects Suppose $f : (Y,\sim) \to (X,\sim)$ is a morphism represented by $F \in \mathcal{P}(A)^{Y \times X}$. This morphism gives rise to a functor $f^* : \mathrm{Sub}(X) \to \mathrm{Sub}(Y)$ by pullback; for $(X,\sim_K) \rightarrowtail (X,\sim)$ we have the pullback squares:

$$\begin{array}{ccccc} f^*(X,\sim_K) & \longrightarrow & (X,\sim_K) & \longrightarrow & 1 \\ \downarrow & & \downarrow & & \downarrow \\ (Y,\sim) & \xrightarrow{f} & (X,\sim) & \xrightarrow{\chi_K} & (\Omega,\Leftrightarrow) \end{array}$$

The characteristic map of $f^*(X,\sim_K)$ is therefore represented by

$$(\exists x) F(y,x) \wedge [x \sim x] \wedge (K(x) \Leftrightarrow A).$$

this can be easily seen to be equivalent to

$$[y \sim y] \wedge (\exists x F(y,x) \wedge K(x) \Leftrightarrow A)$$

so $f^*(X,\sim_K)$ is given by the strict relation $[\![K(f(y))]\!] = \exists x F(y,x) \wedge K(x)$, which is the pullback of K along F.

This pullback operation has left and right adjoints, denoted $\exists_f, \forall_f$ defined on the strict relations on $(Y,\sim)$ by:

$$[\![(\exists_f L)(x)]\!] := (\exists y) L(y) \wedge F(y,x)$$
$$[\![(\forall_f L)(x)]\!] := [x \sim x] \wedge \forall y \big(F(y,x) \to L(y)\big)$$

Epimorphisms Similar to monomorphisms, a morphism $f : (X,\sim) \to (Y,\sim)$ represented by F is an epimorphism if and only if

$$\Vdash \forall y \big([y \sim y] \to (\exists x F(x,y))\big).$$

4.5 The Inclusion Set → RT(A)

Recall the constant objects functor $\nabla : \mathrm{Set} \to \mathrm{Ass}(A)$. By composing this with the inclusion $I : \mathrm{Ass}(A)$, we can view the constant objects functor as a functor $\nabla : \mathrm{Set} \to \mathrm{RT}(A)$ given by:

$$\nabla(X) = (X, \exists_\delta(\top))$$

where $\top$ is the top element in $\mathcal{P}(A)^X$ (the constant map with value A) and $\delta : X \to X \times X$ is the diagonal embedding. Concretely, $\exists_\delta(\top)(x,x') = A$ if $x = x'$ and $\emptyset$ otherwise. Since $I : \mathrm{Ass}(A) \to \mathrm{RT}(A)$ is regular, we find that $\mathrm{RT}(A)$ is a $\nabla-$category in a canonical way (definition 3.2.1).

In the same way, we have a functor $\Gamma : \mathrm{RT}(A) \to \mathrm{Set}$ given by $\Gamma(X,\sim) = \mathrm{RT}(A)(1,(X,\sim))$ or:

$$\Gamma(X,\sim) = \{x \in X \mid [x \sim x] \neq \emptyset\}.$$

In the same way as before, we have an adjunction $\Gamma \dashv \nabla$, and Γ preserves finite limits. So $\mathrm{RT}(A)$ is a $\Gamma\nabla$-category. Since ∇ is full and faithful, $\nabla : \mathrm{Set} \to \mathrm{RT}(A)$ is an embedding of toposes. In the rest of the section, I'll show that this coincides with the subtopos of $\neg\neg-$sheaves in $\mathrm{RT}(A)$.

Recall the following definitions:

Definition 4.5.1. Let $\mathcal{E}$ be a topos. An operator $\overline{(\cdot)} : \mathrm{Sub}(X) \to \mathrm{Sub}(X)$, defined for every object $X \in \mathcal{E}$ is a *closure operator* if it commutes with pullback and satisfies for all $X, A \in \mathrm{Sub}(X)$:

(i) $A \leq \overline{A}$

(ii) $\overline{\overline{A}} \cong \overline{A}$

(iii) $\overline{A \wedge B} \cong \overline{A} \wedge \overline{B}$

A subobject $A \rightarrowtail X$ is called *dense* with respect to the closure operator whenever $\overline{A} \cong X$.

Definition 4.5.2. For a closure operator $\overline{(\cdot)}$ in a topos $\mathcal{E}$, an object $F \in \mathcal{E}$ is a *sheaf* with respect to $\overline{(\cdot)}$ if for all dense monomorphisms $m : A \rightarrowtail X$, precomposition with m yields a bijection

$$m^* : \mathcal{E}(X,F) \to \mathcal{E}(A,F).$$

When m^* is only a monomorphism, we say that F is *separated*.

In any topos, the subobjects of an object form a Heyting algebra, and the map $\neg\neg : A \to \neg\neg A$ is a closure operator. For $\mathrm{RT}(A)$, this is easily seen since we can just work with strict relations, and observe that for a strict relation $K : X \to \mathcal{P}(A)$ on $(X, \sim)$:

$$\neg\neg K(x) \simeq \begin{cases} [x \sim x] & \text{if } K(x) \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

Therefore, a strict relation is $\neg\neg-$dense if and only $K(x) \neq \emptyset$ whenever $[x \sim x] \neq \emptyset$. We have the following proposition:

Proposition 4.5.3. *An object $(F, \sim)$ of $RT(A)$ is a $\neg\neg-$sheaf if and only if $(F, \sim) \cong \nabla\Gamma(F, \sim)$.*

Proof. Let $(F, \sim) \in \mathrm{RT}(A)$ be a $\neg\neg-$sheaf. Consider the monomorphism $m : (F, \sim) \rightarrowtail \nabla\Gamma(F, \sim)$ given by the strict relation $K(f) := [f \sim f]$, which is $\neg\neg$-dense.

Therefore, there is an arrow $g : \nabla\Gamma(F, \sim) \to (F, \sim)$ such that $gm = 1_{(F,\sim)}$ whence we can deduce that

$$\bigcap_{f \in F} [f \sim f] \neq \emptyset.$$

It is now easy to show that $(F, \sim) \cong \nabla\Gamma(F, \sim)$.

The converse is also not hard. □

Corollary 4.5.4. *The subtopos of $\neg\neg$-sheaves in $RT(A)$ is equivalent to Set.*

Lastly, we have another way of characterizing the assemblies in $\mathrm{RT}(A)$:

Proposition 4.5.5. *The full subcategory of assemblies in $RT(A)$ is equivalent to the full subcategory of $\neg\neg$-separated objects.*

Proof. Since the inclusion of assemblies $\mathrm{Ass}(A) \to \mathrm{RT}(A)$ is surjective on subobjects, every subobject of an assembly in $\mathrm{RT}(A)$ is an assembly (this is also not hard to see directly). It is easy to see that every assembly $(X, \sim)$ is a subobject of $\nabla\Gamma(X, \sim)$. Since the latter is a sheaf, so especially separated, every assembly is separated.

Conversely, if $(X, \sim)$ is a separated object, then by standard topos theory, it embeds into a sheaf (see for example [MLM94], section V.3). Therefore it is a subobject of an assembly, so it must be an assembly too. □

4.6 Assemblies versus Triposes

An alternative approach to the construction of a realizability topos uses the notion of a *tripos* . This construction is due to Hyland, Pitts and Johnstone in [HJP80] and stretches beyond realizability. Andrew Pitts further elaborates on tripos theory in his thesis [Pit81], where he also treats realizability triposes in depth. The book [vO08] also uses triposes.

The starting point of the construction is basically section 4.4. The pre-order we described, or more accurately the functor $\mathcal{P}(A)^{(-)}$ is an instance of what is called a *tripos*. Any tripos gives rise to a similar semantics, and the category consisting of equivalence relations as objects, and functional relations as morphisms (according to that semantics) turns out to be a topos.

The advantage of the approach is that it is more general, and the language is convenient. We have experienced that ourselves since we used strict relations to describe subobjects, so we were using triposes all along. However, for the special case of realizability toposes, the abstraction is often too much and not needed.

However, assemblies are much easier to understand, and contain as much information as the corresponding realizability topos. Also, assemblies have more of a computational nature to them (recall the similarity with programming languages in the introduction). For developing intuition on realizability toposes, I would say that assemblies are indispensable. So for someone who wishes to really understand the subject, I would advice to study both approaches, but that is something that applies to all of mathematics, really.

Chapter 5

Subtoposes of RT(A)

In this chapter we discuss subtoposes of a realizability topos on a partial combinatory algebra. We have already seen that Set is such a subtopos. It turns out that the theory of subtoposes of RT(A) is closely related to many constructions on pcas from chaper 2. First, we will quickly review geometric morphisms between realizability toposes.

5.1 Geometric Morphisms between Realizability Toposes

Here, we extend the theory of morphisms between categories of assemblies to morphisms between the corresponding realizability toposes. I will just state the main results that we need without proof. We will not need much of this treatment in detail. For proofs, I refer to [vO08], [Lon95] and [Joh13]. Let us recall what a geometric morphism between toposes is.

Definition 5.1.1. Let $\mathcal{E}, \mathcal{F}$ be toposes. A *geometric morphism* $f : \mathcal{F} \to \mathcal{E}$ is a pair of adjoint functors $f^* \dashv f_*$ where $f_* : \mathcal{F} \to \mathcal{E}$, $f^* : \mathcal{E} \to \mathcal{F}$ and f^* preserves finite limits.
We write Top for the category consisting of toposes as objects and geometric morphisms as arrows.

In the above, we have already seen the geometric morphism $\nabla : \mathrm{Set} \to \mathrm{RT}(A)$.
Recall definition 3.2.1. The following corresponds to Lemma 2.3.5 in [Lon95]:

Lemma 5.1.2. *Let A, B be pcas, let $I_B : Ass(A) \to RT(B)$ be the canonical inclusion. Then*

$$I_B \circ (-) : \nabla Reg(Ass(A), Ass(B)) \to \nabla Reg(Ass(A), RT(B))$$

is a natural equivalence of categories.

Theorem 5.1.3. *There is a natural equivalence of categories*

$$\nabla Reg(RT(A), RT(B)) \cong PCA(A, B).$$

Proof. By 4.3.3, the inclusion $I_A : \mathrm{Ass}(A) \to \mathrm{RT}(A)$ gives an equivalence

$$(-) \circ I_A : \nabla\mathrm{Reg}(\mathrm{RT}(A), \mathrm{RT}(B)) \to \nabla\mathrm{Reg}(\mathrm{Ass}(A), \mathrm{RT}(B)).$$

So we can apply lemma 5.1.2. Combining this lemma with propositions 3.3.6 and 3.3.8 yields the theorem. □

Under the above equivalence, an applicative morphism $\gamma : A \to B$ corresponds to the morphism $\mathcal{P}(\gamma)^* : \mathrm{RT}(A) \to \mathrm{RT}(B)$ given by

$$\mathcal{P}(\gamma)^*(X, \sim) = (X, \mathcal{P}(\gamma) \circ \sim)$$

A result by Hofstra and van Oosten [HvO03] is that *geometric morphisms* $f : \mathrm{RT}(B) \to \mathrm{RT}(A)$ for which $f^* \circ \nabla_B \cong \nabla_A$ correspond precisely to functors $\mathcal{P}(\gamma)^* : \mathrm{RT}(A) \to \mathrm{RT}(A)$ for which γ is computationally dense, which is the case if and only if $\mathcal{P}(\gamma)$ has a right adjoint Δ as proto-applicative morphisms.

