Partial Combinatory Algebras and Realizability Toposes

Pieter J.W. Hofstra
University of Ottawa
Department of Mathematics and Statistics
585 King Edward Avenue
K1N 5N6 Ottawa, ON, Canada.

June 9, 2004

Abstract

These are the lecture notes for a tutorial at FMCS 2004 in Kananaskis. The aim is to give a first introduction to Partial Combinatory Algebras and the construction of Realizability Toposes. The first part, where Partial Combinatory Algebras are discussed, requires no specific background (except for some of the examples perhaps), although familiarity with combinatory logic and lambda calculus will not hurt. The second part on realizability toposes presupposes some knowledge of category theory; more specifically, we will assume that the reader knows what a topos is. Apart from that the material is self-contained.

1 Partial Combinatory Algebras

We give the basic definitions and properties of Partial Combinatory Algebras in the first subsection. Next, we discuss some of the important examples. Finally, we touch upon the theory of Partial Combinatory Algebras.

1.1 Partial Applicative Structures and Combinatory Completeness

We first introduce the basic concept of a Partial Applicative Structure, which may be viewed as a universe for computation. Then look at terms over an applicative structure, we formulate what it means for an algebraic function to be representable and state the fundamental property of Combinatory Completeness. Finally, we present a simple test for Combinatory Completeness.

Definition 1.1 (Partial Applicative Structure) A *Partial Applicative Structure*, or *PAS* for short, is a set A equipped with a partial binary operation

$$\bullet : A \times A \rightharpoonup A.$$

For elements $a, b \in A$, we think of $a \bullet b$ (which we often abbreviate to ab) as "a applied to b". The partiality of $\bullet$ means that this application need not always be defined. If $(a, b) \in dom(\bullet)$, then we write $a \bullet b \downarrow$ or $ab \downarrow$. Otherwise, we write $ab \uparrow$.

Some more conventions and terminology: we usually omit brackets, assuming associativity to the left. So: abc stands for $(ab)c$. Also, for two expressions x, y, we write $x \simeq y$ to indicate that x is defined whenever y is, in which case they are equal. Finally, we say that A is *total* if $dom(\bullet) = A \times A$.

Even though these Partial Applicative Structures do not possess many interesting properties due to the lack of axioms, they already highlight one of the key features of combinatorial structures, namely the fact that we have a domain of elements that can act both as functions and as arguments. This behaviour goes under the heading of *polymorphism*, and can be traced back to Von Neumann's idea that programs (functions, operations) live in the same realm and are represented in the same way as the data (arguments) that they act upon. In particular, programs can act on programs. In a PAS, the left-hand argument of $\bullet$ is the element that is to be regarded as a function, whereas the right-hand argument is to be viewed as the argument.

Next, we wish to single out those PASs that are interesting from a computational point of view. This is achieved by adding the requirement that a certain class of functions is representable.

Let $V = \{x_0, x_1, \ldots\}$ be a countable set of fresh variables. We define $\mathcal{T}(A)$, the set of terms over A, to be the least set such that

- $A \subseteq \mathcal{T}(A)$
- $V \subseteq \mathcal{T}(A)$
- $t, t' \in \mathcal{T}(A) \Rightarrow (tt') \in \mathcal{T}(A)$.

We think of these terms as representatives of partial algebraic functions (in several variables). If $t \in \mathcal{T}(A)$, then we write $FV(t)$ for the set of variables that occur in t.

Definition 1.2 Let $t \in \mathcal{T}(A)$ be a term with $FV(t) \subseteq \{x_0, \ldots x_n\}$ and $a \in A$ an element of A. We say that *a represents t* if for all $a_0, \ldots, a_n \in A$

1. $aa_0 \cdots a_{n-1}\downarrow$
2. $t[a_0/x_0, \ldots a_n/x_n] \simeq aa_0 \cdots a_n$.

In words, a represents t if a is a total function in the first n arguments, and, as $n+1$-ary partial function, is equal to t. Note that there may be different elements representing the same term t.

Definition 1.3 (Combinatory Completeness) A partial applicative structure A is *Combinatory Complete* if every term $t \in \mathcal{T}(A)$ can be represented by some $a \in A$.

Now we can state the definition of a Partial Combinatory Algebra:

Definition 1.4 (Partial Combinatory Algebra) Let A be a PAS. Then A is a Partial Combinatory Algebra, PCA for short, if A is Combinatory Complete.

It must be said that this is not the definition that is usually encountered in the literature; a simplification (due to Curry) can be obtained by showing that combinatory completeness is already ensured by two of its instances.

Proposition 1.5 *Suppose a PAS A has elements k, s such that*

- $kab \simeq a$ *for all* $a, b \in A$
- $sa\downarrow, sab\downarrow$ *and* $sabc \simeq ac(bc)$ *for all* $a, b, c \in A$.

Then A is Combinatory Complete.

Proof. The proof is based on the fact that one can imitate lambda-abstraction as follows. Let t be a term, and x a variable. By induction on the structure of t, we define a new term $\lambda^* x.t$ such that $FV(\lambda^* x.t) = FV(t) \backslash x$:

$$\begin{aligned} \lambda^* x.x &= skk \\ \lambda^* x.t &= kt \quad \text{if } y \notin FV(t) \\ \lambda^* x.tt' &= s(\lambda^* x.t)(\lambda^* x.t') \end{aligned}$$

Next, one proves by induction that $(\lambda^* x.t)a = t[x/a]$ for all $a \in A$. □

As an example, consider the identity function I on A. This function can be obtained from the combinators k, s as $I = skk$. Another example: the second projection $(x, y) \mapsto y$ can be defined as sk. Finally, the function $(x, y) \mapsto xy$ (the application function) is represented by $sk(skk)$.

Remarks. One might wonder why the requirements $sa\downarrow$ and $sab\downarrow$ are present. Part of the answer is, that one does not get full combinatory completeness when they are omitted. On the other hand, one can show that if $sab\downarrow$ is left out, then the structure is equivalent, in a suitable sense, to a combinatory complete PAS with the same underlying set (but with different application).

Another point worth noting is that there may be many different choices for k and s. In some treatments, a particular choice for k and s is taken to be part of the structure of a PCA, whereas in others the property of combinatorial completeness (or the existence of k and s) is required. This is dependent as much on one's taste as on the applications (see also the remarks on morphisms).

1.2 Examples of PCAs

Before exploring some more advanced features of PCAs, we give four important examples: Kleene's recursion-theoretic PCA, the Graph Model, Engeler's Graph Algebra and Scott's D_∞ model.

Kleene's Recursion-Theoretic PCA $\mathcal{K}_1$. The underlying set of $\mathcal{K}_1$ is $\mathbb{N}$, the set of natural numbers. On this set we define a partial application by

$$n \bullet m \simeq \{n\}(m),$$

where the right-hand side stands for the application of the partial recursive function coded by n applied to the number m. The verification that this is a PCA reduces to checking that k and s are partial recursive.

This is the most basic example of a PCA, in a sense that will be made more precise later. Also, it is the basic building block of the Effective Topos, to be discussed in section 3.

Graph Model $\mathcal{P}\omega$. The Graph Model was the first model for untyped lambda calculus, and is due to Scott and Plotkin, independently. When constructing a model for polymorphism, one is confronted with the difficulty that in set theory, the cardinality of the function space C^C of a set C is always larger than that of C itself. This problem is solved by adding structure to the sets and reducing the size of the function space, only regarding functions which are continuous for a suitable topology.

We will not just sketch the graph model, but show how one can build PCAs from certain complete partial orders.

Definition 1.6 (Directed Complete Partial Order) 1. A *Directed Complete Partial Order*, *DCPO* for short, is a partial order $(A, \leq)$ which has a bottom element $\perp$ and which has suprema for all directed subsets. This means that if $D \subseteq A$ has the property that for each $d, d' \in D$ there is a $d'' \in D$ with $d \leq d'', d' \leq d''$, then there is an element $\bigvee D \in A$ such that (i) $\bigvee D \geq d$, for all $d \in D$ and (ii) for any $a \in A$ satisfying $a \geq d$ for all $d \in D$ we have that $\bigvee D \leq a$.

2. A homomorphism $\mu : (A, \leq) \to (B, \leq)$ of DCPOs is an order-preserving function $\mu : A \to B$ such that μ preserves suprema of directed subsets, i.e. for directed $D \subseteq A$ we ask that $\mu(\bigvee D) = \bigvee\{\mu(d) | d \in D\}$.

Every DCPO $(A, \leq)$ carries a topology, where a subset $U \subseteq A$ is open if and only if

- U is upwards closed, i.e. if $u \in U, v \geq u$ then $v \in U$;
- if, for a directed subset D, $\bigvee D \in U$, then $U \cap D \neq \emptyset$.

One can now check that a homomorphism of DCPOs is precisely a continuous function for this topology.