It is a recent result by Peter Johnstone in [Joh13] that for *every* geometric morphism $f : \mathrm{RT}(A) \to \mathrm{RT}(B)$, we have in fact that $f^* \circ \nabla_A \cong \nabla_B$. So if $\mathrm{PCA}_{\mathrm{cd}}$ denotes the category of pcas and computationally dense morphisms, we get:

Theorem 5.1.4. *There is a natural equivalence of categories*

$$\mathit{Top}(\mathit{RT}(B), \mathit{RT}(A)) \cong \mathit{PCA}_{cd}(A, B).$$

From lemma 2.1 in [Joh13] we know that also the direct image functor f_* is a $\Gamma\nabla$-functor. Since assemblies are all subobjects of objects of the form ∇X, and f_* preserves regular monomorphisms, f_* restricts to a limit-preserving functor

$$f_* : \mathrm{Ass}(B) \to \mathrm{Ass}(A).$$

So by proposition 3.3.4, it is induced by some proto-applicative morphism $\Delta : B \to A$. Note that also f^* restricts to a functor on the level of assemblies, left-adjoint to f_*. It now follows by uniqueness of adjoints that $\mathcal{P}(\gamma) \dashv \Delta$. We therefore immediately know how f_* acts on assemblies $(X, E) \in \mathrm{Ass}(B)$:

$$f_*(X, E) \cong (X, \Delta \circ E).$$

For arbitrary objects $(X, \sim) \in \mathrm{RT}(B)$, giving an expression for $f_*(X, E)$ in terms of Δ is a little bit more subtle. The equivalent of equation 5.1 for f_* need not hold (yet it does for assemblies). If you are interested in an explicit expression, see section 2.5.2 of [vO08].

Remark We have shown here that every geometric morphism $f^* \dashv f_* : \mathrm{RT}(B) \to \mathrm{RT}(A)$ is induced by a unique adjoint pair of proto-applicative morphisms $\Gamma \dashv \Delta$. This result (using Johnstone's result) can already be deduced from the fact that every geometric morphism of such toposes is induced by a geometric morphism of *triposes* (see the discussion in section 4.6). The proof of this can be found in [Pit81] or [vO08]. However, working with assemblies is easier, and proposition 3.3.4 actually gives a little more information about the nature of these proto-applicative morphisms.

5.2 Local Operators

Recall that a geometric morphism of toposes $f : \mathcal{F} \to \mathcal{E}$ is an *embedding* if the direct image functor f_* is full and faithful. We also say that $\mathcal{F}$ is a *subtopos* of $\mathcal{E}$. One of the results in topos theory is that subtoposes correspond precisely to *Lawvere-Tierney topologies*, or *local operators* in the topos.

Recall that for any object $X \in \mathcal{E}$, where $\mathcal{E}$ is a topos, the lattice $\mathrm{Sub}(X)$ of subobjects of X is a Heyting algebra. Moreover, the isomorphism

$$\mathcal{E}(X, \Omega) \cong \mathrm{Sub}(X)$$

is natural in X, from which we obtain unique operations $\wedge, \vee, \to : \Omega \times \Omega \to \Omega$ that give Ω the structure of an internal Heyting algebra. For the conjunction $\wedge : \Omega \times \Omega \to \Omega$, this is derived as follows: we start

with a commutative diagram

$$\begin{array}{ccc} \mathrm{Sub}(X)\times\mathrm{Sub}(X) & \xrightarrow{\cap} & \mathrm{Sub}(X) \\ \downarrow\simeq & & \downarrow\simeq \\ \mathcal{E}(X,\Omega\times\Omega) & \xrightarrow{\wedge_X} & \mathcal{E}(X,\Omega) \end{array}.$$

Now for every $f\in\mathcal{E}(X,\Omega\times\Omega)$, we have a natural transformation $(-)\circ f:\mathcal{E}(\Omega\times\Omega,-)\to\mathcal{E}(X,-)$, so by naturality, $\wedge_X=\wedge_{\Omega\times\Omega}(1_{\Omega\times\Omega})\circ f$. So if we define $\wedge:\Omega\times\Omega\to\Omega$ as $\wedge_{\Omega\times\Omega}(1_{\Omega\times\Omega})$, then $\wedge\circ(-)$ is the natural transformation $\wedge_{(-)}:\mathcal{E}((-),\Omega\times\Omega)\to\mathcal{E}((-),\Omega)$. By the Yoneda lemma, $\wedge$ is the unique map $\Omega\times\Omega\to\Omega$ such that

$$\wedge\circ(-)\cong\wedge_{(-)}.$$

Hence $\wedge$ is uniquely defined.

The same goes for the operations $\vee,\to$, and this gives Ω the structure of an internal Heyting algebra. We can now define what a local operator is:

Definition 5.2.1. Let $\mathcal{E}$ be a topos, with subobject classifier $\top:1\to\Omega$, and meet operation $\wedge:\Omega\times\Omega\to\Omega$. A *local operator* on $\mathcal{E}$ is an arrow $j:\Omega\to\Omega$ such that:

(i) $j\circ\top=\top$

(ii) $j\circ j=j$

(iii) $j\circ\wedge=\wedge\circ(j\times j)$.

The internal Heyting structure on Ω induces an order on local operators:

$$j\le j'\iff\wedge\circ(j\times j')=j.$$

Local operators give rise to a closure operator $\overline{(-)}:\mathrm{Sub}(X)\to\mathrm{Sub}(X)$ as in definition 4.5.1, by the map $j\circ(-):\mathcal{E}(X,\Omega)\to\mathcal{E}(X,\Omega)$. The fact that this defines a closure operator follows easily from the definition. By proposition V.1 in [MLM94], we have in fact that every closure operator gives rise to a local operator in a topos. A subobject $A\rightarrowtail X$ is called *j-closed* (resp. *j-dense*) if it is closed (resp. dense) with respect to the closure operator defined by j. The definition of a *$j-$sheaf* is now the same as definition 4.5.2 for the closure operator defined by j, as is the definition of a *$j-$separated* object.

We now state some results on $j-$sheaves, for the proofs I refer to chapter V of [MLM94].

Theorem 5.2.2. *Let j be a local operator on a topos $\mathcal{E}$. Then the full subcategory $Sh_j(\mathcal{E})$ of $j-$sheaves is a topos and the inclusion $Sh_j(\mathcal{E})\to\mathcal{E}$ preserves finite limits and exponentials.*

Theorem 5.2.3. *Let j be a local operator on a topos $\mathcal{E}$. Then the inclusion $i:Sh_j(\mathcal{E})\to\mathcal{E}$ has a left adjoint $\mathbf{a}:\mathcal{E}\to Sh_j(\mathcal{E})$. Moreover, $\mathbf{a}$ preserves finite limits.*

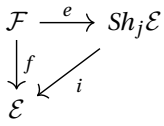
The above theorem shows that a local operator gives rise to a geometric inclusion $\mathbf{a}\dashv i$. The functor $\mathbf{a}$ is called the *associated sheaf functor*, or *sheafification functor*. For an object $X\in\mathcal{E}$, we call $\mathbf{a}(X)$ its *sheafification*. The following theorem corresponds to corollary VII.4.7 in [MLM94].

Theorem 5.2.4. *For a geometric morphism $f:\mathcal{F}\to\mathcal{E}$, the following are equivalent:*

(i) f is an embedding (i.e. f_* is full and faithful)

(ii) The counit $\epsilon_F:f^*f_*F\to F$ is an isomorphism for every $F\in\mathcal{F}$

(iii) There is a topology j on $\mathcal{E}$ and an equivalence of categories $e : \mathcal{F} \to Sh_j\mathcal{E}$ such that the following diagram of geometric morphisms commutes:



where $i : Sh_j\mathcal{E} \to \mathcal{E}$ is the inclusion.

5.2.1 Local Operators in RT(A)

Local operators in RT(A) can be presented in a particular nice way. Recall from section 4.4 that subobjects of an object $(X, \sim)$ in RT(A) are given by *strict relations* on X. A local operator

$$j : (\mathcal{P}(A), \Leftrightarrow) \to (\mathcal{P}(A), \Leftrightarrow)$$

determines a unique subobject of $(\mathcal{P}(A), \Leftrightarrow)$, which is in turn defined by a strict relation $\mathcal{J} : \mathcal{P}(A) \to \mathcal{P}(A)$. One can verify that local operators are in one-to-one correspondence with equivalence classes of maps (or strict relations) $\mathcal{J} : \mathcal{P}(A) \to \mathcal{P}(A)$ such that:

(i) $\Vdash (\forall P) P \to \mathcal{J}P$

(ii) $\Vdash (\forall P) \mathcal{J}\mathcal{J}P \to \mathcal{J}P$

(iii) $\Vdash (\forall P, Q)\,(P \to Q) \to (\mathcal{J}P \to \mathcal{J}Q)$.

In what follows, we will say that j is *represented* by $\mathcal{J}$. In the below, $\mathcal{J}$ is often just called a *local operator* .

We will first exhibit some results on these local operators by Andrew Pitts, taken from his thesis [Pit81]. After that, we will be able to see what subtoposes induced by a local operator $\mathcal{J}$ look like.

Recall that every map $\mathcal{P}(A) \to \mathcal{P}(A)$ is a strict relation on $(\mathcal{P}(A), \Leftrightarrow)$. We call $M : \mathcal{P}(A) \to \mathcal{P}(A)$ a *monotone operator* if

$$\Vdash (\forall P, Q)\,(P \to Q) \to (M(P) \to M(Q))\,.$$

Observe that any local operator is monotone. For M a monotone operator, we define $\mathcal{J}_M$ as follows:

$$\mathcal{J}_M(P) := (\forall Q)((M(Q) \to Q) \wedge (P \to Q)) \to Q.$$

We have the following proposition, which corresponds to lemma 5.4 in [Pit81]

Proposition 5.2.5. *(i) For any monotone operator M, $\mathcal{J}_M$ is a local operator, and $\mathcal{J}_M$ is the least local operator such that*

$$\Vdash (\forall P) M(P) \to \mathcal{J}_M(Q).$$

(ii) Let $(X, \sim) \in RT(A)$, and $R : X \to \mathcal{P}(A)$ a strict relation. Then the least local operator that makes $(X, \sim_R) \rightarrowtail (X, \sim)$ dense is $\mathcal{J}_M$ where

$$M(P) := [\![(\exists x)(R(x) \to P)]\!].$$

The ordering on local operators in the above representation coincides with the induced order on the strict relations on $(\Omega, \Leftrightarrow)$. Concretely, we have that $\mathcal{J} \leq \mathcal{J}'$ if and only if

$$\Vdash (\forall Q) \mathcal{J}Q \to \mathcal{J}'Q.$$

We have a smallest local operator $\mathcal{J}_\bot$ that does nothing:

$$\mathcal{J}_\bot(P) = P$$

and a biggest $\mathcal{J}_\top$:

$$\mathcal{J}_\top(P) = A.$$

Recall the closure operator defined in section 4.5 corresponding to the $\neg\neg$-sheaves. We denote the strict relation representing this local operator also by $\neg\neg$, and it is defined by:

$$\neg\neg(P) = \begin{cases} \emptyset & \text{if } P = \emptyset \\ A & \text{otherwise.} \end{cases}$$

The following corresponds to lemma 5.5 in [Pit81]. Strictly speaking, it is only proven for the effective topos. However, the proof still works. Note that one has to be careful here, because there are similar topos-theoretic properties that hold for the effective topos, but not for just any pca. In corollary 5.5.10, we will see an example of this.