For any DCPO A, the set of continuous functions $[A, A]$ is again a DCPO; all structure is taken pointwise.

Definition 1.7 (Reflexive DCPO) A DCPO A is *reflexive* if there is a diagram

$$\begin{tikzcd}[A,A] \arrow[r, bend right, "\phi"'] & A \arrow[l, bend right, "\psi"']\end{tikzcd} \qquad (1)$$

in which both ϕ and ψ are continuous and where $\psi \circ \phi = 1$.

In words, reflexivity means that the function space $[A, A]$ is a retract of A.

As an example of a reflexive DCPO, consider $\mathcal{P}\omega$, the powerset of the natural numbers, ordered by inclusion. This is clearly a DCPO where supremum is simply union. Recall that there exists a bijective pairing operation $\langle -, - \rangle : \omega \times \omega \to \omega$, and that there exists an enumeration of finite subsets of ω, which we denote by $b : \omega \to \mathcal{P}\omega$. Having fixed such operations, we can define maps $\phi : [\mathcal{P}\omega, \mathcal{P}\omega] \to \mathcal{P}\omega$ and $\psi : \mathcal{P}\omega \to [\mathcal{P}\omega, \mathcal{P}\omega]$ by

$$\begin{aligned}\phi(f) &= \{\langle n, m\rangle | m \in f(b(n))\} \\ \psi(U)(V) &= \{m | \exists n \in \omega : b(n) \subseteq V \& \langle n, m\rangle \in U\}\end{aligned}$$

The proof that these maps are continuous and that $\psi \circ \phi = 1$ is left as an exercise.

Now back to the construction of a PCA from a reflexive DCPO A. The existence of the pair of maps ϕ, ψ enables us to interpret elements of A as functions $A \to A$. This suggests the following definition of application on A:

$$a \bullet b =_{def} \psi(a)(b).$$

This application is total. It remains to be checked that we have combinatory completeness. But this follows straightforwardly from the fact that any continuous function $f : A \to A$ is representable by $\phi(f)$. Also note that the representable functions are precisely the continuous ones.

Remark. The above construction can be generalized: if, in any cartesian closed category, we have an object U such that U^U is a retract of U, then U carries a PCA structure.

Engeler's Graph Algebra. This example starts with a nonempty set A. We define inductively:

$$\begin{aligned}G_0(A) &= A \\ G_{n+1}(A) &= G_n(A) \cup \{(B, b) | B \subseteq G_n(A), B \text{ finite}, b \in G_n(A)\}\end{aligned}$$

Now put

$$G(A) = \bigcup_{n \in \mathbb{N}} (A).$$

Then the set $\mathcal{P}(G(A))$ is a PCA when we define application by

$$X \bullet Y = \{b \in G(A) | \exists B \subseteq Y : (B, b) \in X\}.$$

Scott's D_∞. Here we start with a complete partial order (abbreviated CPO) D, and we observe that there is an *embedding-projection pair* ϕ, ψ, as in

$$\begin{tikzcd}D \arrow[r, bend right, "\phi"'] & {[D,D]} \arrow[l, bend right, "\psi"']\end{tikzcd}$$

where $\phi(d) = (x \mapsto d)$, the constant function with value d and $\psi(f) = f(\bot)$. Both ϕ and ψ are morphisms of CPOs, and clearly we have $\psi \circ \phi = 1$. Now this can be iterated: define $D_0 = D$, $D_{n+1} = [D_n, D_n]$, and we obtain a diagram

$$\begin{tikzcd}D_0 \arrow[r, bend right, "\phi_0"'] & D_1 \arrow[l, bend right, "\psi_0"'] \arrow[r, bend right, "\phi_1"'] & D_2 \arrow[l, bend right, "\psi_1"'] & \cdots\end{tikzcd}$$

in the category of CPOs. Now define D_∞ to be the limit of the diagram of the D_i together with the ψ_i[1]. Explicitly, an element of D_∞ is a sequence $(d_0, d_1, \ldots)$, where each $d_i \in D_i$, and where $\psi_i(d_{i+1}) = d_i$. The CPO-structure is componentwise. Moreover, one can now show that the function space $[D_\infty, D_\infty]$ is isomorphic to D_∞, which is a sufficient condition for D_∞ to carry a combinatory algebra structure.

Other Examples. There are other kinds of PCAs which we will not go into here. Just to mention a few: Kleene's $\mathcal{K}_2$ has the set of functions $\mathbb{N} \to \mathbb{N}$ as underlying set (see [8]). Also, one can consider generalizations of PCAs, such as *ordered PCAs.* We will briefly return to these when discussing realizability toposes. Finally, one has *term models*, obtained by taking equivalence classes of (closed) terms of (some extension of) the lambda calculus.

1.3 The Theory of PCAs

We have a brief look at the properties of PCAs. The main purpose is to give the flavour of the kind of problems and ideas that one encounters when investigating PCAs.

Combinatory Logic and Lambda Calculus. When we consider *total* structures (which we simply call combinatory algebras, or CAs), then it is easily seen that each combinatory algebra is a model for combinatory logic. Also, as should be evident from the proof of Proposition 1.5, we can interpret lambda terms in a combinatory algebra A. However, this interpretation will not respect β-equality in general. If it does, then we call A a *lambda algebra.* So by definition, a lambda algebra is a combinatory algebra such that if $M =_\beta N$ in the lambda calculus, then $M = N$ in A.

If a lambda algebra satisfies the further requirement that $M = N$ implies $\lambda^* xM = \lambda^* xN$, then it is called a *lambda model.* The interpretation of lambda terms then also respects the ξ-rule in the lambda calculus.

Extensionality. A PCA A is called *extensional* if $\forall x(ax \simeq bx)$ implies that $a = b$. In words, if elements represent the same partial function, then they must be equal. Kleene's $\mathcal{K}_1$, for example, is not extensional, since there are many codes (programs) that compute the same function. In fact, every function has infinitely many representatives.

For PCAs obtained via DCPOs, like $\mathcal{P}\omega$, we have extensionality precisely when the continuous function space A^A is not just a retract of A, but in fact isomorphic to A.

The model D_∞ is extensional; the function space $D_\infty^{D_\infty}$ is isomorphic to D_∞.

As for Graph Algebras, these are not extensional but can be made extensional by quotienting out by a suitable equivalence relation. Such a procedure is known as "extensional collapse". In [2] it is shown that these structures are a special instance of the D_∞ models. In fact, graph algebras, made extensional, are precisely those D_∞ that one obtains by taking D to be a complete atomic Boolean algebra (i.e. a powerset).

It can be shown that for total combinatory algebras one has the following proper inclusions

$$CAs \subset \lambda\text{-algebras} \subset \lambda\text{-models} \subset \text{extensional CAs}.$$

For more on this, we refer to [2, 1].

Totality and Completability. We already saw that a PCA A is called *total* if $ab\downarrow$ for all $a, b \in A$. Scott's $\mathcal{P}\omega$, for example, is total, whereas Kleene's $\mathcal{K}_1$ is not. Now A is called *completable* if it can be embedded into a total PCA. Here, we mean by embedding an injective map preserving the application and the combinators k, s. We mention the following facts (and refer for details to [2]): first, if a PCA has *unique head normal forms* then it is completable. We define a PCA A to have unique head normal forms if for any $x, y \in A$ we have that the elements k, s, kx, sx and sxy are all different and moreover that $sxy = sx'y'$ implies $x = x'$ and $y = y'$. Second, if a PCA is extensional but not total then it is not completable.

A related type of question that one can ask for PCAs is whether it contains sub-PCAs, which are total and/or extensional.

[1]Alternatively, one could take the colimit of the D_i together with the ϕ_i; this situation is known as a *limit-colimit coincidence.*

Lambda Calculus, Natural Numbers and Recursion Theory. As we observed in the proof of Proposition 1.5, one can use the combinators k and s to mimick λ-abstaction inside a PCA. Anyone familiar with the λ-calculus will recognize that this guarantees expressive power. In particular, one can define the natural numbers inside a PCA (as a corollary, every non-trivial PCA is at least countably infinite) and recursive functions are representable. It must be noted that there may be many different codings of the natural numbers in a PCA (just as there may be many different choices for k and s). A particular given coding is also called a *choice of numerals*. One possibility is: $n = \lambda^* x \lambda^* y.x^n y$.