Proposition 5.2.6. *Let M be a monotone operator. Then the following are equivalent:*

(i) $\mathcal{J}_M < \mathcal{J}_\top$

(ii) $\mathcal{J}_M \leq \neg\neg$

(iii) $M(\emptyset) = \emptyset$

Moreover, we have:

(iv) $\mathcal{J}_\bot < \mathcal{J}_M$ *if and only if*

$$\bigcap_{P \subseteq A} M(P) \to P = \emptyset$$

(v) $\neg\neg \leq \mathcal{J}$ *if and only if*

$$\bigcap_{a \in A} \mathcal{J}\{a\} \neq \emptyset.$$

The following lemma is easy, yet useful.

Lemma 5.2.7. *Every local operator $\mathcal{J}$ on $\mathrm{RT}(A)$ is isomorphic to one that preserves inclusions.*

Proof. We can define $\mathcal{J}'$ as:

$$\mathcal{J}'P := \bigcup_{Q \subseteq P} \mathcal{J}Q.$$

Then the identity $\mathsf{i} \in A$ together with an index a such that

$$a \Vdash (\forall P, Q)(Q \to P) \to (\mathcal{J}Q \to \mathcal{J}P)$$

yields that

$$a\mathsf{i} \Vdash (\forall P)\mathcal{J}'P \to \mathcal{J}P.$$

The other direction is realized by i. □

For subobject $(X, \sim_K) \rightarrowtail (X, \sim)$ of an object $(X, \sim) \in \mathrm{RT}(A)$, and a local operator j on $\mathrm{RT}(A)$ represented by $\mathcal{J}$, the j-closure is given by $(X, \sim_{\mathcal{J}\circ K})$. First, I'll show that above $\neg\neg$, there are no interesting local operators.

Proposition 5.2.8. *Let j be a local operator on RT(A), represented by $\mathcal{J}$. Then $\mathcal{J} > \neg\neg$ if and only if $\mathcal{J}\emptyset \neq \emptyset$, and in that case the subtopos defined by $\mathcal{J}$ is degenerate.*

Proof. By the lemma we may assume that $\mathcal{J}$ preserves inclusions.

Suppose $\mathcal{J}\emptyset \neq \emptyset$. Since we have that:

$$\Vdash (\forall P)(\emptyset \to P) \to (\mathcal{J}\emptyset \to \mathcal{J}P).$$

and for every P, $\emptyset \to P = A$, it follows that

$$\bigcap_{P\neq\emptyset} \mathcal{J}P \neq \emptyset.$$

Now $\neg\neg \leq \mathcal{J}$ by proposition 5.2.6(v), but we do not have equality since $\neg\neg\emptyset = \emptyset$.

Conversely, assume that $\neg\neg \leq \mathcal{J}$. Then by proposition 5.2.6(v) (and the fact that $\mathcal{J}$ preserves inclusions):

$$\bigcap_{P\neq\emptyset} \mathcal{J}P \neq \emptyset.$$

Because of this, we might as well assume that $\mathcal{J}P = A$ for all P non-empty. Now if $\mathcal{J}\emptyset = \emptyset$, it follows easily that $\neg\neg \cong \mathcal{J}$, because

$$\neg\neg P = \begin{cases} A & \text{if } P \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

Therefore we must have $\mathcal{J}\emptyset \neq \emptyset$.

In this case, it is easy to see that for $(X,\sim) \in \mathrm{RT}(A)$, every subobject is dense, even the empty subobject, which is an initial object. Therefore the only sheaves are terminal objects, hence the subtopos defined by j is trivial. □

For any local operator $\mathcal{J} \leq \neg\neg$, the objects $\nabla(X)$, for ∇ the constant objects functor, are sheaves, since they are precisely the $\neg\neg$-sheaves. Therefore, to see what the sheafification of an assembly $(X,E) \in \mathrm{RT}(A)$ is, we just have to define its $\mathcal{J}$-closure as a subobject of ∇X, which is given by the assembly $(X, \mathcal{J}\circ E)$.

Definition 5.2.9. We define define a subcategory $\mathrm{Ass}_{\mathcal{J}}(A)$ of $\mathrm{Ass}(A)$, called the $\mathcal{J}-$assemblies on A, as the full subcategory of objects of the form $(X, \mathcal{J}\circ E)$.

One can check that $\mathrm{Ass}_{\mathcal{J}}(A)$ is cartesian closed, it is also proven in [vO]. Moreover, $\mathrm{Ass}_{\mathcal{J}}(A)$ inherits regularity and finite colimits from $\mathrm{Ass}(A)$, since sheafification preserves those. Therefore the sheafification functor restricts to a full, faithful and regular functor $\mathrm{Ass}(A) \to \mathrm{Ass}_{\mathcal{J}}(A)$. Moreover:

Proposition 5.2.10. *The full subcategory $\mathrm{RT}(A)_{\mathcal{J}}$ of $\mathcal{J}$-sheaves is equivalent to $\mathrm{Ass}_{\mathcal{J}}(A)_{ex/reg}$.*

Proof. The embedding

$$\mathrm{Ass}_{\mathcal{J}}(A) \to \mathrm{RT}(A)_{\mathcal{J}}$$

is regular, full, faithful and surjective on subobjects. Because the sheafification functor is a left adjoint, it preserves colimits, so every object in $\mathrm{RT}(A)_{\mathcal{J}}$ is a quotient of a $\mathcal{J}-$assembly. By theorem 4.3.2, we must have an equivalence of categories $\mathrm{Ass}_{\mathcal{J}}(A)_{\mathrm{ex/reg}} \simeq \mathrm{RT}(A)_{\mathcal{J}}$. □

Hence, we obtain a full description of the subtopos of $\mathcal{J}$-sheaves by just looking at $\mathcal{J}$-assemblies. For a specific local operator (that we will discuss below), the category of $\mathcal{J}$-assemblies was studied in [vO].

5.2.2 Embeddings of Realizability Toposes

Recall from theorem 5.1.4 that geometric morphisms between realizability toposes correspond to computationally dense applicative morphisms. A natural question is, what do geometric *inclusions* correspond to? For $\gamma : A \to B$ computationally dense, $\mathcal{P}(\gamma)$ has a right adjoint Δ. For $(X, F) \in \mathrm{Ass}(B)$, we have by the discussion in section 5.1:

$$\mathcal{P}(\gamma)^* \Delta_* (X, F) = (X, \mathcal{P}(\gamma) \circ \Delta \circ F)$$

where for $P \subseteq A$:

$$\mathcal{P}(\gamma) \circ \Delta(P) = \bigcup_{a \in A} \{\gamma(a) \mid m\gamma(a) \subseteq P\}.$$

It is now not hard to see that we have a natural transformation $1 \to \mathcal{P}(\gamma)^* \Delta_*$ if and only if we have an $e \in B$ such that

$$e \in \bigcap_{P \subseteq B} \left(P \to \bigcup_{a \in A} \{\gamma(a) \mid m\gamma(a) \subseteq P\} \right).$$

In other words, for every $b \in B$, there is an $a \in A$ such that $eb \in \gamma(a)$ and $m\gamma(a) = \{b\}$. By proposition 2.5.10 and 2.5.17, we obtain:

Theorem 5.2.11. *Geometric inclusions*

$$RT(B) \to RT(A)$$

correspond precisely to effectively quasi-surjective applicative morphisms $A \to B$, which correspond precisely to adjoint pairs of proto-applicative morphisms $\Gamma \dashv \Delta$ such that $\Gamma\Delta \simeq I_B$.

5.3 Relative Recursion and Local Operators

In chapter 2, we have seen a lot of effectively quasi-surjective applicative morphisms; namely, for every partial function $f : A \to A$, we had a decidable applicative morphism $\iota : A \to A[f]$ given by $\iota_f(a) = \{a\}$, which is clearly effectively quasi-surjective. The same thing holds for the pcas $A[F]$, where $F : A^A \to A$ is a functional.

By the above, these constructions all give rise to subtoposes of $\mathrm{RT}(A)$. In this section, we will explore these subtoposes. Already in [Hyl82] there is given an embedding of the Turing degrees in the local operators, by defining for a total function $f : \mathbb{N} \to \mathbb{N}$, the least local operator j_f that "forces f to be recursive". To be precise, it is the least local operator that forces the subobject of $(\mathbb{N}, N)$, given by the strict relation $R(n) = \{f(n)\}$ to be dense. Here $(\mathbb{N}, N)$ is the natural numbers object of $\mathrm{RT}(\mathcal{K}_1) = \mathit{Eff}$, with $N(n) = \{n\}$. We can easily present this local operator using proposition 5.2.5. In [Pho89], it is shown that this subtopos, given by a local operator j_f, is equivalent to $\mathrm{RT}(\mathcal{K}_1^f)$. Here, we will carry out the same result for $\mathrm{RT}(A[f])$. Recall the definition of a local operator $\mathcal{J}_M$ for a monotone operator M:

$$\mathcal{J}_M(P) := (\forall Q)((M(Q) \to Q) \wedge (P \to Q)) \to Q.$$

The following lemma is due to Andrew Pitts [Pit81], but the formulation and the proof below is taken from [LvO13].

Lemma 5.3.1. *The operator $\mathcal{J}_M$ is isomorphic to $L(M)$ where:*

$$L(M)(P) = \bigcap \{Q \in \mathcal{P}(A) \mid \{\mathsf{T}\} \wedge P \subseteq Q \text{ and } \{\mathsf{F}\} \wedge M(Q) \subseteq Q\}.$$

Proof. Now let $e \in \mathcal{J}_M(P)$, and let $a = \langle x \rangle p \mathsf{T} x$, $b = \langle x \rangle p \mathsf{F} x$. Then if $\{\mathsf{T}\} \wedge P \subseteq Q$ and $\{\mathsf{F}\} \wedge M(Q) \subseteq Q$, then $a \in P \to Q$ and $b \in M(Q) \to Q$. Therefore, $e(pab) \in Q$, and hence $\langle e \rangle e(pab) \in \mathcal{J}_M(P) \to L(M)(P)$ for all P, so we have $\mathcal{J}_M \leq L(M)$.