Homomorphisms? The question what a good notion of homomorphism of PCAs is does not have a unique answer. In fact, this is highly dependent on one's viewpoint and the applications one has in mind. People who are interested in CAs *qua* models of combinatory logic and lambda calculus, will usually define a homomorphism of CAs to be a homomorphism of models, i.e. a function preserving application and combinators on the nose (note that this presupposes that one takes the combinators k and s to be part of the structure). This definition has the advantage that combinatory algebras form an algebraic variety, so that all kinds of constructions such as products, quotients and polynomial algebras are possible. On the other hand, if you wish to regard (P)CAs has applicative structures with an additional property (combinatory completeness) then it is not so natural to have k and s as part of your structure, and a more tempting definition of homomorphism $\phi : A \to B$ might be: ϕ preserves application, and if $t \in \mathcal{T}(A)$ is a term represented by $a_t \in A$, then $\overline{\phi}(t)$ is represented by $\phi(a_t)$. Here, $\overline{\phi} : \mathcal{T}(A) \to \mathcal{T}(B)$ is the map induced by ϕ.

In the next section we will outline why a much more relaxed notion of homomorphism is desired when considering PCAs as building blocks for realizability toposes.

2 Realizability Toposes

We are now going to show how a PCA A gives rise to a topos $\mathbf{RT}(A)$, called the *Realizability Topos over A*. This is done in a couple of steps: first we define some structure on the powerset of A. Then we show how this structure gives rise to a certain **Set**-indexed category. We discuss some of the properties of this indexed category, which make it into what is called a *tripos*. Finally, we show how to build a topos from a tripos.

Next, some of the properties of realizability toposes will be discussed: we explain the difference with Grothendieck toposes and have a look at arithmetic and the relation to realizability. Finally, we briefly address the functoriality of the construction.

2.1 From PCA to Tripos

Structure on $\mathcal{P}A$. We fix a PCA A and consider its powerset $\mathcal{P}A$. By standard tricks, one can define a pairing operation $\langle -, - \rangle : A \times A \to A$ with unpairings $(-)_0$ and $(-)_1$ satisfying $\langle a, b \rangle_0 = a, \langle a, b \rangle_1 = b$. Now define two operations $\wedge, \Rightarrow : \mathcal{P}A \times \mathcal{P}A \to \mathcal{P}A$. For $U, V \subseteq A$:

$$\begin{aligned} U \wedge V &=_{def} \{\langle u, v \rangle | u \in U, v \in V\} \\ U \Rightarrow V &=_{def} \{a \in A | \forall u \in U : au\downarrow \& au \in V\} \end{aligned}$$

Also observe that $\mathcal{P}A$ has a bottom element (the empty set) denoted $\bot$, and a top element (A itself) denoted $\top$.

Non-standard Predicates. We now extend the structure on $\mathcal{P}A$ to $\mathcal{P}A^X$, where X is an arbitrary set. For $\phi, \psi : X \to \mathcal{P}A$, define

$$\begin{aligned} (\phi \wedge \psi)(x) &=_{def} \phi(x) \wedge \psi(x) \\ (\phi \Rightarrow \psi)(x) &=_{def} \phi(x) \Rightarrow \psi(x). \end{aligned}$$

We preorder $\mathcal{P}A^X$ by putting, for $\phi, \psi : X \to \mathcal{P}A$:

$$\phi \vdash_X \psi \text{ if and only if } \bigcap_{x \in X} (\phi \Rightarrow \psi)(x) \neq \emptyset.$$

Equivalently, $\phi \vdash_X \psi$ if and only if there exists an $a \in A$ such that for all $x \in X$ and all $b \in \phi(x)$ we have $ab \downarrow$ and $ab \in \psi(x)$. In this case, we often refer to a as a *realizer* for $\phi \vdash_X \psi$.

A good way to think of this preorder is to consider X as a *type*, $\mathcal{P}A$ as a set of (non-standard) truth-values and $\phi : X \to \mathcal{P}A$ as a *non-standard predicate with free variable of type* X. Thus for each $x \in X$, think of $\phi(x) \in \mathcal{P}A$ as the extent to which ϕ is true of x.

The preorder relation can now be thought of as *entailment*. Moreover, the operations $\wedge$ and $\Rightarrow$ play the role of conjunction and implication. One readily verifies that $\wedge$ defines a meet on $\mathcal{P}A^X$ and that $\Rightarrow$ defines a Heyting implication.

There is more structure: the preorder $\mathcal{P}A^X$ has a top and bottom element, given by the constant function $x \mapsto \top$ and $x \mapsto \bot$, respectively. One can define disjunction and negation. A preorder like $\mathcal{P}A^X$ which has $\top, \bot, \wedge, \vee, \Rightarrow, \neg$ is called a *Heyting Pre-Algebra*.