Conversely, since $(\forall P) P \to \mathcal{J}_M(P)$ and $(\forall P) M(\mathcal{J}_M(P)) \to \mathcal{J}_M(P)$, we can take elements

$$k \in \bigcap_{P \in \mathcal{P}(A)} P \to \mathcal{J}_M(P)$$
$$l \in \bigcap_{P \in \mathcal{P}(A)} M(\mathcal{J}_M(P)) \to \mathcal{J}_M(P).$$

Since M is a monotone operator, we can define $d \in A$ such that:

$$d \in (\forall P)(\forall Q) P \to Q \to (M(P) \to M(Q)) = \bigcap_{P,Q \in \mathcal{P}(A)} P \to Q \to (M(P) \to M(Q))$$

By the recursion theorem, let c be such that:

$$ct \simeq \text{ if } p_0 t \text{ then } k p_1 t \text{ else } l(dc(p_1 t)).$$

Let P be arbitrary. Let $S = \{z \mid cz \in J_m(P)\}$. Then clearly $\{\mathsf{T}\} \wedge P \subseteq S$. Furthermore, $c : S \to \mathcal{J}_M(P)$, hence $dc : M(S) \to M(\mathcal{J}_M(P))$. So for $y \in M(S)$, $c(p \mathsf{F} y) \in \mathcal{J}_M(P)$. Hence also $\{\mathsf{F}\} \wedge M(S) \subseteq S$.

By definition of $L(M)$ it know follows that $c \in L(M)(P) \to \mathcal{J}_M(P)$ for all P (since $L(M)P \subseteq S$). Therefore $L(M) \leq J_M$. □

We write **A** for the assembly $(A, a \mapsto \{a\})$. The following proposition then generalizes the result in [Pho89] to arbitary pcas:

Proposition 5.3.2. *Let A be a pca, and $f : A \to A$ be a total function. Let $(A, R) \rightarrowtail \mathbf{A}$ be the subobject of* **A** *given by the strict relation*

$$R(a) = \{f(a)\}.$$

Then the local operator given by the inclusion $RT(A[f]) \to RT(A)$ is the least local operator that makes R dense.

Proof. The local operator that arises from the incusion $\mathrm{RT}(A[f]) \to \mathrm{RT}(A)$ is represented by

$$\mathcal{J} P = \{x \mid m \cdot^f x \in P\}$$

where m realizes computational density of the applicative morphism $A \to A[f]$. Let r_f be an index for f in $A[f]$. Then by definition of m, there exists $a' \in A$ such that for all $a \in A$:

$$m \cdot^f (a' a) \simeq r_f \cdot^f a = f(a).$$

Then $a' \Vdash (\forall x : A) \mathcal{J}(R(x))$, hence R is $\mathcal{J}$-dense.

Note that,by proposition 5.2.5 and the lemma 5.3.1, the least local operator that makes $(A, R) \rightarrowtail \mathbf{A}$ dense is isomorphic to $L(M)$, where:

$$M(P) = \bigcup_{a \in A} \{a\} \wedge (R(a) \to P) = \bigcup_{a \in A} \{a\} \wedge (\{f(a)\} \to P)$$

and

$$L(M)(P) = \bigcap \{Q \in \mathcal{P}(A) \mid \{\mathsf{T}\} \wedge P \subseteq Q \text{ and } \{\mathsf{F}\} \wedge M(Q) \subseteq Q\}.$$

We have to show that $\mathcal{J} \leq L(M)$.

Let P be arbitrary. Then if $x \in \mathcal{J}(P)$, $m \cdot^f x \in P$. Let u be the maximal dialogue between m and x,i.e.

$$m([x] * u) = p\mathsf{T}c \text{ with } c = m \cdot^f x.$$

In other words:

$$m([x] * u) \in \{\mathsf{T}\} \wedge P \subseteq L(M)(P).$$

Using the recursion theorem, define m' such that for all x, z:

$$m'([x] * z) \simeq \text{ if } p_0 m([x] * z) \text{ then } m([x] * z) \text{ else } p\mathsf{F}(p(p_1(m([x] * z))\langle w\rangle m'([x] * z * [w]))).$$

I'll prove that $\langle x\rangle m'([x]) : \mathcal{J}P \to L(M)(P)$ for all P. Clearly $m'([x] * u) \in L(m)(P)$. Now suppose that for some $0 < i \leq \mathsf{lh}\, u$, $m'([x] * u^{<i}) \in L(M)(P)$. Then:

$$m([x] * u^{<i-1}) = p\mathsf{F}v_i$$

and $u_i = f(v_i)$. We know that $m'([x] * u^{<i-1} * [u_i]) = m'([x] * u^{<i}) \in L(M)(P)$, so

$$\langle w\rangle m'([x] * u^{<i} * [w]) \in \{f(v_i)\} \to L(M)(P).$$

But then:

$$\begin{aligned} &p\mathsf{F}(p(p_1(m([x] * u^{<i-1}))\langle w\rangle m'([x] * u^{<i-1} * [w]))) \in \{\mathsf{F}\} \wedge (\{v_i\} \wedge (\{f(v_i)\} \to L(M)(P))) \\ &\subseteq \{\mathsf{F}\} \wedge M(L(M)(P)) \subseteq L(M)(P). \end{aligned}$$

hence we see that

$$m'([x] * u^{<i-1}) \in L(M)(P).$$

Therefore (by reverse induction) we deduce that

$$m'([x] * u^{<0}) = m'([x]) \in L(M)(P).$$

This all being uniform in P and x shows that $\langle x\rangle m'([x]) \in \mathcal{J}(P) \to L(M)(P)$ for all p.
Therefore $\mathcal{J}$ and $L(M)$ are isomorphic. □

For functionals, we can show a similar result. Recall that $\mathbf{A}^{\mathbf{A}} = (Z, E)$ where Z is the set of total recursive functions $A \to A$, and $E(f) = \{a \mid a \text{ is an index of } f\}$. It is easy to see that a functional $F : A^A \to A$ is an effective operation of type 2 (definition 2.7.1) if and only if its restriction to Z is realized as a morphism $\mathbf{A}^{\mathbf{A}} \to \mathbf{A}$.

For a local operator j represented by $\mathcal{J} : \mathcal{P}(A) \to \mathcal{P}(A)$, write $\mathbf{A}_j$ for the sheafification of $\mathbf{A}$ with respect to j, which is just the $\mathcal{J}$-assembly $(A, a \mapsto \mathcal{J}\{a\})$. Then $\mathbf{A}_j^{\mathbf{A}_j} \cong (Z, \mathcal{J}E)$ where Z is the set of morphism $\mathbf{A}_j \to \mathbf{A}_j$ in $\mathrm{Ass}_{\mathcal{J}}(A)$ and

$$E(f) = \bigcap_{a \in A} \mathcal{J}\{a\} \to \mathcal{J}\{f(a)\}.$$

Note that $\mathbf{A}_j^{\mathbf{A}} \cong \mathbf{A}_j^{\mathbf{A}_j}$. We now have the following theorem:

Theorem 5.3.3. *Let $\mathcal{J}_F$ be the local operator induced by the inclusion $RT(A[F]) \to RT(A)$. Suppose j is a local operator, represented by $\mathcal{J}$ such that the restriction of F to the domain of $\mathbf{A}_j{}^{\mathbf{A}_j}$ can be realized as a morphism of $\mathcal{J}-$assemblies*

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

Then it follows that $\mathcal{J}_F \leq \mathcal{J}$.

Proof. Recall that we can assume that $\mathcal{J}$ preserves inclusions.

Since $a \mapsto \{a\}$ is a computationally dense applicative morphism $A \to A[F]$, there exists $t \in A$ such that for all $v, x \in A$:

$$m \cdot^F (t\,v\,x) \simeq v \cdot^F x.$$

Since m is independent of F, we also know that if the right side halts at some stage (see definition 2.8.6), the left hand side also halts at that stage.

Pick e such that:

$$e \in \bigcap_{P,Q \subseteq A} (P \to Q) \to (\mathcal{J}P \to \mathcal{J}Q).$$

Let c be such that:

$$c \in \bigcap_{P,Q \subseteq A} (P \to \mathcal{J}Q) \to (\mathcal{J}P \to \mathcal{J}Q).$$

Moreover, we let $j \in A$ be such that for all u, v:

$$j \in \mathcal{J}\{u\} \to (\mathcal{J}\{v\} \to \mathcal{J}\{u * [v]\}).$$

Suppose that $F : \mathbf{A}_j{}^{\mathbf{A}} \to \mathbf{A}_j$ is tracked by r as morphism of $\mathcal{J}$-assemblies. Here $\mathbf{A}_j{}^{\mathbf{A}}$ is represented by the assembly (Z, E), where Z is the set of morphisms $\mathbf{A} \to \mathbf{A}_j$ in $\mathrm{Ass}(A)_{\mathcal{J}}$, and

$$E(f) = \bigcap_{a \in A} \{a\} \to \mathcal{J}\{f(a)\}.$$

We write $a =^{\mathcal{J}} b$ to express that if there is c such that $b \in \mathcal{J}\{c\}$, then $a \in \mathcal{J}\{c\}$. Convince yourself that there exists $\mathsf{m} \in A$ that satisfies for all x, w:

$$\mathsf{m}\,x\,w =^{\mathcal{J}} \begin{cases} e(\langle y\rangle \mathsf{p}_1(m([x] * y)))\,w & \text{if } e(\langle y\rangle \mathsf{p}_0(m([x] * y)))\,w \in \mathcal{J}\{\mathsf{T}\} \\ \mathsf{m}\,x(j\,w((c(\langle z\rangle r(\langle y\rangle(\mathsf{m}(t\,z\,y)\,l))))\,v)) & \text{if } e(\langle y\rangle \mathsf{p}_0(m([x] * y)))\,w \in \mathcal{J}\{\mathsf{F}\} \\ & \text{where } v = e(\langle y\rangle \mathsf{p}_1(m([x] * y)))\,w \end{cases}$$

Let $l \in \mathcal{J}\{[]\}$. We will prove by induction on the stage at which $m \cdot^F x \downarrow$ (see definition 2.8.6) that for all x,

$$\mathsf{m}\,x\,l \in \mathcal{J}\{m \cdot^F x\}.$$

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

$$\begin{aligned} &\langle y\rangle \mathsf{p}_0(m([x] * y)) \in \{[]\} \to \{\mathsf{T}\} \text{ so} \\ &e(\langle y\rangle \mathsf{p}_0(m([x] * y))) \in \mathcal{J}\{[]\} \to \mathcal{J}\{\mathsf{T}\} \text{ thus} \\ &\mathsf{m}\,x\,l = e(\langle y\rangle \mathsf{p}_1(m([x] * y)))\,l \in \mathcal{J}\{c\}. \end{aligned}$$

Stage $\lambda > 0$: Let $u = [u_0, \ldots, u_n]$ be the halting dialogue between m and x. We know that for each $0 \le i \le n$:

$$m([x] * u^{<i}) = \mathsf{p}\mathsf{F}\,v_i$$

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

$$\mathsf{m}(t\,v_i\,y)\,l \in \mathcal{J}\{v_i \cdot^F y\}$$

for each y, so $\langle y\rangle \mathsf{m}(t\,v_i\,y)\,l$ tracks f_i as morphism of $\mathcal{J}-$assemblies. So

$$r(\langle y\rangle(\mathsf{m}(t\,v_i\,y)\,l)) \in \mathcal{J}\{F(f_i)\}.$$

Hence

$$c(\langle v\rangle r(\langle y\rangle(\mathsf{m}(tvy)l))) \in \mathcal{J}\{v_i\} \to \mathcal{J}\{f_i\}$$

for all i. It follows that for all $0 \le i \le n$

$$\mathsf{m}x\mathcal{J}\{u^{<i}\} \subseteq \mathsf{m}x(jw((c(\langle z\rangle r(\langle y\rangle(\mathsf{m}(tzy)l))))\mathcal{J}\{v_i\})) \subseteq \mathsf{m}x\mathcal{J}\{[u_0,\ldots,u_i]\}.$$

One can now prove (use reverse induction on i) that

$$\mathsf{m}xl \in \mathcal{J}\{m \cdot^F x\}.$$

Therefore

$$\langle x\rangle \mathsf{m}xl \in \bigcap_{P\subseteq A} \mathcal{J}_F P \to \mathcal{J}P.$$

Hence $\mathcal{J}_F \le \mathcal{J}$. □

5.4 The Local Operator $L(F)$

In this section, we turn to a local operator in the effective topos defined by Andrew Pitts in [Pit81] and also studied in [vO]. It is defined as the least local operator that makes the subobject of $\nabla(\mathbb{N})$ defined by $R(n) = \{m \in \mathbb{N} \mid m \ge n\}$ dense. Combining proposition 5.2.5 and 5.3.1, we see that this local operator is isomorphic to $L(F)$ where F is the monotone operator defined by:

$$F(P) = \bigcup_{n\in\mathbb{N}} (R(n) \to P).$$

In the following, we will denote the corresponding local operator $L(F)$ by $\mathcal{J}$. An explicit description of $\mathcal{J}$ is given by:

$$\mathcal{J}(P) = \bigcap\{B \subseteq \mathbb{N} \mid \{0\} \wedge P \subseteq B, \{1\} \wedge F(B) \subseteq B\}.$$

Note we took $0,1$ instead of T,F, which makes no difference and is the convention in [vO].

We will first prove a result in [vO] as a corollary of theorem 5.3.3. Recall the functional $E : \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ defined in section 2.8.2, it is given by:

$$E(\alpha) = \begin{cases} 0 & \text{if } (\exists m)\alpha(m) = 0 \\ 1 & \text{otherwise .} \end{cases}$$

Denote by $N_{\mathcal{J}}$ the assembly with domain $\mathbb{N}$ defined by the map $n \to \mathcal{J}\{n\}$. We now have the following proposition:

Proposition 5.4.1. *There is an index $r_E \in \mathbb{N}$ that tracks E as morphism of $\mathcal{J}$-assemblies ${N_{\mathcal{J}}}^{N_{\mathcal{J}}} \to N_{\mathcal{J}}$.*

Proof. Pick elements

$$\mathsf{a} \in \bigcap_{A\subseteq\mathbb{N}} A \to \mathcal{J}A$$
$$\mathsf{b} \in \bigcap_{A,B\subseteq\mathbb{N}} (A \to B) \to (\mathcal{J}A \to \mathcal{J}B)$$
$$\mathsf{d} \in \bigcap_{A\subseteq\mathbb{N}} \mathcal{J}\mathcal{J}A \to \mathcal{J}A$$

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

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

we have

$$G(\langle x_0,\ldots,x_{n-1}\rangle) \in \mathcal{J}\{0\} \text{ if for some } i<n, a_i = 0$$
$$G(\langle x_0,\ldots,x_{n-1}\rangle) \in \mathcal{J}\{1\} \text{ otherwise}$$

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

$$t_E e n = G(\langle e(\mathsf{a}0),\ldots,e(\mathsf{a}n)\rangle)$$

Then whenever e realizes a total function $f : N_{\mathcal{J}} \to N_{\mathcal{J}}$, we $(\exists n) f(n) = 0$ if and only if $(\exists n) e(\mathsf{a}n) \in \mathcal{J}\{0\}$, which holds if and only if there is n such that

$$t_E e \in \{m \mid m \geq n\} \to \mathcal{J}\{0\}$$

since otherwise

$$t_E e \in \{m \mid m \geq 0\} \to \mathcal{J}\{1\}.$$

By definition of $\mathcal{J}$, we have in the first case that $t_E e \in \mathcal{J}\{0\}$, and in the latter case $t_E e \in \mathcal{J}\{1\}$. Now $r_E = \langle x\rangle \mathsf{c}(\mathsf{b} r_E x)$ tracks E as morphism of $\mathcal{J}-$assemblies ${N_{\mathcal{J}}}^{N_{\mathcal{J}}} \to N_{\mathcal{J}}$. □

The following corollary corresponds to theorem 2.2 in [vO].

Corollary 5.4.2. *The morphisms of $\mathcal{J}$-assemblies $N_{\mathcal{J}} \to N_{\mathcal{J}}$ are precisely the hyperarithmetical functions.*

Proof. Let $\mathcal{J}_E$ be the local operator defined by the inclusion $\mathrm{RT}(\mathcal{K}_1[E]) \to \mathrm{RT}(\mathcal{K}_1)$. By theorem 5.3.3, we have $\mathcal{J}_E \leq \mathcal{J}$. It is easily seen that a total Π^1_1 function is hyperarithmetical. Using theorem 2.8.10, it follows that every hyperarithmetical function is representable as a morphism $N_{\mathcal{J}} \to N_{\mathcal{J}}$.

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

$$f(n) = m \iff (\forall B)\{\langle 0, m\rangle\} \subseteq B \wedge \{1\} \wedge F(B) \subseteq B \to e(\mathsf{a}n) \in B$$

where e is a realizer for f as a morphism $N_{\mathcal{J}} \to N_{\mathcal{J}}$. Notice that the above expression is Π^1_1 since $F(B)$ is arithmetical in B.

Therefore the morphisms $N_{\mathcal{J}} \to N_{\mathcal{J}}$ are precisely the hyperarithmetical functions. □

5.5 Partial $\mathcal{J}$-representable Functions

In [vO], corollary 5.4.2 was actually proven for the so called total $\mathcal{J}-$representable functions (these are the elements of $N^N_{\mathcal{J}}$, which is isomorphic to $N^{N_{\mathcal{J}}}_{\mathcal{J}}$). We have the following definition:

Definition 5.5.1. A partial $\mathcal{J}$-representable function F is a partial function such that there exists an index e for which:

$n \in \mathrm{dom}\, F$ if and only if $(\exists m) e n \in \mathcal{J}\{m\}$, and in that case $F(n) = m$. Since such an m is necessarily unique we can index these functions by e and we say $F = \psi_e$.

This definition makes sense for any local operator $\mathcal{J}$ on a realizability topos such that:

$$\mathcal{J}\{a\} \cap \mathcal{J}\{b\} = \emptyset \text{ for all } a, b \text{ such that } a \neq b. \tag{5.1}$$

It is easy to see that this is true only if $\mathcal{J}\{\mathsf{T}\} \cap \mathcal{J}\{\mathsf{F}\} = \emptyset$ (so, $1+1 \rightarrowtail \nabla(2)$ is not $\mathcal{J}$-dense). For decidable pcas, it is equivalent to it.

Martin Hyland showed in [Hyl82] that in the effective topos $\mathrm{RT}(\mathcal{K}_1)$,

$$\bigcap_n \mathcal{J}\{n\} = \emptyset \iff \mathcal{J}\{0\} \cap \mathcal{J}\{1\} = \emptyset.$$

By proposition 5.2.6(iv) and the above, it follows that (5.1) holds for any local operator $\mathcal{J} < \neg\neg$ on $\mathrm{RT}(\mathcal{K}_1)$. One may wonder whether what Hyland showed is true for any partial combinatory algebra. In corollary 5.5.10 below, I show that this is not the case and provide a necessary and sufficient condition for this property, generalizing Hyland's result.

In what follows, we will assume that $\mathcal{J}$ is a local operator on $\mathrm{RT}(A)$ such that (5.1) holds. Note that the set of such local operators is downward closed.

First, the above definition induces an applicative structure $*$ on A, by defining:

$$a * a' \downarrow, a * a' = c \iff \psi_a(a') = c.$$

We will call this partial applicative structure $\mathcal{J}_1$. The natural question is whether $\mathcal{J}_1$ is a partial combinatory algebra. We cannot prove this in general, however, we can prove that $\mathcal{J}_1$ is a *weak* pca:

Definition 5.5.2. A partial applicative structure A is a *weak* partial combinatory algebra if there exists $\mathsf{k}, \mathsf{s} \in A$ such that for all $a, b \in A$:

(i) $\mathsf{s}ab \downarrow$

(ii) $\mathsf{k}ab = a$

(iii) $\mathsf{s}abc \preceq ac(bc)$.

Where we recall that for terms $t(x), s(x)$, $t(c) \preceq s(c)$ means that if $s(c) \downarrow$, then $t(c) \downarrow$ and $t(c) = s(c)$. A lot of properties of pcas can be proven for weak pcas, where in most cases one should replace $\simeq$ by $\preceq$. However, this is not a general rule and one should be careful. The most important fact is the analogue of lemma 2.2.3, which I'll state here (the proof is analogous).

Proposition 5.5.3. *Let A be a weak pca. Then for any term $t(x, x_1, \ldots, x_n)$ there is a term $\langle x \rangle t(x_1, \ldots, x_n)$ such that every substitution instance of $\langle x \rangle t$ denotes, and for all $a, a_1, \ldots, a_n \in A$*

$$(\langle x \rangle t)(a_1, \ldots, a_n)a \preceq t(a, a_1, \ldots, a_n).$$

Conversely, if A has a partial applicative structure for which the above holds, then A is a weak pca.

The notion of applicative morphisms for weak pcas is the same. I'll now present a proof that every presentation $\mathcal{J}$ of a local operator yields a weak pca $\mathcal{J}_1$. After that, it is easy to see that isomorphic local operators yield isomorphic weak pcas, that is, the weak pca $\mathcal{J}_1$ is independent of the choice of presentation $\mathcal{J}$. Lastly, we will see that in fact these weak pcas are isomorphic to a partial combinatory algebra.

Theorem 5.5.4. *For $\mathcal{J}$ be a presentation of a local operator on $RT(A)$ that satisfies* (5.1)*, the partial applicative structure $\mathcal{J}_1$ is a weak pca.*

Proof. We have to find elements $\mathsf{k}, \mathsf{s} \in \mathbb{N}$ as in definition 5.5.2. We denote the applicative structure on A as usual (just as terms), and the applicative structure on $\mathcal{J}_1$ by $*$

First, pick

$$\mathsf{a} \in \bigcap_{P \subseteq A} P \to \mathcal{J}P.$$

We have for all a' that $\mathsf{a}a' \in \mathcal{J}\{a'\}$. Let $k_0 \in A$ be such that $k_0ab = a$ for all $a, b \in A$. Then $\mathsf{k} = \langle x \rangle \mathsf{a}(k_0(\mathsf{a}x))$ satisfies (ii) in 5.5.2.