Reindexing and Quantification. Since the above constructions made no assumptions about the set X, we see that we have an operation, which assigns to each set X a Heyting Pre-Algebra $\mathcal{P}A^X$. Furthermore, if $f : X \to Y$ is any function between sets, then f induces by composition a map

$$f^* : \mathcal{P}A^Y \to \mathcal{P}A^X, \quad (\phi : Y \to \mathcal{P}A) \mapsto (\phi \circ f : X \to \mathcal{P}A).$$

It is not hard to see that the induced map f^* preserves the preorder and all the connectives. So altogether we have a contravariant functor from **Set** to the category of Heyting Pre-Algebras. Such a structure is called a **Set**-*indexed* Heyting Pre-Algebra. One thinks of the reindexing functors f^* as a relabelling of the variables.

The reindexing maps f^* have both adjoints $\exists_f$ and $\forall_f$. These are given by (for $\phi : X \to \mathcal{P}A$)

$$\begin{aligned}(\exists_f \phi)(y) &= \bigcup_{f(x)=y} \phi(x) \\ (\forall_f \phi)(y) &= \bigcap_{f(x)=y} (A \Rightarrow \phi(x)).\end{aligned}$$

As suggested by their notation, these adjoints serve to interpret existential and universal quantification.

These quantifiers satisfy the so-called Beck-Chevalley Condition (BCC): for any pullback square in **Set**:

$$\begin{tikzcd} A \arrow[r, "q"] \arrow[d, "p"'] & B \arrow[d, "g"] \\ X \arrow[r, "f"'] & Y \end{tikzcd}$$

the canonical map $\Sigma_q p^* \to g^* \Sigma_f$ is an isomorphism. This guarantees that quantification is well-behaved with respect to substitution.

Generic Predicates. There is one predicate that plays a special role, namely the identity function $Id : \mathcal{P}A \to \mathcal{P}A$. It has the special property that for every predicate $\phi : X \to \mathcal{P}A$ there is a map $f_\phi : X \to \mathcal{P}A$ such that $\phi \cong f_\phi^*(Id)$. In words, every predicate can be obtained by relabelling the identity predicate along some map. Because of this, we call $Id : \mathcal{P}A \to \mathcal{P}A$ a generic predicate.

Triposes. A **Set**-indexed Heyting Pre-Algebra which has left and right adjoints to reindexing functors satisfying the BCC and has a generic predicate is called a *tripos*. This is an acronym for Topos Representing Indexed PreOrdered Set. The precise definition is a bit more subtle than what we've seen, and we refer to [9, 5] for details and many more examples of triposes. The intuition that should be kept in mind is that a tripos is a setting in which one can interpret intuitionistic, higher-order typed logic *without equality*.

2.2 From Tripos to Topos

The second step in our construction of a realizability topos is the tripos-to-topos construction. We will not treat this in full generality (all details and related issues can be found in [9, 5]), but just outline the case when the tripos arises from a PCA A as described above. The idea is, that the construction adds a non-standard equality predicate to every type.

First, some notation. Remember that for $\phi, \psi : X \to \mathcal{P}A$ we defined $\phi \vdash \psi$ if and only if for some $a \in A$, for all $x \in X$ and for all $b \in \phi(x)$ we have $ab\downarrow$ and $ab \in \psi(x)$. To stress that x is the relevant variable here, we will write $\phi(x) \vdash_x \psi(x)$. We will also extend this notation to several variables, e.g. if $\phi : X \times Y \to \mathcal{P}A$ and $\psi : Y \times Z \to \mathcal{P}A$, then $\phi(x,y) \vdash_{x,y,z} \psi(y,z)$ means that there is an $a \in A$ such that for all $x \in X, y \in Y, z \in Z$ and for all $b \in \phi(x,y)$ we have $ab\downarrow$ and $ab \in \psi(y,z)$.

Objects of RT(A). The objects of $\mathbf{RT}(A)$ are defined to be pairs $(X, =_X)$, where X is a set, and $=_X$ is a function $X \times X \to \mathcal{P}A$, which we think of as a non-standard equality predicate on X. Instead of $=_X (x, x')$ we write $x =_X x'$. We require:

- $x =_X x' \vdash_{x,x'} x' =_X x$ (Symmetry)
- $x =_X x' \wedge x' =_X x'' \vdash_{x,x',x''} x =_X x''$ (Transitivity)

We do not require reflexivity, since that would amount to a uniform realization of existence. Note that there can be many different equality predicates on the same set X.

Morphisms of RT(A). Let $(X, =_X)$ and $(Y, =_Y)$ be two objects. We first define what a functional relation from $(X, =_X)$ to $(Y, =_Y)$ is. This is a predicate $F : X \times Y \to \mathcal{P}A$ subject to the conditions

- $F(x,y) \vdash_{x,y} x =_X x \wedge y =_Y y$
- $F(x,y) \wedge x =_X x' \wedge y =_Y y' \vdash_{x,x',y,y'} F(x',y')$
- $F(x,y) \wedge F(x,y') \vdash_{x,y,y'} y =_Y y'$
- $x =_X x \vdash_x \exists_y F(x,y)$

Next, we say that two such functional relations F, G are equivalent if and only if

$$F(x,y) \vdash_{x,y} G(x,y) \quad \text{and} \quad G(x,y) \vdash_{x,y} F(x,y).$$

Finally, a morphism from $(X, =_X)$ to $(Y, =_Y)$ is defined to be an equivalence class of functional relations from $(X, =_X)$ to $(Y, =_Y)$.

2.3 Features of Realizability Toposes

We will now investigate the resulting topos $\mathbf{RT}(A)$ a bit. First we discuss some facts about realizability toposes in general and then we look at a special case, the Effective Topos. Finally, the question whether the operation $A \mapsto \mathbf{RT}(A)$ is functorial is addressed.

RT(A) as a Topos. Most toposes, such as sheaves on a topological space, presheaves, or sheaves on a locale, are *Grothendieck Toposes*. There are several equivalent definitions of this notion. For example, one can say that a topos $\mathcal{E}$ is Grothendieck if it is cocomplete and has a set of generators. As a result, a Grothendieck topos comes equipped with a geometric morphism $\gamma : \mathcal{E} \to \mathbf{Set}$. The direct image functor γ_* is the representable functor $\mathcal{E}(1, -)$, and the inverse image functor γ^* is the constant objects functor $X \mapsto \sum_{x \in X} 1$.

For realizability toposes the situation is quite different. Of course there is a global sections functor $\Gamma : \mathbf{RT}(A) \to \mathbf{Set}$. We can describe this functor more explicitly as follows: for an object $(X, =_X)$ of $\mathbf{RT}(A)$, we define a partial equivalence relation $\sim$ on X, where $x \sim x'$ if $x =_X x' \neq \emptyset$. Then $\Gamma(X, =_X)$ is the set of inhabited equivalence classes of X.

Instead of being a direct image functor, Γ is an inverse image functor. Its right adjoint ∇ (which is also called a constant objects functor) sends a set X to the object $(X, =_\nabla)$ where

$$x =_\nabla x' = \begin{cases} \emptyset & \text{if } x \neq x' \\ A & \text{if } x = x'. \end{cases}$$

The counit of the adjunction $\Gamma \dashv \nabla$ is easily seen to be an isomorphism, so that **Set** is a subtopos of $\mathbf{RT}(A)$. The corresponding topology is the double-negation topology, exhibiting **Set** as the world of classical mathematics inside $\mathbf{RT}(A)$.

Another point worth observing is that $\mathbf{RT}(A)$ is not cocomplete (unless A is the trivial one-point PCA, in which case $\mathbf{RT}(A) \simeq \mathbf{Set}$). One way of seeing why is by considering the A-indexed coproduct of copies of 1 and the object $(A, =_A)$, where $[a =_A a'] = \{a\}$ if $a = a'$ and $[a = a'] = \emptyset$ otherwise. Now every function $1 \to A$ induces a map $1 \to (A, =_A)$ in $\mathbf{RT}(A)$. If $\sum_{a \in A} 1$ would exist, then every function $A \to A$ would induce a map $\sum_{a \in A} 1 \to (A, =_A)$, amounting to the representability of all functions $A \to A$, which is impossible for cardinality reasons.

The Effective Topos. When one starts out with Kleene's PCA $\mathcal{K}_1$, then the resulting topos is known as the Effective Topos, **Eff**. This was the first realizability topos that was investigated, and its discovery is due to Martin Hyland (see [4]).

So far, we have not explained where the nomenclature "realizability topos" comes from. Realizability is an interpretation for a formal system of intuitionistic arithmetic called *Heyting Arithmetic*. This interpretation, devised by S.C. Kleene in 1945 (see [7, 10]), is based on recursion theory and has had many applications in proof theory and metamathematics of constructive systems. Since the PCA $\mathcal{K}_1$ is also based on recursive functions one can expect a relation between realizability and the Effective Topos. Indeed, **Eff** has a natural number object, so one can interpret arithmetic. Now one can prove that a sentence in the language of arithmetic is valid under the realizability interpretation precisely when it is true in the Effective Topos.