It remains to define s. Let a, b, c be arbitrary indices. We assume that $ac(bc)\downarrow$, so there are u, v, m such that

$$ac \in \mathcal{J}\{u\}$$
$$bc \in \mathcal{J}\{v\}$$
$$uv \in \mathcal{J}\{m\}$$

and $ac(bc) = m$.

Take the following elements:

$$r \in \bigcap_{P,Q\subseteq A} \mathcal{J}P \wedge \mathcal{J}Q \to \mathcal{J}(P \wedge Q)$$
$$p \in \bigcap_{P\subseteq \mathcal{J}Q,\, P,Q\subseteq A} \mathcal{J}P \to \mathcal{J}Q$$
$$t \in \bigcap_{P,Q\subseteq A} (P \to Q) \to (\mathcal{J}P \to \mathcal{J}Q)$$

As to the element p, we know that any local operator $\mathcal{J}$ is isomorphic to a local operator $\mathcal{I}$ that preserves inclusions. Taking such a local operator, we have the following elements:

$$\alpha \in \bigcap_{P\subseteq A} \mathcal{J}P \to \mathcal{I}P,\ \beta \in \bigcap_{P\subseteq \mathbb{N}} \mathcal{I}P \to \mathcal{J}P,\ \gamma \in \bigcap_{P\subseteq A} \mathcal{I}\mathcal{J}P \to \mathcal{I}P.$$

Then $p = \langle x\rangle \beta(\gamma(\alpha x))$ works.

Now, let $h = \langle yx\rangle \mathsf{p}(\mathsf{p}_0 xy)(\mathsf{p}_1 xy)$. Then

$$hc \in \{a\} \wedge \{b\} \to \mathcal{J}\{u\} \wedge \mathcal{J}\{v\}.$$

Now let $g = \langle x\rangle \mathsf{p}_0 x(\mathsf{p}_1 x)$ and $\overline{h} = \langle zx\rangle\, tg(r(hzx))$. Then

$$p(\overline{h}c) \in \{a\} \wedge \{b\} \to \mathcal{J}\{m\}$$

since:

$$\{a\} \wedge \{b\} \xrightarrow{hc} \mathcal{J}\{u\} \wedge \mathcal{J}\{v\} \xrightarrow{r} \mathcal{J}\{\langle u, v\rangle\} \xrightarrow{tg} \mathcal{J}\underbrace{\{uv\}}_{\subseteq \mathcal{J}\{m\}} \xrightarrow{p} \mathcal{J}\{m\}$$

Now if we let $f = \langle zyx\rangle\, p(\overline{h}z)(\mathsf{p}xy)$ then for all $a', b' \in A$, $fa'b'\downarrow$ and

$$fab \in \{c\} \to \mathcal{J}\{m\}.$$

Now define $\mathsf{s} = \langle x\rangle \mathsf{a}(\langle y\rangle \mathsf{a}(fxy))$. Then

$$s \in \{a\} \to \mathcal{J}\{\langle y\rangle \mathsf{a}(fay)\}$$

and since

$$\langle y\rangle \mathsf{a}(fay) \in \{b\} \to \mathcal{J}\{fab\}$$

We can see that $\mathsf{s} * a * b * c = fab * c = m$. Since s is defined uniformly in a, b, this finishes the proof. □

Corollary 5.5.5. *Up to isomorphism of weak pcas, $\mathcal{J}_1$ does not depend on the choice of presentation $\mathcal{J}$ for a local operator.*

Proof. Suppose $\mathcal{J} \cong \mathcal{J}'$, we want to show that $\mathcal{J}_1 \cong \mathcal{J}'_1$ as weak pcas. Take

$$\mathsf{a} \in \bigcap_{P \subseteq A} P \to \mathcal{J}' A$$
$$r \in \bigcap_{P \subseteq A} \mathcal{J} P \to \mathcal{J}' P$$

Then $a \mapsto \{a\}$ is realized as applicative morphism $\mathcal{J}_1 \to \mathcal{J}'_1$ by

$$\langle x \rangle \mathsf{a}(\langle y \rangle r(xy))$$

and the other direction is symmetric. □

Theorem 5.5.6. *For every local operator $\mathcal{J}$ for which* (5.1) *holds, we have that $\mathcal{J}_1 \cong A[f]$ where*

$$f(a) = \begin{cases} b & \text{if } a \in \mathcal{J}\{b\} \\ \text{undefined} & \text{otherwise} \end{cases}$$

Proof. First, let $r \in A$ such that for all $a, b \in A$:

$$r\,a([b]) = \mathsf{pF}(ab)$$
$$r\,a([b,c]) = \mathsf{pT}\,c$$

Then if r' is an index such that $r' \cdot^f a = r\,a$, then r' realizes $a \mapsto \{a\}$ as an applicative morphism $\mathcal{J}_1 \to A[f]$.

The converse can be seen by generalizing theorem 2.6.3 to weak pcas, which can be done, a corollary is then that $a \mapsto \{a\}$ has a realizer as morphism $A[f] \to \mathcal{J}_1$ since in $\mathcal{J}_1$, f is represented by the identity $\mathsf{i} = \mathsf{skk}$, and it is decidable.

One can also give a direct proof by simulating computations in $A[f]$, just like in the proof theorem 2.7.4. □

Let us apply the above to the local operator $\mathcal{J} = L(F)$ from section 5.4.

By theorem 5.5.6, $\mathcal{J}_1 \cong \mathcal{K}_1[f]$ where f is defined by:

$$f(n) = \begin{cases} m & \text{if } n \in \mathcal{J}\{m\} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

From section 5.4, we can infer that the functional E is representable with respect to $\mathcal{K}_1 \to \mathcal{K}_1[f]$ defined by $n \mapsto \{n\}$. By theorem 2.7.4, $n \mapsto \{n\}$ factors through as an applicative morphism $\mathcal{K}_1[E] \to \mathcal{K}_1[f]$. Conversely, f is a Π^1_1 function, so it is representable with respect to $n \mapsto \{n\} : \mathcal{K}_1 \to \mathcal{K}_1[E]$ (as a consequence of theorem 2.8.12). Hence $n \mapsto \{n\}$ is also realized as applicative morphism $\mathcal{K}_1[f] \to \mathcal{K}_1[E]$ (theorem 2.6.3). We have the following corollary:

Corollary 5.5.7. *For $\mathcal{J} = L(F)$ defined as above, the weak pca $\mathcal{J}_1$ of partial $\mathcal{J}$-representable functions is isomorphic to $\mathcal{K}_1[E]$.*

5.5.1 When does Forcing ∇ to Preserve Coproducts Collapse the Topos to Set?

Here, I wish to elaborate on the remark that I made about the condition (5.1). The question is, when does it hold for all local operators $\mathcal{J} < \neg\neg$? In other words, when does, when we try to force $1 + 1 \rightarrowtail \nabla(2)$ isomorphic, the topos collapse to Set?

In RT(A), $1+1$ may be represented by the assembly $(\{0,1\}, E)$ where $E(0) = \{\mathsf{T}\}$, $E(1) = \{\mathsf{F}\}$. The least local operator that forces $1+1 \rightarrowtail \nabla(2)$ dense is $L(M)$ (section 5.3.1) where

$$M(P) = (\{\mathsf{T}\} \to P) \cup (\{\mathsf{F}\} \to P).$$

Let's see what $L(M)(P)$ looks like for $P \subseteq A$:

$$L(M)(P) = \bigcap \{Q \subseteq A \mid \{\mathsf{T}\} \wedge P \in Q \text{ and } \{\mathsf{pF}e \mid e \in (\{\mathsf{T}\} \to Q) \cup (\{\mathsf{F}\} \to Q)\} \subseteq Q\}$$

Define a family of sets $V_i(P), i \in \mathbb{N}$ as follows:

$$\begin{aligned} V_0(P) &= \{\mathsf{T}\} \wedge P \\ V_{n+1}(P) &= V_n(P) \cup \{\mathsf{F}\} \wedge ((\{\mathsf{T}\} \to V_n(P)) \cup (\{\mathsf{F}\} \to V_n(P))) \end{aligned}$$

Now let $V(P) = \bigcup_{n\in\mathbb{N}} V_n(P)$. It is not hard to see that for all $P \subseteq A$, $V(P) \subseteq L(M)(P)$. Moreover:

$$\{\mathsf{F}\} \wedge M(V(P)) \subseteq V(P).$$

Hence we have that $V(P) = L(M)(P)$.

So $\mathsf{pF}e \in L(M)(P)$ if and only if there is some finite sequence of T's and F's such that

$$e\mathsf{TF}\ldots\mathsf{TF} = \mathsf{pT}a \text{ for some } a \in P.$$

We have achieved already one thing here; if $\bigcap_{a\in A} L(M)\{a\}$ is non-empty, then A must be countable! But there is more.

We start with two lemmas:

Lemma 5.5.8. *Define a map $\mathcal{I} : \mathcal{P}(A) \to \mathcal{P}(A)$ by:*

$$\mathcal{I}P = (\{\mathsf{T}\} \wedge P) \cup \left(\{\mathsf{F}\} \wedge \Big(\bigcup_{u=[u_0,\ldots,u_n], u_i \in \{\mathsf{T},\mathsf{F}\}, n\in\mathbb{N}} \{u\} \to \{\mathsf{T}\} \wedge P\Big)\right).$$

Then $\mathcal{I}$ is a local operator and $\mathcal{I} \cong L(M)$.

Proof. Convince yourself that there is an index $g \in A$ such that for all $m \in \mathbb{N}$:

$$\begin{aligned} g(\mathsf{pF}e)[u_0,\ldots,u_m] &= eu_0\ldots u_m \\ g(\mathsf{pT}e) &= e \end{aligned}$$

Then $\langle x\rangle \mathsf{p}(\mathsf{p}_0 x)(gx)$ realizes $L(M) \leq \mathcal{I}$.

Let $S \in A$ be a successor function, i.e. $S(\overline{n}) = \overline{n+1}$ for all $n \in \mathbb{N}$. By the recursion theorem, there is $t \in A$ be such that:

$$tx \simeq \langle v\rangle vx(t(Sx)) = \mathsf{if}\ v\ \mathsf{then}\ x\ \mathsf{else}\ t(Sx)$$

Then for each $n \in \mathbb{N}$:

$$t\overline{n} \simeq \mathsf{if}\ v\ \mathsf{then}\ \overline{n}\ \mathsf{else}\ t\overline{n+1}.$$

So for each $n \in \mathbb{N}$, $t\overline{n}\mathsf{T} = \overline{n}$, and

$$t\overline{n} \in \{\mathsf{F}\} \to (\{\mathsf{T}\} \to \overline{n+1}).$$

It is now not hard to see that for all $m \in \mathbb{N}$:

$$t\overline{0}\underbrace{\mathsf{FF}\ldots\mathsf{F}}_{m \text{ times}}\mathsf{T} = \overline{m}.$$

Using the above, it is easy to see that there is an index $r \in A$ such that for all $m \in \mathbb{N}, e \in A$:

$$r(\mathsf{p}\mathsf{F}e)\underbrace{\mathsf{F}\mathsf{F}\ldots\mathsf{F}}_{m \text{ times}}\mathsf{T}u_0\ldots u_m \simeq e[u_0,\ldots,u_m]$$
$$r(\mathsf{p}\mathsf{T}e) = e$$

so $\langle x\rangle \mathsf{p}(\mathsf{p}_0 x)(r\,x)$ realizes $\mathcal{I} \leq L(M)$. We have also proven that $\mathcal{I}$ is a local operator. □

Lemma 5.5.9. *Define a map $\mathcal{J} : \mathcal{P}(A) \to \mathcal{P}(A)$ by:*

$$\mathcal{J}P = (\{\mathsf{T}\} \wedge P) \cup (\bigcup_{n\in\mathbb{N}} \{\overline{n}\} \to \{\mathsf{T}\} \wedge P).$$

Then $\mathcal{J} \cong L(M)$.