For one thing, this places the study of realizability on a topos-theoretic context. Since toposes allow for the interpretation of higher-order logic, we obtain a natural generalization of realizability to higher-order arithmetic.

Also, we can study certain logical principles in the Effective Topos. For example, in **Eff** every function from $\mathbb{N}$ to $\mathbb{N}$ is recursive! And when one looks at the real numbers, one will find that they provide a model for constructive analysis.

Finally, there is a famous result stating that there exists an internal category in **Eff** which is small, complete, but not a poset. For a detailed account, see [6]. For ordinary categories, this is impossible by a theorem by Peter Freyd. This category is the internalization of a full subcategory of **Eff**, called PER (Partial Equivalence Relations), which plays a fundamental role in the semantics of programming languages.

Homomorphisms. In topos theory, the natural notion of a map between toposes is that of a *geometric morphism*. Given the construction of a topos $\mathbf{RT}(A)$ from a PCA A, one might wonder which homomorphisms of PCAs induce geometric morphisms between the realizability toposes. This problem is addressed in [3]. It turns out that a very different notion of homomorphism is needed than the "model-theoretic" homomorphism. First, we consider a generalization of PCAs, which we call *Ordered PCAs*. This is a poset with application satisfying $ab\downarrow, a' \leq a, b' \leq b$ implies $a'b'\downarrow$ and $a'b' \leq ab$, such that the combinatory completeness property holds (up to inequality rather than equality). The construction of a realizability topos goes through without much modification. Now a map $A \to B$ of ordered PCAs is a function $f : A \to B$ which preserves the ordering and application *up to a realizer*, in the sense that for some $p, q \in A$ we have

$$a \leq b \quad \Rightarrow \quad p \bullet f(a)\downarrow \quad \& \quad p \bullet f(a) \leq f(b)$$

and

$$ab\downarrow \quad \Rightarrow \quad q \bullet f(a) \bullet f(b)\downarrow \quad \& \quad q \bullet f(a) \bullet f(b) \leq f(ab).$$

Moreover, on the category **OPCA** thus obtained, one has a monad T, which acts on objects by taking non-empty downward closed subsets. Now one can prove that a map $A \to B$ in the Kleisli category of T is the same thing (up to natural isomorphism) as an exact functor $\mathbf{RT}(B) \to \mathbf{RT}(A)$ which respects ∇, the inclusion of **Set**. Finally, we identify a condition on maps $A \to B$ called *computational density* such that computationally dense maps $A \to B$ in the Kleisli category are the same (again up to isomorphism) as geometric morphisms $\mathbf{RT}(A) \to \mathbf{RT}(B)$. This shows that we can reduce the study of the category of realizability toposes to the study of the category **OPCA** of ordered PCAs.

References

[1] H.P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic*. North-Holland, 1981.

[2] I. Bethke. *Notes on Partial Combinatory Algebras*. PhD thesis, Universiteit van Amsterdam, 1988.

[3] P.J.W. Hofstra and J. van Oosten. Ordered partial combinatory algebras. *Mathematical Proceedings of the Cambridge Philosophical Society*, 134:445–463, 2003.

[4] J.M.E. Hyland. The effective topos. In A.S. Troelstra and D. Van Dalen, editors, *The L.E.J. Brouwer Centenary Symposium*, pages 165–216. North Holland Publishing Company, 1982.

[5] J.M.E. Hyland, P.T. Johnstone, and A.M. Pitts. Tripos theory. *Math. Proc. Camb. Phil. Soc.*, 88:205–232, 1980.

[6] J.M.E. Hyland, E.P. Robinson, and G. Rosolini. The discrete objects in the effective topos. *Proceedings of the London Mathematical Society*, 60:1–60, 1990.

[7] S.C. Kleene. On the interpretation of intuitionistic number theory. *Journal of Symbolic Logic*, 10:109–124, 1945.

[8] S.C. Kleene and R.E. Vesley. *The Foundations of Intuitionistic Mathematics, especially in relation to recursive functions*. North-Holland Publishing Company, 1965.

[9] A.M. Pitts. *The Theory of Triposes*. PhD thesis, Cambridge University, 1981.

[10] A.S. Troelstra. Realizability. In S.R. Buss, editor, *Handbook of Proof Theory*, pages 407–473. North-Holland, 1998.

Acknowledgement. I wish to express my gratitude towards Robin Cockett for organizing the FMCS 2004 conference and for inviting me to give this tutorial.