Proof. It is enough to show that $\mathcal{J} \cong \mathcal{I}$ for $\mathcal{I}$ as in the above lemma.

Again, convince yourself that we have an index $h \in A$ that enumerates all tuples $[u_0,\ldots,u_m]$ where $u_i \in \{\mathsf{T},\mathsf{F}\}$. This is not hard to see because it involves checking on the length of the tuple, checking all values, and binary addition with 1:

$$h\overline{0} = [\mathsf{F}]$$
$$h\overline{1} = [\mathsf{T}]$$
$$h\overline{2} = [\mathsf{F},\mathsf{F}] \text{ etcetera.}$$

Moreover, since we can "wait for a tuple to appear", there is an index g such that for all $n, m \in \mathbb{N}$, $u_i \in \{\mathsf{T},\mathsf{F}\}$:

$$g[u_0,\ldots,u_n] = \overline{m} \text{ where } h\overline{m} = [u_0,\ldots,u_n].$$

We do not need decidability for this!

Now let $r \in A$ be such that:

$$r(\mathsf{p}\mathsf{F}e) = \mathsf{p}\mathsf{F}(\langle y\rangle e(h\,y))$$
$$r(\mathsf{p}\mathsf{T}e) = e$$

and $t \in A$ such that:

$$t(\mathsf{p}\mathsf{F}e) = \mathsf{p}\mathsf{F}(\langle y\rangle e(g\,y))$$
$$t(\mathsf{p}\mathsf{T}e) = e$$

Then $\langle x\rangle \mathsf{p}(\mathsf{p}_0 x)(r\,x)$ realizes $\mathcal{I} \leq \mathcal{J}$, and $\langle x\rangle \mathsf{p}(\mathsf{p}_0 x)(t\,x)$ realizes $\mathcal{J} \leq \mathcal{I}$. □

Corollary 5.5.10. *For $RT(A)$ a realizability topos on a pca A, $\neg\neg$ is the least local operator that forces $1+1 \rightarrowtail \nabla(2)$ dense if and only if there is an A-computable partial surjection $\mathbb{N} \to A$; that is, an index $g \in A$ such that for all $a \in A$, there is $n \in \mathbb{N}$ such that*

$$g\overline{n} = a.$$

Proof. The least local operator that forces $1+1 \rightarrowtail \nabla(2)$ dense is $L(M)$ above, and it is isomorphic to $\neg\neg$ if and only if $\bigcap_{a\in A} L(M)\{a\} \neq \emptyset$ by 5.2.6(v). Let $\mathcal{J}$ be as in the above lemma. Then $\bigcap_{a\in A} L(M)\{a\} \neq \emptyset$ if and only if $\bigcap_{a\in A} \mathcal{J}\{a\} \neq \emptyset$. The latter holds if and only if there exists $h \in A$ such that for all $a \in A$, there is an $n \in \mathbb{N}$ with:

$$h\overline{n} = \mathsf{p}\mathsf{T}a.$$

Of course this implies the right hand side, and the converse. □

Remark Jaap van Oosten attended me on the fact that "existence of a partial computable surjection" $\mathbb{N} \to A$ is the same as the property that

$$n \mapsto \{\overline{n}\} : \mathcal{K}_1 \to A \text{ is quasi-surjective (equivalently, computationally dense).}$$

The following lemma (due to Longley) shows that this condition is equivalent to the existence of *any* computationally dense applicative morphism $\mathcal{K}_1 \to A$.

Lemma 5.5.11. *If $\gamma : \mathcal{K}_1 \to A$ is computationally dense, then $\gamma \simeq \nu$, where $\nu(n) = \{\overline{n}\}$.*

Proof. By proposition 2.5.12, γ is decidable. Now it follows from proposition 2.4.18 in [Lon95] that γ is equivalent to ν. □

Using the lemma, we could reformulate the above result as:

Theorem 5.5.12. *For RT(A) a realizability topos on a pca A, there is a geometric morphism RT(A) → RT($\mathcal{K}_1$) if and only if $\neg\neg$ is the least local operator that forces $1 + 1 \rightarrowtail \nabla(2)$ dense (in RT(A)). Moreover, such a geometric morphism is unique up to equivalence.*

So for a pca A that does not admit a geometric morphism $\mathrm{RT}(A) \to \mathrm{RT}(\mathcal{K}_1)$, we can force $\nabla : \mathrm{Set} \to \mathrm{RT}(A)$ to preserve finite coproducts without collapsing $\mathrm{RT}(A)$ to Set. In the resulting subtopos of $\mathrm{RT}(A)$, the natural numbers object is isomorphic to $\nabla(\mathbb{N})$, so all functions $\mathbb{N} \to \mathbb{N}$ exist. It would be interesting to construct a countable such pca A; its realizability topos must have remarkable properties.

The following is the analogue of proposition 3 in [Pho89]. For a pca A, denote the local operator corresponding to the inclusion $\mathrm{RT}(A[f]) \to \mathrm{RT}(A)$ for some $f : A \to A$ by $\mathcal{J}_f$.

Theorem 5.5.13. *Let $\mathcal{J}$ be a local operator on RT(A) such that $\mathcal{J} \geq \mathcal{J}_f$ for all total $f : A \to A$. Then*

$$\mathcal{J}\{\mathsf{T}\} \cap \mathcal{J}\{\mathsf{F}\} \neq \emptyset.$$

Proof. Suppose that $\mathcal{J}\{\mathsf{T}\} \cap \mathcal{J}\{\mathsf{F}\} = \emptyset$, I'll derive a contradiction. Define $g : A \to A$ by:

$$g(a) = \begin{cases} \mathsf{T} & \text{if } a \in \mathcal{J}\{\mathsf{T}\} \\ \mathsf{F} & \text{otherwise .} \end{cases}$$

By assumption, g is a function.

Pick a total function $f : A \to \{\mathsf{T}, \mathsf{F}\}$ such that f is not computable in $A[g]$, this can be done for cardinality reasons.

Then f is representable as a morphism $\mathbf{A}_j \to \mathbf{A}_j$ by proposition 5.3.2. We can infer that there is $e \in A$ such that for all $a \in A$:

$$ea \in \mathcal{J}\{f(a)\}$$

But then f is clearly computable in $A[g]$, which is contradictory. □

The exact statement of proposition 3 in [Pho89] is that, under the assumptions, we have in fact that $\mathcal{J} \geq \neg\neg$, or equivalently:

$$\bigcap_{a \in A} \mathcal{J}\{a\} \neq \emptyset.$$

We can prove this in the countable case:

Corollary 5.5.14. *Let A be a countable pca. Let $\mathcal{J}$ be a local operator on RT(A) such that $\mathcal{J} \geq \mathcal{J}_f$ for all total $f : A \to A$. Then $\mathcal{J} \geq \neg\neg$.*

Proof. Let $g : A \to A$ be a total function such that for all $a \in A$, there is $n \in \mathbb{N}$ such that $g(\overline{n}) = a$. Since A is countable, there is such a map. It is now not hard to see that $A[g]$ satisfies the right hand side in the statement of corollary 5.5.10. Since $\mathcal{J} \geq \mathcal{J}_g$ by assumption, the $\mathcal{J}-$sheaves are a subtopos of $\mathrm{RT}(A[g])$ and by corollary 5.5.10, $\mathcal{J} \geq \neg\neg$. □

I leave it as an open question whether this holds in the uncountable case.

5.6 Lifting Geometric Morphisms of Realizability Toposes

In this short section, we will apply the notion of *lifting* from section 2.9 to geometric morphisms of realizability toposes. Recall from theorem 5.1.4 that a geometric morphism $f : \mathrm{RT}(B) \to \mathrm{RT}(A)$ is induced by some computationally dense morphism $\gamma : A \to B$. Since we can only lift discrete such morphisms, we will assume from now on that A is decidable, so that any computationally dense $\gamma : A \to B$ is discrete. It may be interesting to find out what geometric morphisms are induced by discrete computationally dense morphisms in the case that A is not decidable.

Definition 5.6.1. A geometric morphism $f : \mathcal{F} \to \mathcal{E}$ between toposes is a *surjection* if the inverse image f^* is faitful.

Definition 5.6.2. A geometric morphism $f : \mathcal{F} \to \mathcal{E}$ between toposes is *exact* if the direct image f_* is exact (i.e. regular).

Theorem 5.2.11 and 2.9.3 yield an interesting result:

Theorem 5.6.3. *Let $i : RT(B) \to RT(A)$ be a geometric inclusion. Then there is a geometric surjection $RT(A[f]) \to RT(B)$ for some f. If i is exact, then $RT(B) \simeq RT(A[f])$.*

Proof. By theorem 5.2.11, i is induced by an applicative morphism $\gamma : A \to B$ for which there is a $\delta : B \to A$ with $\gamma\delta \simeq \iota_B$. By theorem 2.9.3, γ lifts to a retraction $\delta : B \to A[f]$, so that δ induces a geometric surjection

$$\mathrm{RT}(A[f]) \to \mathrm{RT}(B).$$

If i is exact, then it was induced by a pair $\gamma \dashv \delta$ such that $\gamma\delta \simeq \iota_B$. By theorem 2.9.2, the lift of γ is an equivalence, so $\mathrm{RT}(B) \simeq \mathrm{RT}(A[f])$. □

For the effective topos, this theorem has an important consequence:

Corollary 5.6.4. *For $RT(A) \to RT(\mathcal{K}_1)$ any geometric inclusion, $RT(A) \cong RT(\mathcal{K}_1[f])$ for some (partial) f.*

Proof. By lemma 5.5.11 and theorem 5.5.12, any geometric inclusion $\mathrm{RT}(A) \to \mathrm{RT}(\mathcal{K}_1)$ is induced by the computationally dense morphism $\nu : \mathcal{K}_1 \to A$. Since this morphism is projective, ν has a right adjoint $\delta : A \to \mathcal{K}_1$ (theorem 2.5.16). So $f^* \dashv f_*$ is induced by the pair $\nu \dashv \delta$, which implies that f_* is regular. So f is exact, and by theorem 5.6.3 $\mathrm{RT}(A) \cong \mathrm{RT}(\mathcal{K}_1[f])$. □

Note that the property that every inclusion $\mathrm{RT}(A) \to \mathrm{RT}(\mathcal{K}_1)$ is exact is a specific property of the effective topos. Recall from example 2.5.18 that there is a computationally dense applicative morphism $\gamma : \mathcal{K}_{2\mathrm{rec}} \to \mathcal{K}_1$ that sends each function to its set of indices. It is not hard to see that this is in fact an inclusion, i.e. it is effectively quasi-surjective. So we have a geometric inclusion

$$\mathrm{RT}(\mathcal{K}_1) \to \mathrm{RT}(\mathcal{K}_{2\mathrm{rec}})$$

that is not exact, since γ does not have right adjoint as applicative morphism.

Chapter 6

Conclusions

Code-free recursion and realizability

The notion of a partial combinatory algebra is an interesting attempt to describe *abstract* or *code-free* recursion theory. Or, since it is not very clear what we mean by that, it is at least an interesting attempt to describe what one could mean by code-free recursion theory. It yields a powerful mathematical theory that is worthwhile investigating. Its connections to category theory, mathematical logic and also to branches in theoretical computer science support this viewpoint.

At least for me, a problem with partial combinatory algebras is a lack of examples, especially in the case of *decidable* pcas. In fact, some result of this thesis (e.g. corollary 5.6.4) are evidence that up to equivalence and adjoining a partial function, there might be only a few examples, or at least a few that can be presented in a nice way. However, there are still several examples of total pcas, and there are pcas of higher cardinality.

The justification of applicative morphisms is their correspondence to exact functors between categories of assemblies, or realizability toposes. However, they do make sense from a computational point of view. In that light, the correspondence between computationally dense morphisms and geometric morphisms is remarkable. This correspondence has also been studied using triposes. I have shown in this thesis that if one wants to use assemblies instead, proto-applicative morphisms can be useful since they basically play the role of morphisms between triposes. Moreover, computationally dense applicative morphisms have an easy description using proto-applicative morphisms. Still, the old definition as discovered by Hofstra and van Oosten has been very useful in many proofs.

I hope to have shed some light on the importance of relative recursion in the study of realizability toposes. For instance, we have corollary 5.6.4 that classifies all geometric inclusions $\mathrm{RT}(A) \to \mathrm{RT}(\mathcal{K}_1)$. Using relative recursion one is able to clearify the relations between different pcas and realizability toposes. This realization might help to understand realizability toposes, and with it the mystical idea of abstract recursion theory.

This also motivates the new notion of recursive functionals as defined in section 2.8.1, and the related theory on effective operations in section 2.7. This theory generates new pcas from old ones with a very interesting and easy description. In this way we obtain insightful presentations of subtoposes that are realizability toposes themselves. Such a presentation is desirable to, for example, understand the logic of a subtopos. We have seen that a description of a subtopos using a functional allows us to determine its complexity, which has a lot of implications for its logic. There is a connection to local operators (section 5.3) that goes beyond the embedding of the Turing degrees in the lattice of local operators that was already described in [Hyl82]. This connection still needs some clarification. I especially hope to find something that also relates functionals of higher type (type 3 and up) to local operators.

Where to go from here

There is a lot left to explore. For instance, a lot of subtoposes of the effective topos are not realizability toposes. However, they still have a computational nature to them. What characterizes this computational nature?

We have shown that one can express computational density by just using morphisms, so without explicit reference to the combinatorial structure. A lot of examples of pcas, especially total pcas, arise from a retraction of a certain partial order into the space of continuous automorphisms (see [Hof]). It might be that a theory on pcas can be built up by mostly looking at morphisms. For example, certain functions can be computable with respect to an applicative morphism. When lifting an applicative morphism, one basically translates this added complexity back to the partial combinatory algebra.

Another direction is the possible generalizations of pcas. There is the notion of an *order pca* (introduced in [HvO03]), which has a slightly more general definition but also gives rise to assemblies and realizability toposes. A stronger notion than that, but weaker than a pca is the *weak* pca from definition 5.5.2. I have the feeling that every weak pca is isomorphic (or equivalent) to a pca (like the weak pcas in theorem 5.5.6), since applicative morphisms "are not able to tell them apart". However, this remains a mere conjecture since I haven't been able to prove nor refute this. [1]

The more categorical approaches to the study of realizability toposes are topos theory in general, but also the theory of triposes. Another interesting approach is to look at exact/regular completions of categories. Here I find [Men00] to be an excellent source, it treats assemblies in detail.

Lastly, I think there can be made improvement in code-free recursion theory by restricting to particular partial combinatory algebras. A lot of the structure in ordinary recursion theory does not generalize to pcas. However, it seems to me that some results of ordinary recursion theory do generalize to, for instance, pcas that satisfy the statements in corollary 5.5.10. The corollary then serves as a categorical motivation for this restricted definition. For now, I leave this idea for future research.

Acknowledgement

I owe much gratitude to Jaap van Oosten for his patience and for giving me so much freedom while working on this thesis. Thanks to several valuable discussions and his contributions, I have been able to work out ideas of my own, which has been a great experience. He has also been a great mentor for the past years, allowing me to fiddle about, but giving me honest advise whenever I needed it.

[1] Update (June 10, 2014): In fact, I now claim to have a proof of this.

Bibliography

[BGO71] M. BARR, P. GRILLET, and D. V. OSDOL, *Exact Categories and Categories of Sheaves*, *Lecture Notes in Mathematics* **236**, Springer-Verlag, 1971.

[CF68] H. B. CURRY and R. FEYS, *Combinatory logic. Vol. I, With two selections by William Craig. Second printing. Studies in Logic and the Foundations of Mathematics*, North-Holland Publishing Co., Amsterdam, 1968.

[FS90] P. J. FREYD and A. SCEDROV, *Categories, allegories*, *North-Holland Mathematical Library* **39**, North-Holland Publishing Co., Amsterdam, 1990. MR 1071176 (93c:18001).

[Hin78] P. G. HINMAN, *Recursion-theoretic hierarchies*, Springer-Verlag, Berlin, 1978, Perspectives in Mathematical Logic. MR 499205 (82b:03084).

[Hof] P. HOFSTRA, *Partial combinatory algebras and realizability toposes*, Lecture notes for a tutorial at FMCS 2004 in Kananaski.

[HvO03] P. HOFSTRA and J. VAN OOSTEN, Ordered partial combinatory algebras, *Math. Proc. Cambridge Philos. Soc.* **134** (2003), 445–463. MR 1981211 (2004e:03103). `http://dx.doi.org/10.1017/S0305004102006424`. Available at `http://dx.doi.org/10.1017/S0305004102006424`.

[Hyl82] J. M. E. HYLAND, The effective topos, in *The L.E.J. Brouwer Centenary Symposium (Noordwijkerhout, 1981), Stud. Logic Foundations Math.* **110**, North-Holland, Amsterdam, 1982, pp. 165–216. MR 717245 (84m:03101).

[HJP80] J. M. E. HYLAND, P. T. JOHNSTONE, and A. M. PITTS, Tripos theory, *Math. Proc. Cambridge Philos. Soc.* **88** (1980), 205–231. MR 578267 (81i:03102). `http://dx.doi.org/10.1017/S0305004100057534`. Available at `http://dx.doi.org/10.1017/S0305004100057534`.

[Joh13] P. JOHNSTONE, Geometric morphisms of realizability toposes, *Theory Appl. Categ.* **28** (2013), No. 9, 241–249. MR 3065947.

[Joh02a] P. T. JOHNSTONE, *Sketches of an elephant: a topos theory compendium. Vol. 1*, *Oxford Logic Guides* **43**, The Clarendon Press Oxford University Press, New York, 2002. MR 1953060 (2003k:18005).

[Joh02b] P. T. JOHNSTONE, *Sketches of an elephant: a topos theory compendium. Vol. 2*, *Oxford Logic Guides* **44**, The Clarendon Press Oxford University Press, Oxford, 2002. MR 2063092 (2005g:18007).

[Kle59] S. C. KLEENE, Recursive functionals and quantifiers of finite types. I, *Trans. Amer. Math. Soc.* **91** (1959), 1–52. MR 0102480 (21 #1273).

[LvO13] S. LEE and J. VAN OOSTEN, Basic subtoposes of the effective topos, *Ann. Pure Appl. Logic* **164** (2013), 866–883. MR 3056301. http://dx.doi.org/10.1016/j.apal.2013.04.001. Available at http://dx.doi.org/10.1016/j.apal.2013.04.001.

[Lon95] J. R. LONGLEY, *Realizability Toposes and Language Semantics*, Ph.D. thesis, University of Edinburgh, 1995.

[Lon05] J. R. LONGLEY, Notions of computability at higher types. I, in *Logic Colloquium 2000*, *Lect. Notes Log.* **19**, Assoc. Symbol. Logic, Urbana, IL, 2005, pp. 32–142. MR 2143877 (2006e:03062).

[MLM94] S. MAC LANE and I. MOERDIJK, *Sheaves in geometry and logic*, *Universitext*, Springer-Verlag, New York, 1994, A first introduction to topos theory, Corrected reprint of the 1992 edition. MR 1300636 (96c:03119).

[Men00] M. MENNI, *Exact completions and toposes*, Ph.D. thesis, University of Edinburgh, 2000. Available at http://www.lfcs.inf.ed.ac.uk/reports/00/ECS-LFCS-00-424/.

[vO] J. VAN OOSTEN, Realizability with a Local Operator of A.M. Pitts, *Theoretical Computer Science*, To appear.

[vO06] J. VAN OOSTEN, A general form of relative recursion, *Notre Dame J. Formal Logic* **47** (2006), 311–318 (electronic). MR 2264700 (2007j:03022). http://dx.doi.org/10.1305/ndjfl/1163775438. Available at http://dx.doi.org/10.1305/ndjfl/1163775438.

[vO08] J. VAN OOSTEN, *Realizability: an introduction to its categorical side*, *Studies in Logic and the Foundations of Mathematics* **152**, Elsevier B. V., Amsterdam, 2008. MR 2479466 (2010c:03044).

[OM07] J. V. OOSTEN and I. MOERDIJK, *Topos Theory*, Lecture notes to a master class course, 2007. Available at http://www.staff.science.uu.nl/~ooste110/syllabi/toposmoeder.pdf.

[Pho89] W. PHOA, Relative computability in the effective topos, *Math. Proc. Cambridge Philos. Soc.* **106** (1989), 419–422. MR 1010365 (90h:18004). http://dx.doi.org/10.1017/S0305004100068146. Available at http://dx.doi.org/10.1017/S0305004100068146.

[Pit81] A. PITTS, *The Theory of Triposes*, Ph.D. thesis, Cambridge University, 1981. Available at http://www.cl.cam.ac.uk/~amp12/papers/thet/thet.pdf.

[Rog87] H. ROGERS, JR., *Theory of recursive functions and effective computability*, second ed., MIT Press, Cambridge, MA, 1987. MR 886890 (88b:03059).

[Sch24] M. SCHÖNFINKEL, Über die Bausteine der mathematischen Logik, *Math. Ann.* **92** (1924), 305–316. MR 1512218. http://dx.doi.org/10.1007/BF01448013. Available at http://dx.doi.org/10.1007/BF01448013.

[Wag69] E. G. WAGNER, Uniformly reflexive structures: On the nature of gödelizations and relative computability, *Trans. Amer. Math. Soc.* **144** (1969), 1–41. MR 0249297 (40 #2543).

Index